Springer Optimization and Its Applications 73

Giorgio Fasano and János D. Pintér *Editors*

Modeling and Optimization in Space Engineering

Springer

Springer Optimization and Its Applications

VOLUME 73

Aims and Scope
Optimization has been expanding in all directions at an astonishing rate during the last few decades. New algorithmic and theoretical techniques have been developed, the diffusion into other disciplines has proceeded at a rapid pace, and our knowledge of all aspects of the field has grown even more profound. At the same time, one of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field. Optimization has been a basic tool in all areas of applied mathematics, engineering, medicine, economics, and other sciences.

The series *Springer Optimization and Its Applications* publishes undergraduate and graduate textbooks, monographs, and state-of-the-art expository work that focus on algorithms for solving optimization problems and also study applications involving such problems. Some of the topics covered include nonlinear optimization (convex and nonconvex), network flow problems, stochastic optimization, optimal control, discrete optimization, multiobjective programming, description of software packages, approximation techniques and heuristic approaches.

For further volumes:
http://www.springer.com/series/7393

Giorgio Fasano • János D. Pintér
Editors

Modeling and Optimization in Space Engineering

Springer

Editors
Giorgio Fasano
Thales Alenia Space SpA
Turin, Italy

János D. Pintér
Pintér Consulting Services Inc.
Halifax, Nova Scotia
Canada

ISSN 1931-6828
ISBN 978-1-4614-4468-8 ISBN 978-1-4614-4469-5 (eBook)
DOI 10.1007/978-1-4614-4469-5
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012946722

MSC 2010 codes: 49-06, 05B40, 37N05, 37N40, 65K05, 70M20, 90-08, 90BXX, 90C11, 90C26, 90C29, 90C30

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Since the very beginnings of modern space exploration activities, the simultaneous consideration of key mission objectives and options—including technical feasibility, mission safety, and cost-efficiency aspects—has been essential. Space engineering projects have required, inter alia, the analysis and optimization of trajectories, fuel consumption, cargo handling, and other aspects of mission logistics, with paramount consideration given to crew and environmental safety. The experimental and commercial revenues expected from space activities such as the ones associated with the International Space Station have given rise to further challenging cost–benefit as well as risk analysis issues. The ambitious goals of future interplanetary explorations including manned missions will also require advanced analysis and optimization of the systems and resources involved.

While the necessary depth and quality of the decisions required in space engineering is ever increasing, we have also witnessed continuing innovation regarding both theoretical advances and ready-to-use tools for actual applications. The results of scientific innovation and algorithmic developments are supported and enhanced by today's advanced computational modeling and optimization environments. Since the earliest space engineering applications, the solution of increasingly hard optimization problems has become necessary, giving rise to complex analytical and computational issues. Until just a few years ago, the common numerical approaches were essentially limited to handle certain continuous (linear or convex nonlinear) optimization and linearly structured combinatorial and mixed integer-continuous optimization problems. Recent advances in the area of computational global optimization allow the handling of (often) more realistic non-convex problem formulations. Furthermore, the consideration of integer decision variables in more flexible nonlinear modeling frameworks gives rise to often even harder (again, non-convex) combinatorial and mixed integer-continuous optimization problems. The solution of such computational challenges is becoming gradually more viable as a direct result of algorithmic advances and software development. The book emphasizes these new paradigms, by providing an in-depth discussion of mathematical modeling and algorithmic aspects of real-world space engineering applications.

Classic and more recent space engineering problems—including cargo accommodation and object placement, flight control of satellites, system design and trajectory optimization, interplanetary transfers with deep-space maneuvers, low energy transfers, magnetic cleanliness modeling, propulsion system design, sensor system placement, space traffic logistics, and are discussed. Novel points of view related e.g., to computational global optimization and optimal control and to multidisciplinary design optimization are also given proper emphasis. A particular attention is paid (also) to scenarios expected in the context of future interplanetary explorations.

The literature dealing with advanced modeling issues and optimization problems arising in space engineering mostly consists of works that address specialists. At the same time, there are only a few works available that cover a wide range of technical aspects and applications and target a broader readership. The present edited volume is aimed at meeting some of the resulting needs, by discussing current and prospective applications of advanced optimization to handle some of the major challenges in space engineering in an expository manner. The contributing authors are well-recognized researchers and practitioners working in modeling and optimization related (also) to space engineering.

The book offers an in-depth exposition of the mathematical modeling, algorithmic and numerical solution aspects of all topics covered. A proper emphasis is placed upon recently developed modeling paradigms and computational tools, including global as well as local optimization, multidisciplinary design, and optimal control theory.

Modeling and Optimization in Space Engineering primarily addresses researchers and practitioners in the field of space engineering. It will be useful also for aerospace graduate and postgraduate students willing to broaden their horizon, by studying real-world applications and challenging problems that they will be likely to meet in their professional activities. The contributed chapters are more focused on practical aspects than on theoretical rigor: for the latter readers will be referred (as needed) to the cited extensive literature. With this in mind, researchers and practitioners in mathematical modeling, operations research, mathematical programming, optimization and optimal control will also benefit from the case studies presented.

The approaches discussed can be extended also to application areas that are not necessarily related to space applications. The book can be also used as a reference to assist researchers and practitioners in developing novel applications. Readers will obtain a broad overview of some of the most challenging space engineering operational scenarios of today and tomorrow. This aspect will also benefit managers in the aerospace field, as well as in other advanced industrial sectors.

Giorgio Fasano
Turin, Italy

János D. Pintér
Halifax, Nova Scotia, Canada

About the Editors

Giorgio Fasano is a researcher and practitioner at Thales Alenia Space with more than 25 years of experience in the field of optimization, in support to space engineering. He holds a MSc degree in Mathematics, he is also a Fellow of the Institute of Mathematics and its Applications, has been awarded the Chartered Mathematician (IMA, UK), as well as the Chartered Scientist (Science Council, UK) designations. He is a Co-editor of *Operations Research in Space and Air* (Ciriani et al., Eds., Kluwer Academic Publ., 2003), and author of several specialist publications on optimization in space. He was a Finalist at the EURO 2007 Excellence in Practice Award. He currently works as technical manager of the CAST (Cargo Accommodation Support Tool) project, funded by the European Space Agency. His interests include Mixed Integer Programming, Global Optimization, and Optimal Control.

János D. Pintér is a researcher and practitioner with four decades of experience in the area of systems modeling and optimization, including algorithm and software development. He holds MSc, PhD and DSc degrees in Mathematics, with specializations in the Operations Research area. He has authored and edited books including the award-winning monograph Global Optimization in Action. He also wrote nearly 200 journal articles, book chapters, and other technical documents. Dr. Pintér is an active member and officer of several professional organizations (CORS, EUROPT, HORS, INFORMS), and he serves on the editorial board of international professional journals including the Journal of Global Optimization. Dr. Pintér is the principal developer or co-developer of a range of nonlinear optimization software products. These products are used worldwide by researchers and practitioners in academia, and at business and research organizations. For more information, please visit www.pinterconsulting.com.

Acknowledgements

First of all, we wish to thank all contributing authors for their high-quality research work and diligent efforts to make the timely completion and publication of this volume possible. We also express our thanks to our coauthors of joint works cited in our own contributed chapters.

In addition to the review and editorial activities done by ourselves, several colleagues assisted us with valuable comments and suggestions. We wish to express our special thanks to Roberto Armellin, Pierluigi Di Lizia, Fabio Schoen, and Francesco Topputo.

One of the editors (GF) thanks also Gualtiero Brambati, Mario Cardano, Piero Messidoro, and Annamaria Piras of Thales Alenia Space Italia S.p.A., for their support of the research and development activities related to modeling and optimization in a range of space engineering applications. GF also wishes to thank Jane Alice Evans, Editorial Consultant, for her commitment and support of the book review activity.

We have been very glad to work with Vaishali Daimle, Senior Editor of Springer Science+Business Media for Mathematics on this book project, from its initial discussions to its completion.

Giorgio Fasano
Turin, Italy

János D. Pintér
Halifax, Nova Scotia, Canada

Contents

Chapter 1
Model Development and Optimization for Space Engineering: Concepts, Tools, Applications, and Perspectives

Giorgio Fasano and János D. Pintér

Abstract The theory and methodology of finding the best possible solution to a broad range of optimization problems has been of interest since the beginnings of modern operations research. The key theoretical results regarding important model types and algorithmic frameworks have been followed by optimization software implementations that are used to handle a large and still growing variety of applications. Our discussion is focused on the practice of nonlinear—specifically including also global and mixed integer optimization, in the context of space engineering applications. We review some of the prominent solution approaches, model development tools, and software implementations of optimization (solver) engines and then relate our discussion to selected applications in space engineering. The review portion of this work cites contributions by our coauthors to the present volume (Fasano and Pintér, Modeling and Optimization in Space Engineering, Springer Science + Business Media, New York, 2012) while also drawing on an extensive list of other sources.

Keywords Optimization models and methods • Continuous nonlinear (global and local) optimization • Mixed-integer linear and nonlinear models • Modeling languages and systems • Solution strategies and optimization software • LGO solver suite for nonlinear optimization • Current and prospective optimization applications in space engineering

MSC Classification (2000) 68 T20, 90 C11, 90 C30, 90 C59, 90 C90, 90-02, 90-08

G. Fasano
Thales Alenia Space Italia S.p.A., Str. Antica di Collegno 253, Turin 10146, Italy
e-mail: giorgio.fasano@thalesaleniaspace.com

J.D. Pintér (✉)
Pintér Consulting Services Inc., Halifax, NS, Canada
e-mail: janos.d.pinter@gmail.com

1.1 Introduction

1.1.1 Modeling and Optimization: An Operations Research Framework

Operations research (OR) provides a scientifically established objective framework and methodology to assist analysts and decision makers in finding feasible or optimized decisions across a vast range of decision-making issues. Decision problems can be frequently modeled by constrained optimization models: we want to find the best possible decision that satisfies a given set of constraints. In the present discussion, we shall consider the continuous optimization model defined by the following ingredients:

- x: decision vector, an element of the real Euclidean n-space $\boldsymbol{R}^n$
- l, u: explicit, finite n-vector bounds of x that define an interval ("box") in $\boldsymbol{R}^n$
- $f(x)$: continuous objective function, $f{:}\boldsymbol{R}^n \to \boldsymbol{R}$
- $g(x)$: m-vector of continuous constraint functions, $g{:}\boldsymbol{R}^n \to \boldsymbol{R}^m$

Applying these notations, a very general class of optimization models can be concisely formulated as follows:

$$\min f(x) \; x \in D := \{x : l \leq x \leq u, g(x) \leq 0\} \subset \boldsymbol{R}^n \tag{1.1}$$

In (1.1) all vector inequalities are interpreted component-wise: l, x, u, are all n-component vectors, and the 0 in $g(x) \leq 0$ denotes an m-component zero vector. The set of feasible vectors D is defined by l, u, and g. The set of the general constraints g could be empty, leading to a box-constrained model. Formally more general optimization models that include also $=$ and $\geq$ constraint relations and/or explicit lower bounds on the admissible constraint function values can be simply reduced to the concise model form (1.1).

This model evidently subsumes, e.g., all linear and convex nonlinear programming models, under corresponding additional specifications regarding the model functions f and g. It is also easy to demonstrate that (1.1) formally subsumes the entire class of pure combinatorial optimization problems formulated with binary decision variables (and thereby it also covers models with finitely bounded discrete variables). Consequently, all mixed integer-continuous optimization problems (formulated with both discrete and continuous decision variables) are also covered by the concise model form (1.1).

1.1.2 Nonlinear Optimization

The present discussion focuses on nonlinear models since these are essential tools in the context of space engineering applications—as clearly shown also by the contributed chapters briefly reviewed later on. Therefore we will assume that some or all of the functions f and g (the latter component-wise) are nonlinear. Additional analytical properties such as the convexity, Lipschitz continuity (implied, e.g., by smooth model structure), or continuous (first, second) differentiability of the model component functions are often verified or postulated, in order to apply certain types of numerical methods to handle the corresponding problem instance.

If we can assume that D is nonempty, then the optimal solution set X^* in the model (1.1) is non-empty, since this is implied by the continuity of f. This fundamental existence result does not mean that an element of X^* (or all of its elements) is (are) easy to find. In fact, the majority of practically motivated constrained optimization models require numerical solution approaches—as opposed to finding their solution by purely analytical or symbolic methods. This remark is certainly valid in the context of global optimization (GO). If we have to drop the convexity assumption regarding the model functions $\{f, g\}$ in (1.1), then the resulting model instance frequently will become multi-extremal.

To illustrate this point by an example, one can think of solving a square system of nonlinear equations $g(x) = 0$, g:$\boldsymbol{R^n} \to \boldsymbol{R^n}$ assuming that $x \in D := \{x: l \leq x \leq u\} \subset \boldsymbol{R^n}$. Then it is entirely possible that this system has no solutions at all or that it has one “true” (global) solution, as well as a number of (local) “pseudosolutions” with non-negligible residual errors. (There are also many other possible scenarios, of course, but we will consider the one above to illustrate the present discussion.) The possible multi-modality aspect of the problem implies that local scope optimization strategies may not work satisfactorily—unless started “sufficiently close” to the “true” solution. Technically, “sufficiently close” means to have access to a point in the region of attraction of the globally optimal solution point: this region is specific to the model instance and often also to the local scope optimization method used. Therefore the postulated “close guess” requirement may be elusive in the example mentioned: this explains why classical numerical methods may or may not work to handle this practically important problem type.

There are many other structurally similar optimization problems with a typically massive multimodal structure. A few important examples are the fitting of a nonlinear model to a given data set [1–4], data classification [2], or the finding of optimized object configurations [5–9]. Obviously, these general problem classes can be directly related also to numerous space engineering applications; several further examples will be reviewed later on primarily based on contributions to Ciriani et al. [10] and Fasano and Pintér [11]. Hence, in many such cases, one has to use global—as opposed to local—scope search strategies.

For completeness, first we list a selection of textbooks that present in-depth discussions of local nonlinear optimization models, methods, and their applications: consult, e.g., Bazaraa et al. [12], Peressini et al. [13], Bertsekas [148], Chong and Zak [14], Edgar et al. [15], Diwekar [16], Boyd and Vandenberghe [17], Hillier and Lieberman [18], and Nocedal and Wright [19].

In contrast to local search strategies, the objective of global optimization is to find the absolutely best solution of optimization models that (could) have also local optima. In recent decades, GO has become a mature field of mathematical programming. There are several hundred books written on the subject: consult, e.g., Horst and Pardalos [20], Horst and Tuy [21], Kearfott [22], Mockus et al. [23], Pintér [2], Floudas et al. [24], Strongin and Sergeyev [25], Pardalos and Romeijn [26], Zabinsky [27], and Liberti and Maculan [28]. For various GO applications in engineering design, consult e.g. Grossmann [29], Papalambros and Wilde [30], and Pintér [31]. The edited volumes Ciriani et al. [10] and Fasano and Pintér [11] are specifically devoted to space engineering and related applications: we will review work from these volumes later on.

For examples of in-depth GO review articles, we mention Neumaier [32] and Floudas and Gounaris [33]; the reviews by Pintér [34, 35] focus on software and test problems, including also extensive lists of applications with detailed numerical examples. Rios and Sahinidis [36] present a substantial comparative study of derivative-free nonlinear (often global) optimization methods and their software implementations.

The consideration of integer decision variables, possibly in combination with continuous ones, adds significant difficulty to handling the resulting global optimization problems. Grossmann [29], Floudas [37], Nowak [38], Tawarmalani and Sahinidis [39], and Li and Sun [40] discuss mixed-integer linear and nonlinear optimization models and solution strategies; see also the recent review by Burer and Letchford [41].

The above mentioned works advocate exact approaches to solve nonlinear optimization problems with corresponding theoretical (deterministic or stochastic) convergence properties. In many cases—due to the difficulty of the models to solve, the real or perceived lack of suitable software, and various other practical considerations—instead of exact approaches, heuristic strategies are proposed and applied. Therefore we also provide a few references related to prominent heuristic approaches, mostly to solve combinatorial—but also continuous—optimization problems. Consult, e.g., Michalewicz [42], Osman and Kelly [43], Glover and Laguna [44], Rudolph [45], Voss et al. [46], Rothlauf [47], Jones and Pevzner [48], and Michalewicz and Vogel [49]; see also the e-book by Weise [50] that is focused on heuristic strategies in the broader GO context.

1.1.3 Optimization Modeling Systems and Solver Engines

Many of the real-world problems tackled by an OR-based approach are complicated, and the "best possible" model development and solution procedure may not be clear at the beginning of the project. This general statement is certainly valid also for many (if not all) space engineering projects. Under such circumstances, interdisciplinary teams of decision makers, domain experts, and model and algorithm developers have to work together in an "iterative" fashion. The team repeatedly has to modify the model formulation and solution procedure until the resulting system model properly captures the essence of the decision problem, it can be adequately supported by data, its computational solution is viable, and the results are deployable in the real-world setting of the decision problem. The above facts and circumstances imply the essential need for suitable modeling systems (environments, languages) and corresponding robust and efficient optimization solvers.

In formulating and solving an optimization model in this context, a key aspect to consider is the choice of the software platform(s) to use. In many professional research and industrial environments, platform acceptance by the organization, compatibility with the available hardware and operating system platforms as well as with other software, ease of use, quality of software documentation, and technical support will often dominate other (perhaps more technical) aspects.

Driven by practical considerations and requirements, research engineers and scientists frequently may want "only" a high-quality solution to an optimization problem that can be obtained rather quickly, as opposed to finding a rigorously guaranteed solution in days, weeks, months, or even more time. However, let us also note here that there are cases when high-fidelity solutions are absolutely necessary almost regardless of the computational cost. Hence, the choice of the "most suitable" tools strongly depends on specific user criteria. The modeling environments and software revised below serve a broad range of users, including their optimization needs.

To start with the core optimization software implementations, there exists a range of compiler platform-based solvers, with more or less built-in model development functionality. Typically, the optimization engine itself is developed in C or FORTRAN, while the user interface and related features are implemented using C++, C#, Delphi, Java, Visual Basic, or perhaps some other language. Such solver implementations can be built directly into decision support systems and applications due to their customizability and often relatively small hardware + software "footprint." At the same time, these stand-alone solver implementations often require the most expertise to use. For reviews, consult, e.g., Leyffer and Mahajan [51] regarding software for nonlinear (mostly local) optimization and Pintér [52] on software for global optimization.

The core solver engines referred to above are frequently linked to optimization modeling languages (ML). An ML is a model development environment specifically tailored for the optimization of (possibly) complex, large-scale systems: consult, e.g., Kallrath [53]. MLs have a language compiler with model preprocessing, error-checking, and other model development facilities. MLs support the transparent creation of scalable model instances with corresponding data sets, assuming that the key model structure can be well represented. Thereby MLs assist the development of possibly very large models that are easier to maintain and to adapt to new scenarios as needed.

MLs incorporate as additional options a collection of high-performance solvers. These solver engines are collectively aimed at handling the range of linear and nonlinear, continuous, discrete, and mixed-integer optimization problems, often with tailored approaches to more special problem classes from the main optimization areas. Modeling environments often include a substantial library of illustrative models that can effectively assist users to develop their own models. Without going into details, we also mention the options and facilities of MLs that support solver testing and benchmarking.

Examples of prominent commercial modeling systems include AIMMS (by Paragon Decision Technology [146]), AMPL (AMPL LLC [141]), the Excel Solver Platform (Frontline Systems [142]), GAMS (GAMS Development Corporation), the IBM ILOG CPLEX Optimization Studio (IBM), LINGO (LINDO Systems [144]), and MPL (Maximal Software [145]). For current information regarding these MLs, visit the websites of the modeling system developers listed in the references.

There has been also a very remarkable development in the optimization context of high-level integrated scientific and technical computing (ISTC) systems such as Maple [54], *Mathematica* [55], and MATLAB [56]. These systems incorporate substantial ready-to-use function libraries, and they support the complete model development process by offering integrated computing, programming, project documentation, and model visualization facilities. ISTCs also offer optimization features, either as built-in functions or as add-on products. The online help facilities and tutorials of ISTCs assist rapid prototyping and modeling (also) in an interdisciplinary team context. We see a significant role for ISTCs in the development and solution of advanced optimization models with modules that can represent entire subsystem models. A typical application example is the optimal parameterization (calibration) of a system model that—for each studied parameter setting—requires the evaluation of an embedded numerical procedure, the solution of a set of differential or differential-algebraic equations, deterministic or stochastic simulation, and so on.

The modeling environments briefly reviewed above will meet the needs and requirements of different types of users to various extents. The main categories include educational users (instructors and students); research scientists, engineers, consultants, and other practitioners (possibly, but not necessarily equipped with an in-depth optimization related background); optimization experts, software application developers, and other "power users." The pros and cons of the individual software products—in terms of hardware and software platform demands, ease of use, model prototyping options, detailed code development and maintenance features, optimization model checking and processing tools, availability of solver options and other auxiliary tools, program execution speed, overall level of system integration, quality of related documentation and technical support, customization options, and communication with end users—make the corresponding modeling and solver approaches more or less attractive for the user groups listed above.

To illustrate the preceding discussion, in the next section, we introduce the LGO solver suite and its current implementations linked to most of the modeling systems reviewed earlier. The acronym LGO abbreviates a Lipschitz(-continuous) global optimizer program system.

1.2 The LGO Solver Suite for Nonlinear Optimization

1.2.1 Solver Options

LGO seamlessly integrates a suite of derivative-free global and local optimization strategies. The theoretical foundations of the global search components in LGO are described by Pintér [2]. Specifically, this book presents an in-depth discussion of both adaptive deterministic partition strategies and of stochastic search methods to solve GO problems under basic continuity or Lipschitz-continuity assumptions (as discussed briefly in Sect. 1.1). The theoretically exhaustive global search capability of these deterministic or stochastic strategies guarantees their global convergence (to a point of X^* or—under proper analytical conditions—to all points of X^*).

The LGO program system has been continuously developed since the late 1980s. LGO can be used as a compiler platform-based optimization engine or as a solver option linked to various modeling environments. Several of these implementations have been described in Pintér [34, 35, 57, 58], Pintér and Kampas [6, 59], Pintér et al. [60], as well as in platform-specific user manuals. Next, we shall briefly review the key features of the core LGO program system. For a more detailed description we refer to the current User's Guide [61].

The current LGO implementation integrates the following solver modules:

- Regularly spaced sampling, as a global presolver (RSS)
- Branch-and-bound global search method (BB)
- Global adaptive random search (GARS)
- Multi-start-based global random search (MS)
- Constrained local search (LS) by the generalized reduced gradient method

The global search methods implemented in LGO—with the exception of the recently added RSS module—are based on the exposition by Pintér [2]. The integration of RSS with LGO is discussed in detail by Pintér [62], and Pintér and Horváth [63]. The generalized reduced gradient algorithm is described in numerous textbooks: consult, e.g., Edgar et al. [15]. Therefore—and also due to the intended overall scope of this chapter—only a brief overview of the solver options is presented here.

RSS generates non-collapsing space filling designs in the search box region $[l, u]$ (recall (1.1)), and it produces a corresponding global solution estimate. This information is passed along to LGO's other—global and local—solvers for refinement. RSS can be especially useful in solving hard GO problems under severe limitations regarding the number of model function evaluations.

The branch-and-bound search method (BB) is an implementation of a well-established optimization approach. The BB implementation in LGO combines feasible set partition steps with both deterministic and randomized sampling.

Pure random search (PRS) is a simple "folklore" approach to global optimization. The GARS option implements an adaptive stochastic search procedure which—while theoretically globally convergent—typically leads to a significant performance improvement over PRS and other passive search approaches.

The multi-start (MS) global search option applies a similar stochastic search strategy to GARS; however, the total sampling effort is distributed among several global scope searches. Each of these leads to a starting point for subsequent local search. In general, this search strategy requires the most computational effort (due to its multiple local search phases); however, in complicated models, it often finds the best numerical solution. For this reason, MS is the recommended default LGO solver option for global scope optimization.

Each of the global scope solver modules is expected to generate a "sufficiently close" approximation of the global solution point(s), before LGO is switched to local search. This "paradigm switch" is dictated by practical requirements, not by convergence theory: as such, it can lead to (much) faster optimization program termination. However, theoretical global convergence is guaranteed only by using proper global search *ad infinitum*. If we wish to find a rigorously guaranteed close approximation of the solution by LGO's global scope solvers, then LGO runtimes can be expected to grow at an exponential rate as a function of model size (characterized by the number of variables and model functions). For clarity, a similar comment applies to all other theoretically rigorous global search methods. Therefore the addition of a much quicker local search option is a practically important feature of LGO.

In a typical optimization run, the LGO user selects one of the global (BB, GARS, MS) solver options: the corresponding global search phase is then automatically followed by the local search (LS) option. It is also possible to use the LS option in stand-alone mode, started from an automatically set (default) or a user-supplied initial solution "guess." For algorithmic and practical reasons, an initial LS phase precedes all global search options. If RSS is invoked then it follows LS, then RSS is followed by a second LS phase before launching the global search mode chosen.

Ideally—and also in many practical cases—each of BB, GARS, and MS followed by LS will return identical answers, except perhaps small numerical differences. However, LGO users may wish to try using each of the global search options when solving complicated problems to see which one gives the best numerical results with given (or perhaps modified) option and parameter settings. For small to moderate size problems defined by analytically given (i.e. "easily computable") functions, the corresponding runtimes are seconds or at most minutes. Hence, in such cases, trying several solver modes and option settings is entirely feasible.

The global solver options all use an aggregated merit (exact penalty) function. As one of the possible approximations of the constraints g, the objective function $f(x)$ is augmented by terms of the form c_i max $[g_i(x), 0]^2$. In the corresponding (still continuous or Lipschitz-continuous) problem formulation,

$$\min f(x) + \sum\nolimits_i c_i \max[g_i(x), 0]^2 \text{ subject to } x \in D_0 := [l, u] \tag{1.2}$$

D_0 replaces the constraint set D of (1.1) in the global search phase. The aggregated merit function shown in (1.2) attempts to balance the aims of reducing the original objective function and of satisfying the constraints $g_i(x) \leq 0$ for $i = 1,\ldots,m$. The non-negative multipliers c_i associated with the constraints g_i can be either directly assigned or adaptively selected, to enforce the (approximate) numerical feasibility and optimality of the solution found. (Note that instead of using the l_2-norm as shown by (1.2), other l_p-norms can also be used for a suitably chosen $1 \leq p \leq \infty$.) The outlined aggregation of constraints and objective function is automatically supported by LGO.

As noted above, the local search phase is based on an implementation of the generalized reduced gradient algorithm. This search strategy is aimed at "locally precise search": therefore its usage is recommended also as the last search phase, following one of the global searches. The application of LS typically results in an improved solution, since it is aimed at attaining or maintaining feasibility and—simultaneously—at improving the objective function value. LGO's use as a local solver only can be recommended, when one or several good initial solution estimates are directly available, or whenever the LGO user knows that local search suffices (as, e.g., in the case of provably convex models).

Additional search strategies and solver options are also considered for future inclusion in the LGO solver suite as dictated by user demands.

Due to its derivative-free optimization strategies, LGO can be primarily recommended to solve continuous GO problems, in which the unavailability or tedious generation of model function derivatives excludes the possibility of applying more specifically tailored approaches. (It is worth pointing out that in most truly global scope search algorithms local derivative information plays little or no role.) Let us emphasize that LGO can handle optimization models with a "black box" structure: this category specifically includes many advanced models developed in ISTC systems as highlighted earlier.

1.2.2 LGO Program Structure

The core of LGO has been originally developed for use in conjunction with FORTRAN development environments. Specifically, LF77, LF90, and LF95 by Lahey Computer Systems [64, 65] have been mostly used for development (for the current release version of LF95 visit www.lahey.com). Subsequently, LGO has been ported to many other compiler platforms. LF95 directly supports links to other programming languages under Windows operating systems (OS) such as Borland C/C++ and Delphi, Microsoft Visual C/C++, and Visual Basic. Digital, Compaq, Intel Visual FORTRAN, G77, G95, GCC, GFORTRAN, and Salford FORTRAN (FTN95) are further examples of compiler platforms that have been tested to run LGO. LGO can also be used on Mac, Linux, and UNIX platforms, since its core code is fully portable. For instance, GCC and GFORTRAN or G95 are freely available for all hardware and OS platforms mentioned above.

LGO can be used as an executable program, a static object file, or as a dynamic link library (dll). There are secondary and somewhat compiler-dependent differences among the LGO delivery versions: such details are outside of the scope of the present discussion. The changes between delivery versions are straightforward, and the core LGO solver functionality is identical in all cases. For illustration, we will review the object file delivery-based LGO version that is used with a FORTRAN development environment. (In this case, all program files discussed below have a FOR filename extension; if a C compiler is used, then the . FOR filename extensions should be replaced by .C extensions).

To establish a standard application program structure, the following files are written by the model developer, using the templates and sample programs provided with LGO.

- LGOMAIN is the main (driver) program that defines or retrieves from an input file (to be discussed below) LGO's static calling parameters before activating LGO. Here the adjective static refers to model descriptors and LGO solver options defined only once during a given LGO application run. LGOMAIN can also include additional user actions such as links to other program files and to external applications, to report generation, and to the further optional use of LGO results.
- LGOFCT defines the model objective *f* and constraint functions *g* that describe the dynamic components of an optimization problem. Here dynamic means that LGOFCT will be called in every iteration of LGO, to evaluate all model functions for the algorithmically generated sequence of input variable arguments *x*. Again, this file may include calls to other application programs such as external modules and procedures, in order to evaluate the model functions.
- LGO.IN is an optionally used LGO input parameter (text) file that stores LGO's static calling parameters. If LGO.IN is used, then the basic role of LGOMAIN is to read the static information and then to call LGO.

The source code files LGOMAIN and LGOFCT are to be compiled and linked to the LGO object or dll file using a suitable FORTRAN or C compiler. At runtime, LGOMAIN calls LGO: the latter iteratively calls LGOFCT. LGO's operations can be partially controlled by the static input parameter file, or by parameters defined in LGOMAIN: this setup structure facilitates the repeated executions of LGO runs under various model specifications and/or solver parameterizations. Of course, LGOFCT can also be changed if necessary to test or to develop different model variants.

During the LGO run three types of result files can be (optionally and automatically) generated. The first one of these files, called LGO.SUM, presents a concise summary of the results obtained: the optimized decision variables and the function values at these arguments. This information is sufficient in "routine" LGO applications: this way the model developer can avoid browsing through a possibly significant amount of runtime details. The second (optional) report file, called LGO. OUT, provides more detailed information related to the complete optimization process. Specifically, it shows which solver options have been called, how they performed (showing their corresponding final results), and which stopping criterion has led to the termination of the LGO run. Therefore its analysis can be useful, e.g., during the development of a new application or to test various LGO input parameter settings. The third (optional) report file, called LGO.LOG, records the entire sequence of the input vector arguments (x) generated, together with the resulting function values $f(x)$ and $g(x)$. Using this option could lead to a huge amount of stored information: hence, it should be used carefully, mostly during model development or when tracking down some possible (model or solver) error. This option may also be useful in such cases when the number of model function evaluations is very limited due to resource limitations: hence, all generated information can be valuable for the model developer.

Both LGOMAIN and LGOFCT can be provided as compiled object or dll files, possibly written in other languages as noted above. These program modules can also be connected to additional program files or even to executable programs. In such cases, LGO can perform true "black box" optimization, assuming proper input/output communication among LGO, LGOMAIN, and LGOFCT.

To summarize the above discussion, the interdependence structure of the LGO program components is shown by Fig. 1.1.

[An optional input parameter file]

$$\begin{tikzcd}
& \textbf{LGO.IN} \arrow[d] & \\
\textbf{LGOMAIN} \arrow[r, leftrightarrow] & \textbf{LGO} \arrow[r, leftrightarrow] \arrow[d] & \textbf{LGOFCT} \\
\textbf{LGO.SUM} & \textbf{LGO.OUT} & \textbf{LGO.LOG}
\end{tikzcd}$$

[Optionally generated result files]

Fig. 1.1 LGO application program structure

1.2.3 Connectivity to Other Modeling Environments: Current Implementations

LGO can be made available as a solver engine option in various modeling environments, typically as a dll or object file. In the simplest case, this requires only a call to LGO, with proper communication between the modeling platform and the solver. As noted earlier, optimization modeling systems are equipped with model interpreter capabilities and with a range of other features to support transparent and scalable model development.

Due to certain platform-specific compatibility requirements the linked solver options are not necessarily identical regarding all features. Without going into unnecessary details, we list below the currently existing LGO solver implementations for modeling systems, listed in order of release.

- LGO solver suite for nonlinear optimization (compiler platform implementations)
- GAMS/LGO
- MathOptimizer Professional for *Mathematica*
- TOMLAB/LGO for MATLAB
- Global Optimization Toolbox for Maple
- MPL/LGO
- AIMMS/LGO
- AMPL/LGO
- Excel/LGO
- MATLAB/LGO

All listed solver products are directly available for personal computers running currently used MS Windows operating systems. Several of these are also implemented, and all can be made available upon request, across all major hardware and operating system platforms.

1.2.4 An Illustrative Example

To illustrate the model development and solution procedure using LGO, we will consider the solution of a transcendental system of equations $g(x) = 0$ $g{:}\boldsymbol{R}^n \rightarrow \boldsymbol{R}^n$ assuming that all variables take values from a given n-dimensional interval $[l, u] \subset \boldsymbol{R}^n$. As noted earlier, such equation systems may not have solutions at all, or they could have a single solution, or several solutions, or possibly even infinite sets of solutions. In lack of a good initial "solution guess," local scope search strategies (such as Newton's method and its various extensions) may fail to find the solution(s) in this important class of problems. In order to (hopefully) avoid the scenario of multiple solutions, we can attempt to find the solution that has minimal l_p-norm or—more generally—the solution closest to a given point in $\boldsymbol{R}^n$. (As discussed earlier, $1 \le p \le \infty$ is chosen by the model developer.) If the l_p-norm criterion still does not guarantee the uniqueness of the solution vector, then other modeling constraints can be imposed. We will assume that such added constraints are not needed, and—as an example—we look for the minimal l_2-norm solution of the model type shown below.

$$\min||x||^2 \quad g_i(x) = 0 \quad j = 1, \ldots, n \quad x \in D_0 := [l, u] \tag{1.3}$$

Obviously, (1.3) is a specific case of (1.1), with a convex objective function and with possibly complicated non-convex constraints. The model instance used for illustration is described below, for $n = 3$, $x = (x_1, x_2, x_3)$; the symbols sin, cos, ln, and exp below denote the sine, cosine, natural logarithm, and exponential functions:

$$\begin{aligned} f(x) &= \|x\|^2 = {x_1}^2 + {x_2}^2 + {x_3}^2 \\ g_1(x) &= \sin\left(x_1 + {x_2}^2 + x_3\right)^2 - \sin(x_1 x_2 - 3x_3) \\ g_2(x) &= \ln\left(1 + (x_1 - 2x_2 + x_3)^2\right) + \cos(x_1 x_2 - x_3) - 1 \\ g_3(x) &= \sin(\exp(x_1 - x_3)) + \sin(x_1 x_2 - x_3) - \cos(x_2 - x_3) \\ &\quad + 1 - \sin(1) \\ &\quad - 5 \le x_j \le 8 \quad \text{for} \quad j = 1, 2, 3 \end{aligned} \tag{1.4}$$

We created (concocted) this little example for the present discussion: avid Readers will notice that $x^* = (0,0,0)$, $f(x^*) = 0$ is the unique global solution. Of course, in general we do not know in advance if a system of equations like (1.4) has a solution at all: thus a global scope search over [l, u] is justified. We believe that such situations are not infrequent in the real world of research. Figure 1.2 shows that—without specific insight—the system of equations in (1.4) is far from trivial even if we knew that $x_3 = 0$. (The figure has been generated using *Mathematica*).

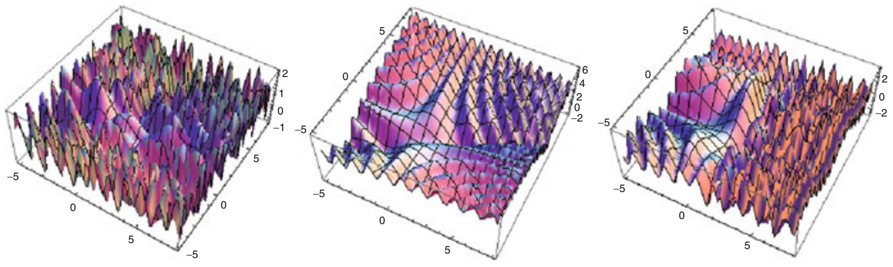


Fig. 1.2 Surface plots of the constraint functions in (1.4) assuming that $x_3 = 0$

In solving the model variants discussed above we used *MathOptimizer Professional*: this is the LGO solver suite linked to the *Mathematica* model development environment.

Here we do not want to go into details regarding the important issue of model scaling, but obviously the model functions in (1.4) could all be multiplied by individual scaling factors, thereby making them more or less influential as a model component. (Specifically, if $f(x) = \|x\|^2$ dominates the other functions, then in this illustrative example the objective of minimizing $f(x)$ would easily "drive home" the solver.) In order to make the model a bit more difficult numerically, f is multiplied by 0.2. Notice that if its multiplier is 0, then we have a dummy objective function and have to solve directly the system of equations $g(x) = 0$ in the given box [l, u]. The resulting system has multiple solutions: as an example, here is a numerical solution found by LGO's local search mode started from the "guess" (7,7,-4) when we set $f = 0$:

x_1* ~ (1.363934386, 1.5414564089, 1.8782048276)
Maximal constraint violation ~ $6.83127.10^{-7}$
Number of function evaluations = 366

Let us remark that the number of function evaluations reported includes also the necessary finite difference-based derivative estimates utilized by the LS mode in LGO.

Next, for illustration, we tried to solve the system of equations locally using the definition $f(x) = 0.2\ \|x\|^2$ for the objective function. Perhaps not too surprisingly, this attempt has led to an infeasible local solution x_2* as shown below:

x_2* ~ (0.6574644425, 1.1785306474, 1.3186902472)
0.2 $f(x_2^*)$ ~ 0.7120275896
Maximal constraint violation ~ 0.8555029107
Number of function evaluations = 472

Finally, we solved the same model version globally, using LGO's MSLS operational mode. The resulting numerical solution is displayed below.

x_3* ~ ($6.1 \cdot 10^{-9}$, 0, $2.3 \cdot 10^{-9}$)
0.2 $f(x_3^*)$ ~ $8.5 \cdot 10^{-18}$
Maximal constraint violation ~ $6.9 \cdot 10^{-9}$
Number of function evaluations = 12039

The LGO execution time in the last (longest) run is 0.03 s on a laptop computer running under Windows XP with Service Pack 3, 3 GB of RAM, and an Intel Core 2 Duo CPU P8400 @ 2.26 GHz.

A practically useful side benefit of using *MathOptimizer Professional* is that it automatically generates the LGOFCT and LGO.IN files, in addition to the result files LGO.OUT and LGO.SUM. The generated program files can be utilized also directly in conjunction with LGO (without using *Mathematica*).

To conclude this section, let us remark that the current standard LGO delivery versions can handle models with a few thousand variables and general constraints (in addition to the bound constraints). The dimension settings can be adapted to user requirements as needed, within the limitations of the hardware / OS / compiler platform(s) used. For further details regarding LGO and its implementations we refer again to the cited works, as well as to a growing range of practical applications handled by these implementations.

1.3 Modeling and Optimization in Space Engineering: A Review of Applications and Perspectives

In order to offer some insight into the vast area of modeling and optimization needs arising in space engineering, a fairly extensive set of case studies is reviewed in this section broadly categorized into subsections. Recalling the discussion of Sect. 1.1, this is where nonlinear programming (NLP), mixed-integer linear or nonlinear programming (MILP and MINLP), global optimization (GO), multi-objective optimization (MO), and multidisciplinary design optimization (MDO) are considered as essential tools to provide robust and efficient solutions to difficult optimization problems. With respect to these concepts, in addition to the textbooks mentioned earlier, cf. also the *Mathematical Programming Glossary* [66]. Further references on the closely related subject of optimal control theory are [67–73, 75–80].

In the following subsections, we will review a number of optimization applications contributed by our coauthors of the present edited volume [11]. To properly extend the scope of our discussion, we will also review a number of further topical studies from space engineering. The topics covered can be broadly categorized into the following areas:

- Mission analysis and trajectory planning
- Planning and scheduling
- Cargo loading and unloading
- Payload accommodation
- System design
- Subsystem design
- Ergonomic aspects
- Payload performance
- Observation data handling and remote monitoring
- Cost and revenue management

Let us point out that close connections are evident among these subject categories. Hence, studies which consider several of the listed subjects can be entirely justifiable—especially when optimized system (subsystem) performance is a focal point of the analysis.

1.3.1 Mission Analysis and Trajectory Planning

Since the very beginning of space engineering, great attention has been paid to trajectory planning, in order to attain the expected key mission objectives (reaching a specific target) within the technological and economic limits of the systems available. The optimization-based approach was indispensable to make the mission feasible, even regardless of performance and cost concerns. This perspective remains valid when coping with the significant space exploration challenges arising today or foreseen for the near future. More recently, trajectory optimization has also been aimed at reducing overall mission expenses while also improving mission effectiveness, in terms of experimental capability and/or cargo capacity.

In current launch systems, the propellant mass constitutes a large fraction of the total weight at launch. Therefore a significant effort is devoted to minimizing the amount of propellant, since such optimization enhances the launch vehicle capabilities in terms of transported payload or achievable destinations in space.

Other important optimization objectives are also studied. For instance, regarding future manned interplanetary missions (such as a manned outpost on Mars), the outward/inward journey duration has to be made minimal, in compliance with stringent safety constraints.

These and similar problems are considered by optimal control theory: in fact, trajectory (or, more generally, mission analysis) optimization represents a long-standing prominent application. The accumulated professional literature is based on significant expertise: nonetheless, many tough challenges are still open to research, attracting the commitment of scientists and experienced practitioners.

Optimal control theory benefits from two main approaches based either on direct or indirect methods. The indirect methods discussed in this book by Colasurdo and Casalino [84] have been successfully utilized by these authors in the second Global Trajectory Optimization Competition [81].

The present volume includes several contributions to trajectory optimization: these are focused on complementary perspectives, in terms of the methodologies adopted and specific applications. In trajectory optimization problems, optimal control, NLP, GO, MILP, and MINLP play an essential role as support methodologies.

Becerra [82] provides an introduction to direct collocation methods for computational optimal control. In a direct collocation method, the system state is approximated using a set of basic functions, and the dynamics are collocated at a given set of points along the time period considered, resulting in a sparse NLP problem. The chapter discusses local direct collocation methods, which are based on low-order basis functions employed to discretize the state variables over a given time period.

Cassioli et al. [83] deal with the application of global optimization techniques to the design of interplanetary trajectories. This is an extremely hard problem, mainly because of the very large number of local minimizers present in real problems. Despite these challenges, it is possible to design low-cost high-energy trajectories with little or no human supervision. The chapter analyzes those modeling techniques that computational experiments have shown to be most successful, along with some of the algorithms that might be used to solve such problems.

Colasurdo and Casalino [84] emphasize the use of indirect methods for the optimization of spacecraft trajectories. They present a general framework in order to apply the theory of optimal control to spacecraft trajectory design. This procedure allows for an almost automatic derivation of the boundary conditions which must be satisfied by an optimal trajectory, depending on the specific constraints of the problem studied. The general method of stating the optimal control problem makes this indirect approach suitable to manage many specific features of the space missions, such as impulsive and/or low-thrust engines, planetary flybys, (high-altitude) atmospheric flights, and so on.

Topputo and Belbruno [85] investigate the optimization of low-energy transfers. The key concepts of trajectory optimization are reviewed, and the direct transfer strategy is briefly outlined. Next, the restricted three- and four-body problems are described, and their properties are discussed. Two different types of low-energy transfers are then optimized: impulsive Earth–Moon exterior low-energy transfers and low-energy, low-thrust transfers to the L1 periodic orbits of the Earth–Moon system.

Di Lizia et al. [86] propose the use of differential-algebraic techniques to globally optimize multi-gravity-assist interplanetary trajectories with deep-space maneuvers. A search-space pruning procedure is adopted, and the overall trajectory design is decomposed into a sequence of subproblems. First the entire parameter search space is partitioned into boxes (subintervals), and then the objective function and the constraints are represented by Taylor series approximations over these boxes, in terms of the design variables. Due to the polynomial representation of the model functions, a relatively coarse grid can be used and efficient design space pruning can be performed.

Cremaschi [87] reviews the problem of trajectory optimization of launch and reentry vehicles: such vehicles have to travel part of their trajectory within the Earth's atmosphere. This very complex problem is approximated by using simplified equations from physics. In a second step, the corresponding mathematical model is represented by a set of differential equations that may be then discretized and solved by nonlinear programming. Both shooting and collocation methods are considered. (Let us remark here that the proper application of the shooting method may require GO methodology as shown by [88].)

Büskens and Wassel [89] describe the European Space Agency's (ESA) NLP solver WORHP. WORHP is aimed at solving possibly very large-scale, sparse nonlinear optimization problems with millions of variables and constraints. WORHP's core solver engine is based on a sparse sequential quadratic programming (SQP) and an interior point (IP) method implementation. In addition, it includes sparse linear algebra tools, efficient routines for computing sparse derivatives, and update techniques for Hessian approximations.

1.3.2 Planning and Scheduling

Space missions are undoubtedly among the most complex human undertakings: therefore they require high-accuracy planning in order to minimize risks and costs both before and during the mission. This general statement is valid for many kinds of topical applications that range from deep-space exploration, satellite constellations planning, in-orbit infrastructures to future manned exploration programs. A notable example is the International Space Station (ISS) by the National Aeronautics and Space Administration (NASA, http://www.nasa.gov).

Significant contributions to this area of research are offered by a number of topical studies using operations research tools. For example, Gabrel and Murat [90] developed a mathematical programming-based approach to plan the missions of Earth observation satellites: the objective is to optimize the shots taken during a given time period, in order to satisfy specific imaging requirements. Gürtuna and Trépanier [91] applied an Earth-based vehicle-routing optimization model to promote a concept of on-orbit satellite service, such as refueling, maintenance, and failure repairing. Lang [92] discusses an optimization-based approach to analyze the launch capacity of commercial satellites, taking into account the aspects of vehicle selection, launch operation resource management, and contingency decision making.

The ISS itself poses a significant number of planning and scheduling optimization problems. An important example deals with its permanent logistic support and on-orbit resource resupply. An international fleet of vehicles provides the station with the necessary upload and download activities. A traffic model based on an MILP formulation (in terms of a lot-sizing problem), aimed at optimizing ISS logistic planning, is discussed in this book by Fasano [94]. The suggested approach has been utilized to look into the introduction of the automated transfer vehicle (ATV) by the ESA (http://www.esa.int) in the already existing fleet of support vehicles, offering significant insights into its sizing during the design phase, and its potential utilization.

The extreme complexity of future manned or unmanned interplanetary missions—involving lunar bases and in-orbit infrastructures, planned on a medium- or long-term multiphases paradigm—is expected to give rise to difficult traffic modeling issues that will necessarily require a systematic optimization approach.

The experimental activity on board of the ISS requires a detailed (again, short and medium term) scheduling optimization, in order to benefit from the available resources such as power, communication channels, data handling, payloads (experimental instruments) as much as possible. In addition to these, the capability to efficiently face off-nominal scenarios (such as failures or equipment malfunctioning) by quick rescheduling is crucial to minimize the possible damage. Advanced optimization methods are then expected to contribute to ad hoc decision support and troubleshooting systems.

In manned on-orbit laboratories, the crew (work) time itself represents a very critical resource. As an example of related experience, the MIR station program by the Russian Federal Space Agency (Roscosmos, http://www.roscosmos.ru) dealt with the optimized scheduling of biomedical activity performed on board by the astronauts to execute the human posture experiment: for a topical discussion, consult Fasano and Piras [95]. The experiment was based on the identification (via infrared flashes) of dedicated markers positioned on the astronaut's body while performing well-defined physical exercises. This allowed the analysis of the posture and its variation during the stay in space. The critical aspects related to the flight procedure were connected to the crew activity needed for the operations of fixing the markers (up to four different types) on the body. Additionally, the marker positioning should be carried out as precisely as possible in order to minimize the measurement errors. Sequences of experiments were executed at different times during the performance on orbit. The related optimization problem consisted in finding the sequence of experiments that minimized the setting-up time to pass from one to the next while guaranteeing the execution of all experiments. A traveling salesman problem (TSP)-based approach based on an MILP model formulation was followed: here each marker layout represented a single *"city,"* taking into account a number of operational constraints. Regarding the TSP and approaches to its solution, consult, e.g., Padberg and Rinaldi [96], Williams [97], or the *Mathematical Programming Glossary* [66].

1.3.3 Cargo Loading and Unloading

While the human experts' time on board of a spaceship is an extremely valuable resource, in many cases the space available is just as important: hence, the solution of cargo accommodation (i.e., loading or packing) problems also becomes essential. Frequently, conditions that hold "automatically" on the ground will not do so in space: therefore stringent conditions are posed, e.g., on load balancing, stability, load bearing strength, and multi-drop operations. In addition to these complications, the loading space and object geometries involved are often fairly complex.

The literature of optimized multidimensional loading and packing problems is extensive, and advanced methods are available to solve difficult instances efficiently. Consult, e.g., Castillo et al. [8], Cagan et al. [98], Dyckhoff et al. [99], Fekete and Schepers [100], Faroe et al. [101], Martello et al. [102, 103], and Pisinger and Sigurd [104, 105]. Most of the related research focuses on the orthogonal placement of rectangles (parallelepipeds) into rectangles (parallelepipeds) or on placing circles (spheres) into circles or rectangles (parallelepipeds). In addition, remarkable studies concerning specific nonstandard packing problems are available: consult, e.g., Dowsland et al. [106], Egeblad et al. [107], Egeblad and Pisinger [108], Fischetti and Luzzi [109], Gomes and Oliveira [110], Ibaraki et al. [111], Kallrath [112], Scheithauer et al. [113], Stoyan et al. [114], and Teng et al. [115]. Balancing conditions are also considered, e.g., by Egeblad et al. [116], Fasano [93], Fasano and Pintér [9], Stoyan and Romanova [119] and Takadama et al. [117].

Junqueira et al. [118] investigate the three-dimensional container loading optimization problem in the presence of stability, load bearing strength, and multi-drop constraints. An MILP-based approach is applied, and computational results are reported, comparing the performance of the models adopted on instances from the topical literature.

Stoyan and Romanova [119] investigate an NLP-based approach to study complex nonstandard two- and three-dimensional (2D and 3D) packing problems. A number of geometric items, called *phi*-objects, are introduced as mathematical representations of real-world objects. The basic concept of *phi*-functions is developed as an analytical tool to describe placement constraints such as containment, nonoverlapping objects, allowable distances, prohibited areas, and object translations and rotations.

The CAST (Cargo Accommodation Support Tool) project has been aimed at cargo optimization of a spacecraft [120, 121]. The project was funded by ESA and developed by Thales Alenia Space to support all ATV missions. The cargo accommodation inside the ATV cargo carrier implies the loading of both fluids (into storage tanks) and non-fluid items (dry cargo). The dry cargo accommodation involves different levels: small items have to be placed into bags, bags and large items are placed into sectors situated inside the racks or on the front panels of the racks, and racks have to be placed into predefined locations inside the spacecraft module. Mass and volume limitations (at bag, rack, and overall carrier level) are set, together with specific positioning rules. Balancing conditions are also stated to guarantee the attitude control of the whole system.

CAST—differently from most off-the-shelf packing optimization tools—has to deal with nonstandard packing issues. This specifically involves the consideration of *tetris*-like items, curved domains, and additional constraints such as those of balancing. A *tetris*-like item is a cluster of linked, mutually orthogonal parallelepipeds: this is more suitable to approximate large real-world items than single parallelepipeds [93]. The solution approach is based on a multilevel MILP-based heuristic procedure [120–123]. This procedure breaks down the original problem into a set of subproblems, corresponding to the different accommodation levels (items, bags, racks). An ad hoc extension of CAST (Fasano and Pintér [9], [11]) is already in use. It has been tailored to the stowage problem of the Columbus laboratory (ESA) attached to the ISS. Here we could take advantage of the significant similarities to the ATV cargo accommodation case studied in the framework of CAST [139, 140]. The specific problem of exploiting empty spaces of partially loaded containers by adding virtual items originated in this context [124].

While cargo optimization problems in space engineering are widely believed to be extremely difficult, issues concerning the on-orbit unloading of vehicles and modules (e.g., when docked at the ISS) are not any easier. An accurate operational scheme—that keeps the spacecraft permanently balanced during the unloading phase (while the internal mass distribution changes)—is mandatory. This key requirement serves to guarantee that the ATV is able to leave the ISS at any time (for possible emergency reasons).

1.3.4 Payload Accommodation

The payload accommodation problem (considered inside a space module) merges scheduling and packing considerations. A related study has been carried out for the Columbus laboratory (Fasano [149]). Formally, payloads consist of a set of facilities with specific resource requirements: such payloads had to be accommodated by predefined locations inside the racks of the internal module. The available on-board accommodation resources had to be used as efficiently as possible in order to optimize the experimental activity performed by the payloads. An overall MILP model was developed to meet two main objectives. The first of these was the identification of operational bottlenecks, in order to suggest different and more suitable on-board resource distributions. The second was the finding of satisfactory *dynamic* (time-dependent) solutions, i.e., feasible accommodations, in compliance with payload operational constraints and with the overall availability of resources. The latter aspects included crew time, electrical power, and heat rejection (i.e.,water and/or air cooling) constraints.

1.3.5 System Design

The increasing complexity of space mission design and assessment defines a challenging context of concurrent engineering scenarios that give rise to significant multidisciplinary and multi-objective optimization (MDO and MOO) applications: consult, e.g., Bandecchi et al. [125], Ciriani and Sarlo [126] and Cramer et al. [138]. The space systems considered consist of several subsystems: these are generally associated to specific disciplines such as avionic, control, electrical, structural, and thermal engineering. The subsystem-specific optimization objectives may be in conflict when considered jointly: therefore a high-level perspective is required to elicit the necessary trade-offs, in order to find an overall (globally) satisfactory solution. As the design complexity associated to each subsystem per se can be very high, the resulting MDO and MOO problems are frequently "black box" global optimization challenges.

The literature of "black box" GO is extensive: consult the related discussion and references in Sects. 1.1 and 1.2. In the context of the present topic, practically significant contributions are based on meta-heuristics and experimental design-based approaches combined with optimization: consult, e.g., Rios and Sahinidis [36], Cassioli and Schoen [127], and Pintér and Horváth [63], with many further references.

A methodological study aimed at identifying an efficient approach to tackle conflicts at different subsystems levels (as arising in space engineering during the design process) was conducted by Amata et al. [128]). Their research focused on a typical scenario that space system engineering teams have to deal with. The introduced MDO methodology can be brought to a wide range of space engineering applications.

The Water vapour and temperature in Troposphere and Stratosphere (WATS) project of ESA had been chosen for a detailed system design case study. The component (mission analysis, power, and propulsion) subsystems were selected as reference disciplines to simulate a realistic—even if somewhat simplified—space engineering environment. The suggested MDO approach benefited from the simultaneous application of optimization tools from neighborhood search, game theory, and multi-criteria decision analysis. The heuristic neighborhood search approach was aimed at finding a set of *Paretian* (i.e., nondominated) solutions at the highest (system) level. As the set of such solutions could be extremely high (even infinite), game theory and multi-criteria aspects were also introduced to explore a tractable set of solutions, *satisfactory* from the overall point of view.

Pastrone and Casalino [129] present an integrated design *and* trajectory optimization approach for hybrid rocket motors and describe a dedicated MDO-based code. They use a few key parameters to define the design of the hybrid engine; next, the trajectory optimization aspect is handled by continuous controls. To handle the different aspects in a unified framework, a mixed optimization procedure has been developed. To maximize overall mission performance, the direct optimization of engine design variables is coupled with the resulting trajectory's indirect optimization. The proposed optimization procedure is illustrated by interesting applications.

1.3.6 Subsystem Design

Within the context of the discussion in Sect. 1.3.5, demanding optimization problems are often associated even with the design of (structural, thermal, avionic, power, software and navigation control) sub-systems. Since the mathematical models involved are typically complex, a simulation approach is often preferred to the more difficult ones based on optimization methodologies. This view is referred to as model-based design (MBD); consult, e.g., Nicolescu and Mosterman [130]. An example of the MBD approach is the determination of the structure and shape of reentry vehicles.

An interesting application is provided also by Boada et al. [150]. They propose an advanced approach to control a formation flying mission acting both on attitude and relative position. Optimization-based formulations are also frequently used, however. An interesting example is discussed by Norstrøm et al. [131] regarding the optimal design of control software systems. Such systems are used, e.g., in satellites to activate certain system functions, such as antenna deployment. The control software receives information from sensors and uses this information to trigger the required functions. Let us point out that both inadvertent activation and delayed response can have severe consequences. Hence, the way sensor information is processed can strongly influence the overall performance. The presented approach models various design options in detail so that the software control flow can be optimized applying decision theory [132].

Norstrøm et al. consider the decision of when to deploy a satellite antenna. More precisely, the control software has to decide when to inspect the sensors and when to deploy the antenna. The authors show how to optimize both the inspection time and the time to deploy the antenna, given the inspection results. The consequences of the control software-based decisions are quantified in monetary loss associated with failure: this approach allows the evaluation of the risk in terms of the expected monetary loss.

1.3.7 Ergonomic Aspects

In manned space missions, the many related biological and biomedical issues also give rise to tough optimization problems. As an example, microbial contamination control inside a sealed module represents a challenging issue in manned interplanetary missions foreseen in the near future.

In this context, an analytical study was carried out by Thales Alenia Space in the framework of the MIR mission [133]. The mathematical modeling paradigm was based on the Lotka–Volterra prey–predator model scheme: consult, e.g., Brauer and Castillo-Chavez [134]. The microbial colonies were modeled as "prey" and the possible environmental control actions such as air filtering and conditioning (in terms of temperature, humidity, and velocity) represented the activities of the "predators." The model also included contamination control efficiency targets: the resulting problem was solved using optimal control tools.

1.3.8 Payload Performance

A primary aspect concerning payload performance is obviously related to optimal on-board resource scheduling. As an example, the objective can be to maximize the number of experimental cycles executed during a given mission (or within a given time period). The optimization of payload design itself will also be beneficial in order to maximize mission efficiency.

A very interesting problem that belongs to this category is Laue's lens, a device designed for gamma-ray astrophysical observations: consult, e.g., Halloin and Bastie [74]. The device consists of a set of different types of crystals, all placed on a platform. The instrument efficiency strongly depends on the following aspects: the partition of the crystal set into subsets of different typologies; the dimension, position, and orientation of each crystal; the layout domain geometrical configuration (i.e., overall shape of the platform); and the areas where the crystals can be placed. The related optimization problem will become a difficult MINLP that (evidently) can be similar in difficulty to several realistic cargo loading problems.

In certain cases, payload performance optimization may be achieved by influencing overall system behavior. This is the case for observation satellites where an efficient pointing error minimization (obtained by a very accurate attitude control system) is mandatory.

Mehlem [135] discusses the problem of optimal magnetic cleanliness. Spacecraft quite often generates undesired magnetic fields that could disturb scientific experiments, specifically including particle-based studies. The chapter discusses a related model development framework and an NLP-based solution approach.

1.3.9 Observation Data Handling and Remote Monitoring

Space observation and telemetry lead to difficult optimization problems, focused on best possible measurement accuracy and precision under technological and logistic constraints. Similarly challenging issues arise in the areas of data handling and interpretation. A well-known example is the paradigm of Global Positioning Systems (GPS): here measured data have to be analytically processed in real time.

The position determination problem (on Earth) by using satellite distance measurements can be cast as a distance geometry exercise: the objective is to find the intersection of spheres. This problem can be formulated and handled by GO: consult, e.g., Migdalas et al. [136].

Future interplanetary exploration is expected to give rise to hard remote monitoring issues. Examples are semiautonomous rover control or the observation of planetary regions. Addressing the latter issue, Fasano and Pintér [9] investigate the optimized configuration of sensor cameras, to be placed in a suitably defined three-dimensional cubic region E, in order to maximize the coverage of an embedded cube $C \subset E$. The sensing regions associated with each of the cameras are convex but not necessarily identical. In order to handle this practically important problem-type, mixed-integer linear and nonlinear programming (MILP and MINLP) model formulations are presented, with illustrative numerical results.

1.3.10 Cost and Revenue Management

As space exploration ambitions are ever increasing, the space engineering community is (has to be) committed to take into account the economic aspects related to prospective projects and programs. Financial constraints have become of paramount concern in recent times. This leads to the determination of key strategic and tactical choices regarding the feasibility and prioritization of space projects. As the space engineering sector has to follow an extremely competitive paradigm, the capability of optimizing research and development (R&D) projects as well as equipment utilization plans—in terms of cost minimization and revenue maximization—is a categorical imperative nowadays.

From a commercial point of view, two main classes of space systems can be considered. Systems in the first category perform a continuing service over a certain time period: examples are telecommunication satellites. Systems in the second category perform recurrent support: examples are launchers. The relevant cost and revenue plans can be formulated either in terms of nonlinear (noncontinuous) control problems or—after a discretization of the time period considered—as MILP and MINLP models.

Thales Alenia Space carried out a survey on this important subject, taking as reference the Space Solar Power Scenario Study (conducted in 1998 by the German Aerospace Center). The latter study focused on the realization of an orbiting solar power station. A specific cash-flow mathematical model was implemented by Fasano adopting an MILP approach. In the study and similar analyses, various optimization objectives could be considered, depending on the perspective requested: examples are minimal break-even-point distance, minimal utilization costs, and maximal final revenue obtained.

1.3.11 Further Application Perspectives

To conclude the discussion of Sect. 1.3, we mention some other—quite different, but at the same time highly relevant—application areas in which OR-based modeling and optimization can play a significant role. These applications are increasingly significant, even if (currently) they are from a relatively small portion of novel application areas. Further miscellaneous applications are also listed to indicate a wide range of perspectives. Most of the applications highlighted here can also contribute to technological spin-offs in several non-space sectors.

Challenging optimization issues arise in the context of guidance and navigation of vehicles: here typically well-tested classic methods such as Kalman filtering are usually adopted: consult, e.g., Welch and Bishop [137]. The increasingly used MBD poses ever more—at least in terms of its subproblems—nonlinear optimization frameworks.

As a second example, we mention the need for a nontrivial propulsion plan during a launch mission. Given the overall available capacity of the boosters, the objective is to provide an optimized fuel supply sequence in order to keep the overall system continuously balanced, thereby making its control easier and more efficient.

Risk analysis—including worst case, what-if, and sensitivity analyses—often can be efficiently combined with advanced optimization tools. This is the case, for instance, in many practical cargo accommodation tasks, since the actual mass of the items to load may not be known exactly up to the last minute.

It is understood that the future manned space missions such as the ones concerning lunar base habitation will require the resolution of many challenging issues of space urbanism and architecture, agriculture, and life science. Issues to consider will include facility location, infrastructure interconnections, ergonomic and livableness conditions in the forthcoming artificial environments. These issues will necessarily lead to very tough optimization problems, to be addressed at different levels.

In spite of the tremendous challenges faced now and in the future, the ambitious long-term objectives of space engineering will go well beyond the goal of "landing a man on the moon and returning him safely to the earth" as conjectured by President J.F. Kennedy [143].

1.4 Concluding Remarks

In this chapter we have reviewed an OR-based approach to the modeling and solution of many important problems arising in the context of space engineering. In Sect. 1.1, the introduction of a general model development framework is followed by highlighting several relevant model types and paradigms such as NLP and GO, MILP and MINLP, MOO and MDO. We also discuss corresponding optimization software implementations, and—as an illustration—in Sect. 1.2 we introduce the LGO solver suite for general constrained nonlinear (global and local) optimization. LGO integrates several algorithmic approaches with strong theoretical convergence properties, including a new option that enables the handling of computationally expensive models under resource constraints. We also discuss a range of model development environments and tools (Sects. 1.1 and 1.2) that can be put to good use also in space engineering applications. A fairly detailed review of current and prospective OR applications in space research is provided in Sect. 1.3: this review draws upon contributed chapters to the present volume, as well as on other important sources. We will be glad to lean about new application areas and challenges.

Acknowledgments First of all, we wish to thank all contributing authors to the present edited volume [11], as well as our coauthors for their contributions to joint work listed in the References. Thanks are also due to M. Parisch and C. Tomatis (Thales Alenia Space) for useful discussions and valuable suggestions regarding the topics reviewed in Sect. 1.3.

References

1. Schittkowski, K.: Numerical Data Fitting in Dynamical Systems. Kluwer, Dordrecht (2002)
2. Pintér, J.D.: Global Optimization in Action. Kluwer, Dordrecht (1996)
3. Pintér, J.D.: Globally optimized calibration of nonlinear models: techniques, software, and applications. Optim. Method. Softw. **18**, 335–355 (2003)
4. Pintér, J.D.: Calibrating artificial neural networks by global optimization. Expert Syst. Appl. **39**, 25–32 (2012)
5. Pintér, J.D.: Globally optimized spherical point arrangements: model variants and illustrative results. Ann. Oper. Res. **104**, 213–230 (2001)
6. Pintér, J.D., Kampas, F.J.: Nonlinear optimization in *Mathematica* with *MathOptimizer Professional*. Mathematica Educ. Res. **10**, 1–18 (2005)
7. Pintér, J.D., Kampas, F.J.: Benchmarking nonlinear optimization software in technical computing environments: global optimization in *Mathematica* with *MathOptimizer Professional*. TOP (Off. J. Spanish Soc. Stat. Oper. Res.). Published online August 11 (2011). Doi: 10.1007/s11750-011-0209-5
8. Castillo, I., Kampas, F.J., Pintér, J.D.: Solving circle packing problems by global optimization: numerical results and industrial applications. Eur. J. Oper. Res. **191**, 786–802 (2008)
9. Fasano, G., Pintér, J.D.: Global optimization approaches to sensor placement: model versions and illustrative results. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013b)
10. Ciriani, T.A., Fasano, G., Gliozzi, S., Tadei, R. (eds.): Operations Research in Space and Air. Kluwer, Dordrecht (2003)
11. Fasano, G., Pintér, J.D. (eds.): Modeling and Optimization in Space Engineering. Springer, New York (2013a)
12. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms. Wiley, New York (1993)
13. Peressini, A.L., Sullivan, F.E., Uhl, J.J.: The Mathematics of Nonlinear Programming. Springer, New York (1988)
14. Chong, E.K.P., Zak, S.H.: An Introduction to Optimization, 2nd edn. Wiley, New York (2001)

15. Edgar, T.F., Himmelblau, D.M., Lasdon, L.S.: Optimization of Chemical Processes, 2nd edn. McGraw-Hill, New York (2001)
16. Diwekar, U.: Introduction to Applied Optimization. Kluwer, Dordrecht (2003)
17. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
18. Hillier, F.J., Lieberman, G.J.: Introduction to Operations Research, 8th edn. McGraw-Hill, New York (2005)
19. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, New York (2006)
20. Horst, R., Pardalos, P.M. (eds.): Handbook of Global Optimization, vol. 1. Kluwer, Dordrecht (1995)
21. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Berlin (1996)
22. Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht (1996)
23. Mockus, J., Eddy, W., Mockus, A., Mockus, L., Reklaitis, G.: Bayesian Heuristic Approach to Discrete and Global Optimization. Kluwer, Dordrecht (1996)
24. Floudas, C.A., Pardalos, P.M., Adjiman, C., Esposito, W.R., Gümüş, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A., Schweiger, C.A.: Handbook of Test Problems in Local and Global Optimization. Kluwer, Dordrecht (1999)
25. Strongin, R.G., Sergeyev, Y.D.: Global Optimization with Non-Convex Constraints. Kluwer, Dordrecht (2000)
26. Pardalos, P.M., Romeijn, H.E. (eds.): Handbook of Global Optimization, vol. 2. Kluwer, Dordrecht (2002)
27. Zabinsky, Z.B.: Stochastic Adaptive Search for Global Optimization. Kluwer, Dordrecht (2003)
28. Liberti, L., Maculan, N. (eds.): Global Optimization: From Theory to Implementation. Springer, New York (2005)
29. Grossmann, I.E. (ed.): Global Optimization in Engineering Design. Kluwer, Dordrecht (1996)
30. Papalambros, P.Y., Wilde, D.J.: Principles of Optimal Design, 2nd edn. Cambridge University Press, Cambridge (2000)
31. Pintér, J.D. (ed.): Global Optimization: Scientific and Engineering Case Studies. Springer, New York (2006)
32. Neumaier, A.: Complete search in continuous global optimization and constraint satisfaction. In: Iserles, A. (ed.) Acta Numerica 2004, pp. 271–369. Cambridge University Press, Cambridge (2004)
33. Floudas, C.A., Gounaris, C.E.: A review of recent advances in global optimization. J. Global Optim. **45**, 3–38 (2009)
34. Pintér, J.D.: Global optimization: software, test problems, and applications. In: Pardalos, P.M., Romeijn, H.E. (eds.) Handbook of Global Optimization, vol. 2, pp. 515–569. Kluwer, Dordrecht (2002)
35. Pintér, J.D.: Nonlinear optimization in modeling environments: software implementations for compilers, spreadsheets, modeling languages, and integrated computing systems. In: Jeyakumar, V., Rubinov, A.M. (eds.) Continuous Optimization: Current Trends and Modern Applications, pp. 147–173. Springer, New York (2005)
36. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. Technical report, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh (to appear)
37. Floudas, C.A.: Deterministic Global Optimization: Theory, Methods, and Applications. Kluwer, Dordrecht (2000)
38. Nowak, I.: Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming. Springer, New York (2005)
39. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming. Kluwer, Dordrecht (2002)
40. Li, D., Sun, X.: Nonlinear Integer Programming. Springer, New York (2006)

41. Burer, S., Letchford, A.N.: Non-convex mixed-integer nonlinear programming: a survey. Technical Report, Department of Management Sciences, University of Iowa. http://www.optimization-online.org/DB_HTML/2012/02/3378.html (2012). Accessed on April 30, 2012
42. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, New York (1996)
43. Osman, I.H., Kelly, J.P. (eds.): Meta-Heuristics: Theory and Applications. Kluwer, Dordrecht (1996)
44. Glover, F., Laguna, M.: Tabu Search. Kluwer, Dordrecht (1997)
45. Rudolph, G.: Convergence Properties of Evolutionary Algorithms. Verlag Dr. Kovac, Hamburg (1997)
46. Voss, S., Martello, S., Osman, I.H., Roucairol, C. (eds.): Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Dordrecht (1999)
47. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms. Physica-Verlag, Heidelberg (2002)
48. Jones, N.C., Pevzner, P.A.: An Introduction to Bioinformatics Algorithms. MIT, Cambridge (2004)
49. Michalewicz, Z., Vogel, D.B.: How to Solve It: Modern Heuristics, 2nd edn. Springer, New York (2004)
50. Weise, T.: Global Optimization Algorithms—Theory and Application. An electronic book available for download at http://www.it-weise.de/projects/book.pdf (2009)
51. Leyffer, S., Mahajan, A.: Software for nonlinearly constrained optimization. Preprint ANL/MCS-P1768-0610, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne (2010)
52. Pintér, J.D.: Software development for global optimization. In: Pardalos, P.M., Coleman, T.F. (eds.) Global Optimization: Methods and Applications. Fields Institute Communications, vol. 55, pp. 183–204. American Mathematical Society, Providence (2009)
53. Kallrath, J. (ed.): Modeling Languages in Mathematical Optimization. Kluwer, Dordrecht (2004)
54. Maplesoft: Maple. Maplesoft, Waterloo (2012). www.maplesoft.com
55. Wolfram Research: Mathematica. Wolfram Research, Champaign (2012). www.wolfram.com
56. The MathWorks: MATLAB. The MathWorks, Natick (2012). www.mathworks.com
57. Pintér, J.D.: LGO—A program system for continuous and Lipschitz optimization. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P.M. (eds.) Developments in Global Optimization, pp. 183–197. Kluwer, Dordrecht (1997)
58. Pintér, J.D.: Nonlinear optimization with GAMS /LGO. J. Global Optim. **38**, 79–101 (2007)
59. Pintér, J.D., Kampas, F.J.: O.R. model development and optimization with Mathematica. In: Golden, B., Raghavan, S., Wasil, E. (eds.) The Next Wave in Computing, Optimization, and Decision Technologies, pp. 285–302. Springer, New York (2005)
60. Pintér, J.D., Linder, D., Chin, P.: Global optimization toolbox for maple: an introduction with illustrative applications. Optim. Method. Softw. **21**(4), 565–582 (2006)
61. Pintér Consulting Services: LGO—A model development and solver system for global–local nonlinear optimization. User's guide. Distributed by Pintér Consulting Services, Halifax (2012). www.pinterconsulting.com
62. Pintér, J.D.: RSS + LGO − a regularly spaced sampling method for experimental design integrated with the LGO solver suite for nonlinear optimization. Project report, TÁMOP 4.2.2-08/01-2008-0021 Project. Széchenyi István University, Győr (2011)
63. Pintér, J.D., Horváth, Z.: Integrated experimental design and nonlinear optimization to handle computationally expensive models under resource constraints. J. Global Optim. Published online March 15 (2012). doi: 10.1007/s10898-012-9882-7 (2012)
64. Lahey Computer Systems: FORTRAN 95 User's Guide. Lahey Computer Systems, Incline Village (2002). www.lahey.com
65. Lahey Computer Systems: FORTRAN 95 Language Reference. Lahey Computer Systems, Incline Village (2002). www.lahey.com

66. INFORMS Computing Society. Mathematical Programming Glossary. http://glossary.computing.society.informs.org/ (2012)
67. Becerra, V.M.: Optimal control. Scholarpedia **3**(1), 5354 (2008)
68. Becerra, V.M.: Solving optimal control problems at no cost with PSOPT. In: Proceedings of IEEE Multiconference on Systems and Control, Yokohama, Japan, 7–10 Sept 2010
69. Betts, J.T.: Practical Methods for Optimal Control Using Nonlinear Programming. SIAM, Philadelphia (2001)
70. Betts, J.T.: Practical Methods for Optimal Control and Estimation Using Nonlinear Programming. SIAM, Philadelphia (2010)
71. Büskens, C., Maurer, H.: SQP-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control. J. Comput. Appl. Math. **120**, 85–108 (2000)
72. Elnagar, G., Kazemi, M.A., Razzaghi, M.: The pseudospectral Legendre method for discretizing optimal control problems. IEEE T. Automat. Contr. **40**, 1793–1796 (1995)
73. Hager, W.: Runge–Kutta methods in optimal control and the transformed adjoint system. Numer. Math. **87**, 247–282 (2000)
74. Halloin, H., Bastie, P.: Laue diffraction lenses for astrophysics: theoretical concepts. Exp. Astron. **20**, 151–170 (2005)
75. Hartl, R.F., Sethi, S.P., Vickson, R.G.A.: A survey of the maximum principles for optimal control problems with state constraints. SIAM Rev. **37**(2), 181–218 (1995)
76. Kirk, D.E.: Optimal Control Theory: An Introduction. Dover, Mineola (2004)
77. Lebedev, L.P., Cloud, M.J.: The Calculus of Variations and Functional Analysis with Optimal Control and Applications in Mechanics. World Scientific, Singapore (2003)
78. Lewis, F.L., Syrmos, V.L.: Optimal Control, 2nd edn. Wiley, New York (1995)
79. Ross, I.M.: A primer on pontryagin's principle in optimal control. Collegiate Publishers, Carmel (2009)
80. Sethi, S., Thompson, G.: Optimal Control Theory: Applications to Management Science and Economics. Kluwer, Norwell (2000)
81. GTOC2: 2nd Global Trajectory Optimization Competition. http://www.esa.int/gsp/ACT/mad/op/GTOC/index.htm (2006). Accessed on April 30, 2012
82. Becerra, V.M.: Practical direct collocation methods for computational optimal control. In: Fasano, G. Pintér, J.D., (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
83. Cassioli, A., Izzo, D., Di Lorenzo, D., Locatelli, M., Schoen, F.: Global optimization approaches for optimal trajectory planning. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
84. Colasurdo, G., Casalino, L.: Indirect methods for the optimization of spacecraft trajectories. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
85. Topputo, F., Belbruno, E.: Optimization of low energy transfers. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
86. Di Lizia, P., Armellin, R., Topputo, F., Bernelli-Zazzera, F., Berz, M.: Global optimization of interplanetary transfers with deep space maneuvers using differential algebra. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
87. Cremaschi, F.: Trajectory optimization for launchers and re-entry vehicles. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
88. Kampas, F.J., Pintér, J.D.: Solving a System of ODEs by the "Shooting Method". An Illustrative Application of *MathOptimizer Professional* described in the User's Guide. Pintér Consulting Services, Canada (2007). www.pinterconsulting.com
89. Büskens, C., Wassel, D.: The ESA NLP solver WORHP. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)

90. Gabrel, V., Murat, C.: Mathematical programming for earth observation satellite mission planning. In: Ciriani, T., Fasano, G., Gliozzi, S., Tadei, R. (eds.) Operations Research in Space and Air, pp. 103–122. Kluwer, Dordrecht (2003)
91. Gürtuna, O., Trépanier, J.: On-orbit satellite servicing: a space-based vehicle routing problem. In: Ciriani, T., Fasano, G., Gliozzi, S., Tadei, R. (eds.) Operations Research in Space and Air, pp. 123–142. Kluwer, Dordrecht (2003)
92. Lang, D.E.: Launch capacity analysis for commercial communications satellites. In: Ciriani, T., Fasano, G., Gliozzi, S., Tadei, R. (eds.) Operations Research in Space and Air, pp. 161–178. Kluwer, Dordrecht (2003)
93. Fasano, G.: A global optimization point of view to handle non-standard object packing problems. J. Global Optim. (2012). DOI: 10.1007/s10898-012-9865-8 (to appear)
94. Fasano, G.: A traffic model for the international space station: an MIP approach. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
95. Fasano, G., Piras, A.: Mission integration optimization techniques in support of timelining, operational sequence definition and resource exploitation. In: Proceedings of the 23rd AIDAA (Italian Association of Aeronautics and Astronautics) Conference, Rome, Italy, 11–15 Sept 1995
96. Padberg, M.W., Rinaldi, G.: A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev. **33**, 60–100 (1991)
97. Williams, H.P.: Model Building in Mathematical Programming, 4th edn. Wiley, Chichester (2000)
98. Cagan, J., Shimada, K., Yin, S.: A survey of computational approaches to three-dimensional layout problems. Comput. Aided Des. **34**, 597–611 (2002)
99. Dyckhoff, H., Scheithauer, G., Terno, J.: Cutting and packing. In: Dell'Amico, M., Maffioli, F., Martello, S. (eds.) Annotated Bibliographies in Combinatorial Optimization, pp. 393–412. Wiley, Chichester (1997)
100. Fekete, S., Schepers, J.: A combinatorial characterization of higher-dimensional orthogonal packing. Math. Oper. Res. **29**, 353–368 (2004)
101. Faroe, O., Pisinger, D., Zachariasen, M.: Guided local search for the three-dimensional bin packing problem. INFORMS J. Comput. **15**(3), 267–283 (2003)
102. Martello, S., Pisinger, D., Vigo, D.: The three-dimensional bin packing problem. Oper. Res. **48**(256), 267 (2000)
103. Martello, S., Pisinger, D., Vigo, D., Den Boef, E., Korst, J.: Algorithm 864: general and robot-packable variants of the three-dimensional bin packing problem. ACM T. Math. Softw. **33**(1) (2007)
104. Pisinger, D., Sigurd, M.M.: The two-dimensional bin packing problem with variable bin sizes and costs. Discrete Optim. **2**(2), 154–167 (2005)
105. Pisinger, D., Sigurd, M.M.: Using decomposition techniques and constraints programming for solving the two-dimensional bin packing problem. INFORMS J. Comput. **19**(1), 36–51 (2006)
106. Dowsland, K.A., Dowsland, W.B., Bennell, J.A.: Jostling for position: local improvement for irregular cutting patterns. J. Oper. Res. Soc. **49**, 647–658 (1998)
107. Egeblad, J., Nielsen, B.K., Odgaard, A.: Fast neighbourhood search for two- and three-dimensional nesting problems. Eur. J. Oper. Res. **183**(3), 1249–1266 (2007)
108. Egeblad, J., Pisinger, D.: Heuristic approaches for the two- and three-dimensional knapsack packing problem. Comput. Oper. Res. **36**, 1026–1049 (2009)
109. Fischetti, M., Luzzi, I.: Mixed-integer programming models for nesting problems. J. Heuristics **15**(3), 201–226 (2009)
110. Gomes, A.M., Oliveira, J.F.: A 2-exchange heuristics for nesting problems. Eur. J. Oper. Res. **141**, 359–570 (2002)
111. Ibaraki, T., Imahori, S., Yagiura, M.: Hybrid metaheuristics for packing problems. In: Blum, C., Blesa Aguilera, M.J., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics: An Emerging

Approach to Optimization. Studies in Computational Intelligence (SCI), vol. 114, pp. 185–219. Springer, Berlin (2008)
112. Kallrath, J.: Cutting circles and polygons from area-minimizing rectangles. J. Global Optim. **43**, 299–328 (2009)
113. Scheithauer, G., Stoyan, Y.G., Romanova, T.Y.: Mathematical modeling of interactions of primary geometric 3D objects. Cybernet. Systems Anal. **41**, 332–342 (2005)
114. Stoyan, Y.G., Scheithauer, G., Gil, N., Romanova, T.Y.: Φ-functions for complex 2D-objects. 4OR (Quart. J. Belg. French Ital. Oper. Res. Soc.) **2**, 69–84 (2004)
115. Teng, H.F., Sun, S.L., Liu, D.Q.: Layout optimization for the objects located within a rotating vessel a three-dimensional packing problem with behavioural constraints. Comput. Oper. Res. **28**(6), 521–535 (2001)
116. Egeblad, J.: Placement of two- and three-dimensional irregular shapes for inertia moment and balance. In: Morabito R, Arenales MN, Yanasse HH (eds.). Int. Trans. Oper. Res. **16**(6), 789–807 (2009) [Special issue on cutting, packing and related problems]
117. Takadama, K., Tokunaga, F., Shimohara, K.: (2004) Capabilities of a multiagent-based cargo layout system for H-II transfer vehicle. In: 16th IFAC Symposium on Automatic Control in Aerospace (ACA'04), pp. 250–255, St. Petersburg, Russia, 14–18 June 2004
118. Junqueira, L., et al.: Optimization models for the three-dimensional container loading problem with practical constraints. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
119. Stoyan, Y.G., Romanova, T.: Mathematical models of placement optimisation: two- and three-dimensional problems and applications. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
120. Bussolino, L., Fasano, G., Novelli, A.: The CAST project. In: Ciriani, T., Fasano, G., Gliozzi, S., Tadei, R. (eds.) Operations Research in Space and Air, pp. 13–26. Kluwer, Dordrecht (2003)
121. Fasano, G., Barrera, M., Lavopa, C., Arguello, L., Steinkopf, M.: Cargo accommodation by interactive engineering: the CAST project. In: 9th international workshop on simulation for european space programmes - SESP 2006- 434950, ESTEC Noordwijk, the Netherlands, 6–8 Nov 2006
122. Fasano, G.: A multi-level MIP-based heuristic approach for the cargo accommodation of a space vehicle.In: 6th ESICUP Meeting, Valencia, Spain, 25–29 Mar 2009
123. Fasano, G., Lavopa, C., Negri, D., Vola, M.C., Castellazzo, A.: Packing optimization problems in space engineering. In: SIMAI Biannual Congresses, Turin, June 25–28 (2012)
124. Fasano, G., Vola, M.C.: Space module on-board stowage optimization by exploiting empty container volumes. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
125. Bandecchi, M., Melton, B., Onagro, F.: Concurrent engineering applied to space mission assessment and design. ESA Bulletin 99, pp. 34–40 (1999)
126. Ciriani, T.A., Sarlo, L.: Operations research applications in space systems development and operations. In: Ciriani, T., Fasano, G., Gliozzi, S., Tadei, R. (eds.) Operations Research in Space and Air, pp. 3–12. Kluwer, Dordrecht (2003)
127. Cassioli, A., Schoen, F.: Global optimization of expensive black box problems with a known lower bound. J. Global Optim. (2011) (to appear)
128. Amata, G.B., Fasano, G., Arcaro, L., Della Croce, F., Norese, M.F., Palamara, S., Tadei, R., Fragnelli, F.: Multidisciplinary Optimisation in Mission Analysis and Design Process. European Space Agency, GSP 1-4487/03/NL/MV, Turin (2004)
129. Pastrone, D., Casalino, L.: Integrated design-trajectory optimization for hybrid rocket motors. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
130. Nicolescu, G., Mosterman, P.: Model-Based Design for Embedded Systems (Computational Analysis, Synthesis, and Design of Dynamic Systems). CRC, Boca Raton (2009)

131. Norstrøm, J.G., Cooke, R.M., Bedford, T.: Value of information based design of control software. In: Ciriani, T., Fasano, G., Gliozzi, S., Tadei, R. (eds.) Operations Research in Space and Air, pp. 179–202. Kluwer, Dordrecht (2003)
132. Edwards, W., Miles, R.F., von Winterfeldt, D. (eds.): Advances in Decision Analysis from Foundations to Applications. Cambridge University Press, Cambridge (2007)
133. Fasano, G.: Simulation and control of microbial contamination within a manned space system. Technical report M95-MI-AI-0057, Thales Alenia Space Italia, Turin (1996)
134. Brauer, F., Castillo-Chavez, C.: Mathematical Models in Population Biology and Epidemiology. Springer, Heidelberg (2000)
135. Mehlem, K.: Optimal magnetic cleanliness modeling of spacecraft. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
136. Migdalas, A., Pardalos, P.M., Värbrand, P. (eds.): From Local to Global Optimization. Kluwer, Dordrecht (2001)
137. Welch, G., Bishop, G.: An introduction to the Kalman filter. SIGGRAPH 2001, Course 8. Department of Computer Science, University of North Carolina, Chapel Hill, NC, USA (2001)
138. Cramer, E.J., Dennis, J.E., Frank, P.D., Lewis, R.M., Shubin, G.R.: Problem formulation for multidisciplinary optimization. SIAM J. Optim. **4**(4), 754–776 (1994)
139. Fasano, G., Lavopa, C., Negri, D., Vola, M.C.: MIP approach for solving the stowage problem on-board the International Space Station. In: 23rd EURO Conference, Bonn, Germany, 5–8 July 2009
140. Fasano, G., Saia, D., Piras, A.: Columbus stowage optimization by CAST (Cargo Accommodation Support Tool). Acta Astronaut. **67**(3), 489–496 (2010)
141. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Modeling Language for Mathematical Programming. Brooks/Cole Publishing Company/Cengage Learning, Independence (2002). See also www.ampl.com (2012)
142. Frontline Systems: Premium Solver Platform Solver Engines. Frontline Systems, Incline Village (2012). See www.solver.com (2012)
143. Kennedy, J.F.: Speech given at Rice University, Houston. (Kennedy spoke about the subject also on other occasions: cf. e.g. http://www.historyplace.com/speeches/jfk-space.htm, http://www.hark.com/clips/nzpntrdvjl-this-nation-should-land-a-man-on-the-moon. Accessed 12 Sept 1962 (2012)
144. LINDO Systems: LINGO. LINDO Systems Inc, Chicago (2012). www.lindo.com (2012)
145. Maximal Software: MPL Modeling System. Maximal Software Inc, Arlington (2012). www.maximal-usa.com (2012)
146. Paragon Decision Technology: AIMMS. Paragon Decision Technology BV, Haarlem (2006). See www.aimms.com (2012)
147. TOMLAB Optimization: TOMLAB. TOMLAB Optimization AB, Västerås (2006). www.tomopt.com (2012)
148. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific Publishing, Cambridge, MA (1999)
149. Fasano, G.: The space-module payload accommodation problem: an MIP formulation. In: Ciriani, T.A., Leachman, R.C. (eds.) Optimization in Industry, **2**, pp. 33–42. John Wiley & Sons, Chichester (1994)
150. Boada, J., Prieur, C., Tarbouriech, S., Pittet C., Charbonnel C.: Formation flying control for satellites: anti-windup based approach. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)

Chapter 2
Practical Direct Collocation Methods for Computational Optimal Control

Victor M. Becerra

Abstract The development of numerical methods for optimal control and, specifically, trajectory optimisation, has been correlated with advances in the fields of space exploration and digital computing. Space exploration presented scientists and engineers with challenging optimal control problems. Specialised numerical methods implemented in software that runs on digital computers provided the means for solving these problems. This chapter gives an introduction to direct collocation methods for computational optimal control. In a direct collocation method, the state is approximated using a set of basis functions, and the dynamics are collocated at a given set of points along the time interval of the problem, resulting in a sparse nonlinear programming problem. This chapter concentrates on local direct collocation methods, which are based on low-order basis functions employed to discretise the state variables over a time segment. This chapter includes sections that discuss important practical issues such as multi-phase problems, sparse nonlinear programming solvers, efficient differentiation, measures of accuracy of the discretisation, mesh refinement, and potential pitfalls. A space relevant example is given related to a four-phase vehicle launch problem.

Keywords Optimal control • Nonlinear programming • Collocation methods

2.1 Introduction to Optimal Control Problems and Their Formulation

Optimal control is the process of finding control and state trajectories for a dynamic system over a period of time to so that the performance of the system is optimal in some specified sense. The index which is used to quantify the performance of the

V.M. Becerra (✉)
School of Systems Engineering, University of Reading, Reading RG6 6AY, UK
e-mail: v.m.becerra@reading.ac.uk

system might include, for example, a measure of the control effort, a measure of the tracking error, a measure of energy consumption, a measure of the amount of time taken to reach a target, or any other quantity of importance to the operation of the system.

There are various types of optimal control problems, depending on the performance index, the type of time domain (continuous, discrete), the presence of different types of constraints, and what variables are free to be chosen. The formulation of an optimal control problem usually requires:

- A mathematical model of the system to be controlled
- A specification of the performance index
- A specification of all boundary conditions on states, and constraints to be satisfied by states and controls
- A statement of what variables are free

2.1.1 Formulation

In the continuous time case, the dynamics of a system are typically defined by a set of ordinary differential equations (ODEs), which, if written in explicit form, can be expressed as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{F}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t], t \in [t_0, t_f], \tag{2.1}$$

where $\mathbf{x} : [t_0, t_f] \rightarrow \mathscr{R}^{n_x}$ is the state vector function, $\mathbf{u} : [t_0, t_f] \rightarrow \mathscr{R}^{n_u}$ is the control vector function, $\mathbf{p} \in \mathscr{R}^{n_p}$ is a vector of static parameters which are independent of t, and $t \in [t_0, t_f] \subset \mathscr{R}$ is an independent variable, which is usually time.

The initial and terminal conditions can be expressed as a set of inequality constraints, which are often called event constraints:

$$\mathbf{e}_L \leq \mathbf{e}[\mathbf{x}(t_0), \mathbf{u}(t_0), \mathbf{x}(t_f), \mathbf{u}(t_f), \mathbf{p}, t_0, t_f] \leq \mathbf{e}_U. \tag{2.2}$$

Sometimes, the problem involves time-dependent constraints on the states and/or control variables, which can be written in the form of time-dependent inequality and are often called path constraints:

$$\mathbf{h}_L \leq \mathbf{h}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t] \leq \mathbf{h}_U,\ t \in [t_0, t_f]. \tag{2.3}$$

There are often bound constraints on controls, states, and static parameters:

$$\begin{aligned} &\mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U,\ t \in [t_0, t_f], \\ &\mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U,\ t \in [t_0, t_f], \\ &\mathbf{p}_L \leq \mathbf{p} \leq \mathbf{p}_U. \end{aligned} \tag{2.4}$$

The following constraints allow for cases where the initial and/or final times are not fixed but are to be chosen as part of the solution to the problem:

$$\begin{aligned} \underline{t}_0 &\leq t_0 \leq \bar{t}_0, \\ \underline{t}_f &\leq t_f \leq \bar{t}_f, \\ t_f - t_0 &\geq 0. \end{aligned} \tag{2.5}$$

In the above expressions, **f**, **h**, and **e** are functions defined as follows:

$$\begin{aligned} \mathbf{F} &: \mathscr{R}^{n_x} \times \mathscr{R}^{n_u} \times \mathscr{R}^{n_p} \times [t_0, t_f] \mapsto \mathscr{R}^{n_x} \\ \mathbf{h} &: \mathscr{R}^{n_x} \times \mathscr{R}^{n_u} \times \mathscr{R}^{n_p} \times [t_0, t_f] \mapsto \mathscr{R}^{n_h} \\ \mathbf{e} &: \mathscr{R}^{n_x} \times \mathscr{R}^{n_u} \times \mathscr{R}^{n_x} \times \mathscr{R}^{n_u} \times \mathscr{R}^{n_p} \times \mathscr{R} \times \mathscr{R} \mapsto \mathscr{R}^{n_e}. \end{aligned} \tag{2.6}$$

Note that some of the above inequality constraints could be equality constraints if the lower bounds are equal to the corresponding upper bounds.

An optimal control problem (henceforth called Problem 1) then involves finding all the free functions and variables: the controls $\mathbf{u}(t), t \in [t_0, t_f]$, the states $\mathbf{x}(t)$, $t \in [t_0, t_f]$, the parameters vector $\mathbf{p}$, and the times t_0, and t_f, to minimise the following performance index, while satisfying the above constraints (2.1)–(2.5):

$$J_1 = \varphi_1[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f], \tag{2.7}$$

where $\varphi_1 : \mathscr{R}^{n_x} \times \mathscr{R} \times \mathscr{R}^{n_x} \times \mathscr{R} \mapsto \mathscr{R}$ is the cost function. Sometimes, the performance index to be optimised has the form

$$J_1 = \varphi_1[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f] + \int_{t_0}^{t_f} L[\mathbf{x}(t), \mathbf{u}(t), t] dt, \tag{2.8}$$

where $L : \mathscr{R}^{n_x} \times \mathscr{R}^{n_u} \times \mathscr{R} \mapsto \mathscr{R}$ is often known as the *running cost* function. A problem involving an objective function with an integral such as Eq. (2.8) can be reduced to the form of Problem 1 by adding an extra state whose time derivative is equal to the integrand L, and accounting for the integral as the value of the additional state at the final time.

2.1.2 Optimality Conditions

The optimality conditions of problems similar to Problem 1 are rather involved and have been studied over the years [7, 14]. For the purposes of the analyses to be presented later in this chapter, a special case of Problem 1 will be considered. The simplified problem (henceforth called Problem 2) has the following form.

Find the control vector trajectory $\mathbf{u} : [t_0, t_f] \subset \mathscr{R} \mapsto \mathscr{R}^{n_u}$ to minimise the performance index:

$$J = \varphi[\mathbf{x}(t_f), t_f]$$

subject to

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] \\ \mathbf{x}(t_0) &= \mathbf{x}_0, \end{aligned} \tag{2.9}$$

where $[t_0, t_f]$ is the time interval of interest, which in this case is fixed. $\varphi : \mathscr{R}^{n_x} \times \mathscr{R} \mapsto \mathscr{R}$ is a terminal cost function, $\mathbf{f} : \mathscr{R}^{\mathbf{n_x}} \times \mathscr{R}^{\mathbf{n_u}} \times \mathscr{R} \mapsto \mathscr{R}^{\mathbf{n_x}}$.

The first-order necessary conditions for a minimum of the performance index J follow [7]. Dynamic constraints

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t], \tag{2.10}$$

adjoint equation

$$\dot{\lambda}(t) = -\nabla_{\mathbf{x}} H, \tag{2.11}$$

transversality conditions

$$\begin{aligned} \lambda(t_f) &= \nabla_{\mathbf{x}} \varphi|_{t=t_f}, \\ \mathbf{x}(t_0) &= \mathbf{x}_0, \end{aligned} \tag{2.12}$$

and stationarity condition

$$\nabla_{\mathbf{u}} H = \mathbf{0}, \tag{2.13}$$

where $\lambda : [t_0, t_f] \mapsto \mathfrak{R}^{n_x}$ is an adjoint function, and the Hamiltonian function H is defined as follows:

$$H[\mathbf{x}(t), \mathbf{u}(t), \lambda(t), t] = \lambda(t)^T \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]. \tag{2.14}$$

It is assumed here that the derivatives involved in the necessary optimality conditions (2.9)–(2.13) exist and also that a solution $(\mathbf{x}(t), \mathbf{u}(t), \lambda(t)), t \in [t_0, t_f]$ of Problem 2 exists which minimises the performance index [15].

2.2 Nonlinear Programming

Nonlinear programming (NLP) [3, 15] involves finding a decision vector $\mathbf{y} \in \mathscr{R}^{n_y}$ to minimise:

$$f(\mathbf{y})$$

subject to

$$\mathbf{h}_l \leq \mathbf{h}(\mathbf{y}) \leq \mathbf{h}_u,$$
$$\mathbf{y}_l \leq \mathbf{y} \leq \mathbf{y}_u,$$

where $f : \mathscr{R}^{n_y} \mapsto \mathscr{R}$ is a twice continuously differentiable scalar function and $\mathbf{h} : \mathscr{R}^{n_y} \mapsto \mathscr{R}^{n_h}$ is a twice continuously differentiable vector function.

A particular case that is simple to analyse occurs when the NLP problem involves only equality constraints, such that the problem can be expressed as follows. This is henceforth referred to as Problem 3. Choose $\mathbf{y}$ to minimise

$$f(\mathbf{y})$$

subject to

$$\mathbf{c}(\mathbf{y}) = \mathbf{0},$$

where $\mathbf{c} : \mathscr{R}^{n_y} \mapsto \mathscr{R}^{n_c}$ is a twice continuously differentiable vector function and $n_c \leq n_y$. Introduce the Lagrangian function:

$$\mathscr{L}[\mathbf{y}, \lambda] = f(\mathbf{y}) + \lambda^T \mathbf{c}(\mathbf{y}), \tag{2.15}$$

where $\mathscr{L} : \mathscr{R}^{n_y} \times \mathscr{R}^{n_c} \mapsto \mathscr{R}$ is a scalar function and $\lambda \in \mathscr{R}^{n_c}$ is a Lagrange multiplier. The first-order necessary conditions for a point $(\mathbf{y}^*, \lambda)$ to be a local minimiser are as follows [3, 15]:

$$\begin{aligned} \nabla_{\mathbf{y}} \mathscr{L}[\mathbf{y}^*, \lambda] &= \nabla_{\mathbf{y}} f(\mathbf{y}^*) + [\mathbf{c}_{\mathbf{y}}(\mathbf{y}^*)]^T \lambda = \mathbf{0}, \\ \nabla_{\lambda} \mathscr{L}[\mathbf{y}^*, \lambda] &= \mathbf{c}(\mathbf{y}^*) = \mathbf{0}, \end{aligned} \tag{2.16}$$

where $\mathbf{c}_{\mathbf{y}}(\mathbf{y}^*) = \partial \mathbf{c}(\mathbf{y})/\partial \mathbf{y}$ is the Jacobian of the constraints evaluated at $\mathbf{y}^*$. Note that Eq. (2.16) is a system of $n_y + n_c$ Equations with $n_y + n_c$ unknowns.

If the NLP problem also includes a set of inequality constraints, then the resulting problem (henceforth called Problem 4) is as follows.

Choose $\mathbf{y}$ to minimise

$$f(\mathbf{y})$$

subject to

$$\mathbf{c}(\mathbf{y}) = \mathbf{0},$$
$$\mathbf{q}(\mathbf{y}) \leq \mathbf{0},$$

where $\mathbf{q} : \mathscr{R}^{n_y} \mapsto \mathscr{R}^{n_q}$ is a twice continuously differentiable vector function. The Lagrangian function is redefined to include the inequality constraints function:

$$\mathscr{L}[\mathbf{y}, \lambda, \mu] = f(\mathbf{y}) + \lambda^T \mathbf{c}(\mathbf{y}) + \mu^T \mathbf{q}(\mathbf{y}), \tag{2.17}$$

where $\mu \in \mathscr{R}^{n_q}$ is a vector of Karush–Kuhn–Tucker multipliers. In this case, the first-order necessary conditions for a local minimiser are the well-known Karush–Kuhn–Tucker conditions [3], which can be expressed as follows:

$$\begin{aligned}
\mu &\geq 0,\\
\nabla_{\mathbf{y}}\mathscr{L}[\mathbf{y}^*,\lambda,\mu] &= \nabla_{\mathbf{y}}f(\mathbf{y}^*) + [\mathbf{c}_{\mathbf{y}}(\mathbf{y}^*)]^T\lambda + [\mathbf{q}_{\mathbf{y}}(\mathbf{y}^*)]^T\mu = \mathbf{0},\\
\nabla_{\lambda}\mathscr{L}[\mathbf{y}^*,\lambda] &= \mathbf{c}(\mathbf{y}^*) = \mathbf{0},\\
\mu^T\mathbf{q}(\mathbf{y}^*) &= 0.
\end{aligned} \tag{2.18}$$

Methods for the solution of NLP problems are well established. Often, these methods involve the solution of a sequence of quadratic programming subproblems. A quadratic programming problem is a special type of NLP problem where the constraints are linear and the objective is a quadratic function. Current NLP implementations incorporate developments in numerical linear algebra that exploit sparsity in matrices, such that it is common to numerically solve NLP problems involving variables and constraints in the order of hundreds of thousands. Popular implementations of NLP methods include SNOPT [11] and IPOPT [21].

2.3 Indirect Methods for Solving Optimal Control Problems

Indirect methods involve iterating on the necessary optimality conditions to seek their satisfaction. This usually involves attempting to solve nonlinear two-point boundary-value problems, through the forward integration of the plant equations and the backward integration of the adjoint equations (which can be ill-conditioned). These methods require to explicitly derive the necessary conditions (adjoint equations, transversality conditions, minimum principle). The region of convergence of indirect methods tends to be quite narrow, so that these methods require good initial guesses, which includes guesses of the adjoint functions. In the case of problems with inequality path constraints, indirect methods require a priori estimates of the sequence of constrained arcs, which may be difficult to guess.

2.4 Direct Collocation Methods

Direct collocation methods involve the discretisation of the differential equations using, for example, trapezoidal, Hermite–Simpson [4], or pseudospectral approximations [9], by defining a grid of N points covering the time interval $[t_0, t_f]$, $t_0 = t_1 < t_2 \cdots < t_N = t_f$. This way, the differential equations become a finite set of equality constraints of the NLP problem.

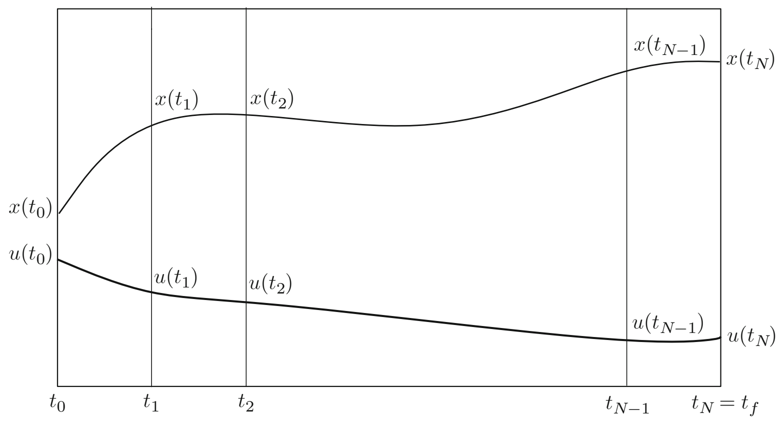


Fig. 2.1 Discretisation of the interval $t \in [t_0, t_f] \subset \mathscr{R}$. The figure illustrates the idea that the control u and state variables x are discretised over the resulting grid

The nonlinear optimisation problems that arise from direct collocation methods may be very large, having possibly hundreds to tens of thousands of variables and constraints. For instance, [6] describes a spacecraft low-thrust trajectory design with 211,031 variables and 146,285 constraints. It is, however, interesting that such large NLP problems are easier to solve than boundary-value problems. The reason for the relative ease of computation of direct collocation methods is that the associated NLP problem is often quite sparse and efficient methods and software exist for its solution [21, 5]. With direct collocation methods there is no need to explicitly derive the necessary conditions of the continuous problem. This makes the method more attractive in complex cases. Moreover, direct collocation methods do not require an a priori specification of the sequence of constrained arcs in problems with inequality path constraints. As a result of the above advantages, the range of problems that can be solved with direct methods is larger than the range of problems that can be solved via indirect methods.

Consider the continuous domain $[t_0, t_f] \subset \mathscr{R}$, and break this domain using intermediate nodes as follows:

$$t_0 < t_1 \leq \cdots < t_N = t_f \tag{2.19}$$

such that the domain has been divided into the N segments $[t_k, t_{k+1}], j = 0, \ldots, N-1$ (Fig. 2.1).

By using direct collocation methods, it is possible to discretise Problem 1 to obtain a NLP problem. The following problem is henceforth referred to as Problem 5. Choose $\mathbf{y} \in \mathscr{R}^{n_y}$ to minimise:

$$F(\mathbf{y}) \tag{2.20}$$

subject to:

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{H}_L \\ \mathbf{e}_L \end{bmatrix} \leq \begin{bmatrix} \mathbf{Z}(\mathbf{y}) \\ \mathbf{H}(\mathbf{y}) \\ \mathbf{E}(\mathbf{y}) \end{bmatrix} \leq \begin{bmatrix} \mathbf{0} \\ \mathbf{H}_U \\ \mathbf{e}_U \end{bmatrix} \tag{2.21}$$

$$\mathbf{y}_L \leq \mathbf{y} \leq \mathbf{y}_U, \tag{2.22}$$

where $F : \mathscr{R}^{n_y} \mapsto \Re$ is simply a mapping resulting from the evaluation of the performance index (2.7), $\mathbf{Z} : \mathscr{R}^{n_y} \mapsto \mathscr{R}^{n_z}$ is the mapping that arises from the discretisation of the differential constraints (2.1) over the grid points, $\mathbf{H} : \mathscr{R}^{n_y} \mapsto \mathscr{R}^{(N+1)n_h}$ arises from the evaluation of the path constraints (2.3) at each of the grid points, with associated bounds $\mathbf{H}_L, \mathbf{H}_U \in \mathscr{R}^{(N+1)n_h}$, and $\mathbf{E} : \mathscr{R}^{n_y} \mapsto \mathscr{R}^{n_e}$ is simply a mapping resulting from the evaluation of the event constraints (2.2). The form of Problem 5 is the same regardless of the method used to discretise the dynamics. What may change depending on the discretisation method used is the decision vector $\mathbf{y}$ and the differential defect constraints. This is discussed in more detail in subsequent sections of this chapter.

2.4.1 Local Methods

Suppose that the solution of the ODE (2.1) is approximated by a polynomial $\tilde{\mathbf{x}}(t)$ of degree M over each interval $t \in [t_k, t_{k+1}], j = 0, \ldots, N-1$:

$$\tilde{\mathbf{x}}(t) = \mathbf{a}_0^{(k)} + \mathbf{a}_1^{(k)}(t - t_k) + \cdots + \mathbf{a}_M^{(k)}(t - t_k)^M, \tag{2.23}$$

where the polynomial coefficients $\left\{\mathbf{a}_0^{(k)}, \mathbf{a}_1^{(k)}, \ldots, \mathbf{a}_M^{(k)}\right\}$ are chosen such that the approximation matches the function at the beginning and at the end of the interval:

$$\begin{aligned} \tilde{\mathbf{x}}(t_k) &= \mathbf{x}(t_k) \\ \tilde{\mathbf{x}}(t_{k+1}) &= \mathbf{x}(t_{k+1}) \end{aligned} \tag{2.24}$$

and the time derivative of the approximation matches the time derivative of the function at t_k and $t_{k\,+\,1}$:

$$\begin{aligned} \frac{\mathrm{d}\tilde{\mathbf{x}}(t_k)}{\mathrm{d}t} &= \mathbf{f}[\mathbf{x}(t_k), \mathbf{u}(t_k), \mathbf{p}, t_k], \\ \frac{\mathrm{d}\tilde{\mathbf{x}}(t_{k+1})}{\mathrm{d}t} &= \mathbf{f}[\mathbf{x}(t_{k+1}), \mathbf{u}(t_{k+1}), \mathbf{p}, t_{k+1}]. \end{aligned} \tag{2.25}$$

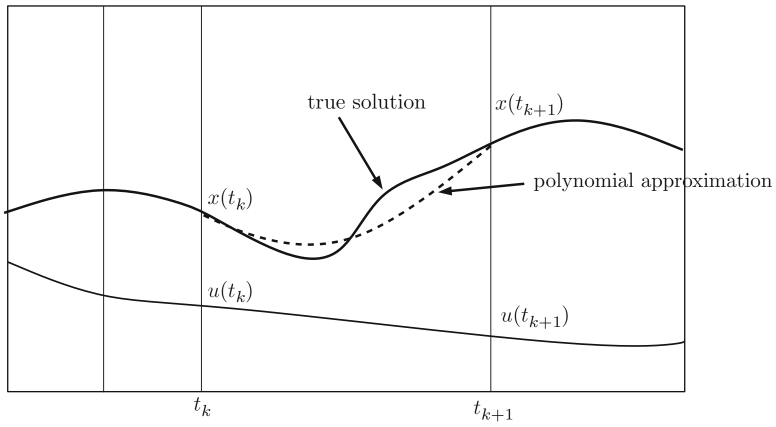


Fig. 2.2 The collocation concept. The solution of the ODE (2.1) is approximated by a polynomial of degree M over each interval, such that the approximation matches at the beginning and the end of the interval both in value and in first derivative

Equations (2.24) and (2.25) are called *collocation conditions* (Fig. 2.2).

Many of the discretisations used in practice for solving optimal control problems belong to the class of so-called classical Runge–Kutta methods [5]. A general K-stage Runge–Kutta method discretises the differential equation as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h_k \sum_{i=1}^{K} b_i \mathbf{f}_{ki}, \tag{2.26}$$

where

$$\mathbf{x}_{ki} = \mathbf{x}_k + h_k \sum_{l=1}^{K} a_{il} \mathbf{f}_{kl}, \tag{2.27}$$

$$\mathbf{f}_{ki} = \mathbf{f}[\mathbf{x}_{ki}, \mathbf{u}_{ki}, \mathbf{p}, t_{ki}], \tag{2.28}$$

where $\mathbf{u}_{ki} = \mathbf{u}(t_{ki})$, $t_{ki} = t_k + h_k \rho_i$, and $\rho_i = (0, 1]$.

2.4.1.1 Trapezoidal Method

The trapezoidal method is based on a quadratic interpolation polynomial, such that Eq. (2.23) becomes

$$\tilde{\mathbf{x}}(t) = \mathbf{a}_0^{(k)} + \mathbf{a}_1^{(k)}(t - t_k) + \mathbf{a}_2^{(k)}(t - t_k)^2. \tag{2.29}$$

Evaluating Eq. (2.29) at node t_k,

$$\tilde{\mathbf{x}}(t_k) = \mathbf{a}_0^{(k)} = \mathbf{x}(t_k). \tag{2.30}$$

Taking the time derivative of the interpolant (2.29), evaluating it at t_k and t_{k+1}, and matching with the corresponding right-hand side of Eq. (2.1), results in the following:

$$\begin{aligned}\frac{\mathrm{d}\tilde{\mathbf{x}}(t_k)}{\mathrm{d}t} &= \mathbf{a}_1^{(k)} = \mathbf{f}[\mathbf{x}(t_k), \mathbf{u}(t_k), \mathbf{p}, t_k] \equiv \mathbf{f}_k,\\ \frac{\mathrm{d}\tilde{\mathbf{x}}(t_{k+1})}{\mathrm{d}t} &= \mathbf{a}_1^{(k)} + 2\mathbf{a}_2^{(k)}(t_{k+1} - t_k) = \mathbf{f}[\mathbf{x}(t_{k+1}), \mathbf{u}(t_{k+1}), \mathbf{p}, t_{k+1}] \equiv \mathbf{f}_{k+1}.\end{aligned} \tag{2.31}$$

Using Eqs. (2.30) and (2.31), the interpolating polynomial is

$$\tilde{\mathbf{x}}(t) = \mathbf{x}(t_k) + \mathbf{f}_k(t - t_k) + \frac{1}{2}\frac{(\mathbf{f}_{k+1} - \mathbf{f}_k)}{(t_{k+1} - t_k)}(t - t_k)^2, \tag{2.32}$$

and its time derivative evaluated at t_k is

$$\frac{\mathrm{d}\tilde{\mathbf{x}}(t_k)}{\mathrm{d}t} = \mathbf{f}_k + \frac{(\mathbf{f}_{k+1} - \mathbf{f}_k)}{(t_{k+1} - t_k)}(t - t_k). \tag{2.33}$$

Evaluating the interpolating polynomial (2.32) at t_{k+1} gives

$$\begin{aligned}\tilde{\mathbf{x}}(t_{k+1}) &= \mathbf{x}(t_k) + \mathbf{f}_k(t_{k+1} - t_k) + \frac{1}{2}(\mathbf{f}_{k+1} - \mathbf{f}_k)(t_{k+1} - t_k)\\ &= \mathbf{x}(t_k) + \frac{1}{2}(\mathbf{f}_k + \mathbf{f}_{k+1})(t_{k+1} - t_k),\end{aligned} \tag{2.34}$$

but since $\tilde{\mathbf{x}}(t_{k+1}) = \mathbf{x}(t_{k+1})$, then Eq. (2.34) can be expressed as follows:

$$\zeta(t_k) = \mathbf{x}(t_{k+1}) - \mathbf{x}(t_k) - \frac{h_k}{2}(\mathbf{f}_k + \mathbf{f}_{k+1}) = \mathbf{0}, \tag{2.35}$$

where $h_k = t_{k+1} - t_k$ and $\zeta(t_k) = \mathbf{0}$ is the differential defect constraint at node t_k associated with the trapezoidal method. Allowing $j = 0, \ldots, N-1$, Eq. (2.35) generates Nn_x differential defect constraints.

Note that the trapezoidal method is a 2-stage Runge–Kutta method with $a_{11} = a_{12} = 0$, $a_{21} = a_{22} = 1/2$, $b_1 = b_2 = 1/2$, $\rho_1 = 0$, and $\rho_2 = 1$. Equation (2.35), which is known as the *compressed form* of the trapezoidal method, is often used in optimal control implementations as it results in an NLP problem with less variables and constraints, compared with the uncompressed form that results from applying Eqs. (2.26)–(2.28). The uncompressed form of the trapezoidal method is as follows:

$$\begin{aligned} \mathbf{x}(t_{k+1}) - \mathbf{x}(t_k) - \frac{h_k}{2}(\mathbf{f}_k + \mathbf{f}_{k2}) &= \mathbf{0}, \\ \mathbf{x}_{k2} - \mathbf{x}_k - \frac{h_k}{2}(\mathbf{f}_k + \mathbf{f}_{k2}) &= \mathbf{0}, \end{aligned} \tag{2.36}$$

where $\mathbf{f}_{k2} = \mathbf{f}[\mathbf{x}_{k2}, \mathbf{u}(t_{k+1}), \mathbf{p}, t_{k+1}]$. It is worth noting that the compressed and uncompressed forms are mathematically equivalent, but there are some differences in their convergence behaviour (see Sect. 2.4.1.4).

Using direct collocation with the compressed trapezoidal discretisation (2.35), the decision vector for single-phase problems is given by

$$\mathbf{y} = \begin{bmatrix} vec(\mathbf{U}^{N-1}) \\ vec(\mathbf{X}^N) \\ \mathbf{p} \\ t_0 \\ t_f \end{bmatrix}, \tag{2.37}$$

where $vec(\mathbf{A})$ represents the vectorisation of matrix $\mathbf{A}$, that is an operator that stacks all columns of the matrix, one below the other, in a single column vector,

$$\mathbf{U}^{N-1} = [\mathbf{u}(t_0)\ \mathbf{u}(t_1)\ \ldots \mathbf{u}(t_{N-1})] \in \mathscr{R}^{n_u \times N}, \tag{2.38}$$

$$\mathbf{X}^N = [\mathbf{x}(t_0)\ \mathbf{x}(t_1)\ \ldots \mathbf{x}(t_N)] \in \mathscr{R}^{n_x \times N+1}. \tag{2.39}$$

2.4.1.2 Hermite–Simpson Method

The Hermite–Simpson method is based on a cubic interpolating polynomial, such that Eq. (2.23) becomes:

$$\tilde{\mathbf{x}}(t) = \mathbf{a}_0^{(k)} + \mathbf{a}_1^{(k)}(t - t_k) + \mathbf{a}_2^{(k)}(t - t_k)^2 + \mathbf{a}_3^{(k)}(t - t_k)^3. \tag{2.40}$$

Evaluating Eq. (2.40) at node t_k,

$$\tilde{\mathbf{x}}(t_k) = \mathbf{a}_0^{(k)} \equiv \mathbf{x}(t_k). \tag{2.41}$$

The time derivative of the interpolating polynomial (2.40) is:

$$\frac{\mathrm{d}\tilde{\mathbf{x}}(t)}{\mathrm{d}t} = \mathbf{a}_1^{(k)} + 2\mathbf{a}_2^{(k)}(t - t_k) + 3\mathbf{a}_3^{(k)}(t - t_k)^2. \tag{2.42}$$

Evaluating the time derivative of the interpolant (2.42) at the beginning t_k, end t_{k+1}, and midpoint $\bar{t}_k = (t_k + t_{k+1})/2$ of the interval, and matching them with the corresponding right-hand side of Eq. (2.1), results in the following expression:

$$\frac{\mathrm{d}\tilde{\mathbf{x}}(t_k)}{dt} = \mathbf{a}_1^{(k)} \equiv \mathbf{f}_k, \tag{2.43}$$

$$\frac{\mathrm{d}\tilde{\mathbf{x}}(t_{k+1})}{dt} = \mathbf{a}_1^{(k)} + 2\mathbf{a}_2^{(k)} h_k + 3a_3^{(k)} h_k^2 \equiv \mathbf{f}_{k+1}, \tag{2.44}$$

$$\frac{\mathrm{d}\tilde{\mathbf{x}}(\bar{t}_j)}{\mathrm{d}t} = \mathbf{a}_1^{(k)} + 2\mathbf{a}_2^{(k)} \bar{h}_k + 3\mathbf{a}_3^{(k)} \bar{h}_k^2 \equiv \bar{\mathbf{f}}_k, \tag{2.45}$$

where $\bar{h}_k = \bar{t}_k - t_k$ and $\bar{\mathbf{f}}_k = \mathbf{f}[\tilde{\mathbf{x}}(\bar{t}_k), \mathbf{u}(\bar{t}_k), \mathbf{p}, \bar{t}_k]$. Note that $\bar{\mathbf{f}}_k$ depends on the interpolated state at the midpoint of the interval $\tilde{\mathbf{x}}(\bar{t}_k)$ (see Eq. (2.50) below), and on the control also at the midpoint $\mathbf{u}(\bar{t}_k)$. Equations (2.41) and (2.43)–(2.45) result in the following interpolating polynomial:

$$\begin{aligned} \tilde{\mathbf{x}}(t) = \mathbf{x}(t_k) + \mathbf{f}_k(t - t_k) + \frac{(4\bar{\mathbf{f}}_k - \mathbf{f}_{k+1} - 3\mathbf{f}_k)}{2h_k}(t - t_k)^2 \\ + \frac{2(\mathbf{f}_k + \mathbf{f}_{k+1} - 2\bar{\mathbf{f}}_k)}{3h_k^2}(t - t_k)^3. \end{aligned} \tag{2.46}$$

Evaluating Eq. (2.46) at the end of the interval gives

$$\begin{aligned} \tilde{\mathbf{x}}(t_{k+1}) &= \mathbf{x}(t_k) + \mathbf{f}_k h_k + \frac{(4\bar{\mathbf{f}}_k - \mathbf{f}_{k+1} - 3\mathbf{f}_k)}{2} h_k + \frac{2(\mathbf{f}_k + \mathbf{f}_{k+1} - 2\bar{\mathbf{f}}_k)}{3} h_k \\ &= \mathbf{x}(t_k) + \frac{h_k}{6}\left[\mathbf{f}_k + 4\bar{\mathbf{f}}_k + \mathbf{f}_{k+1}\right], \end{aligned} \tag{2.47}$$

but from the collocation condition $\tilde{\mathbf{x}}(t_{k+1}) = \mathbf{x}(t_{k+1})$, then Eq. (2.47) can be expressed as follows:

$$\zeta(t_k) = \mathbf{x}(t_{k+1}) - \mathbf{x}(t_k) - \frac{h_k}{6}\left[\mathbf{f}_k + 4\bar{\mathbf{f}}_k + \mathbf{f}_{k+1}\right] = \mathbf{0}, \tag{2.48}$$

where $\zeta(t_k) = \mathbf{0}$ is the differential defect constraint at node t_k associated with the Hermite–Simpson method. Evaluating Eq. (2.46) at the midpoint of the interval

$$\begin{aligned} \tilde{\mathbf{x}}(\bar{t}_k) &= \mathbf{x}(t_k) + \frac{h_k}{2}\mathbf{f}_k + \frac{h_k}{8}(4\bar{\mathbf{f}}_k - \mathbf{f}_{k+1} - 3\mathbf{f}_k) + \frac{h_k}{12}(\mathbf{f}_k + \mathbf{f}_{k+1} - 2\bar{\mathbf{f}}_k) \\ &= \mathbf{x}(t_k) + \frac{h_k}{24}\left[5\mathbf{f}_k + 8\bar{\mathbf{f}}_k - \mathbf{f}_{k+1}\right]. \end{aligned} \tag{2.49}$$

Solving Eq. (2.47) for $\bar{\mathbf{f}}_k$ and replacing in Eq. (2.49) results in

$$\tilde{\mathbf{x}}(\bar{t}_k) = \frac{1}{2}[\mathbf{x}(t_k) + \mathbf{x}(t_{k+1})] + \frac{h_k}{8}[\mathbf{f}_k - \mathbf{f}_{k+1}]. \tag{2.50}$$

Equation (2.50) gives the interpolated value of the state at the midpoint of the interval, which is needed to evaluate $\bar{\mathrm{f}}_k$. Note that the evaluation of Eq. (2.50) and the substitution of $\tilde{x}(\bar{t}_k)$ into Eq. (2.48) results in the so-called *compressed Hermite–Simpson* method. Allowing $j = 0, \ldots, N-1$, Eq. (2.48) generates Nn_x differential defect constraints. Using direct collocation with the compressed Hermite–Simpson method, the decision vector for single-phase problems is given by

$$\mathbf{y} = \begin{bmatrix} vec(\mathbf{U}^N) \\ vec(\bar{\mathbf{U}}^{N-1}) \\ vec(\mathbf{X}^N) \\ \mathbf{p} \\ t_0 \\ t_f \end{bmatrix}, \tag{2.51}$$

where

$$\bar{\mathbf{U}}^{N-1} = [\mathbf{u}(\bar{t}_0)\ \mathbf{u}(\bar{t}_1)\ \ldots \mathbf{u}(\bar{t}_{N-1})] \in \mathscr{R}^{n_u \times N-1}. \tag{2.52}$$

Note that the Hermite–Simpson method is a 3-stage Runge–Kutta method with $a_{11} = a_{12} = a_{13} = 0$, $a_{21} = 5/24$, $a_{22} = 1/3$, $a_{23} = -1/24$, $a_{31} = 1/6$, $a_{32} = 2/3$, $a_{33} = 1/6$, $b_1 = 1/6$, $b_2 = 2/3$, $b_3 = 1/6$, $\rho_1 = 0$, $\rho_2 = 1/2$, and $\rho_3 = 1$. If the midpoint states are also considered decision variables and rewriting Eq. (2.50) as an equality constraint

$$\tilde{\mathbf{x}}(\bar{t}_k) - \frac{1}{2}[\mathbf{x}(t_k) + \mathbf{x}(t_{k+1})] - \frac{h_k}{8}[\mathbf{f}_k - \mathbf{f}_{k+1}] = \mathbf{0}, \tag{2.53}$$

then the use of Eq. (2.53) together with Eq. (2.48) results in the so-called *uncompressed or separated Hermite–Simpson* method.

Using direct collocation with the uncompressed Hermite–Simpson method, the decision vector for single-phase problems is given by

$$\mathbf{y} = \begin{bmatrix} vec(\mathbf{U}^N) \\ vec(\bar{\mathbf{U}}^{N-1}) \\ vec(\mathbf{X}^N) \\ vec(\bar{\mathbf{X}}^{N-1}) \\ \mathbf{p} \\ t_0 \\ t_f \end{bmatrix}, \tag{2.54}$$

where

$$\bar{\mathbf{X}}^{N-1} = [\mathbf{x}(\bar{t}_0)\ \mathbf{x}(\bar{t}_1)\ \ \ldots \mathbf{x}(\bar{t}_{N-1})] \in \mathscr{R}^{n_x \times N-1}. \tag{2.55}$$

2.4.1.3 Optimality of the Discretised Problem

Assume that Problem 2 is discretised using either the trapezoidal or the Hermite–Simpson methods described above. It is relatively easy to show that the discrete necessary optimality conditions (2.16) of the equality constrained NLP problem that results converge to the continuous first-order necessary optimality conditions of Problem 2 (2.10)–(2.13), when the maximum discretisation step size goes to zero ($N \to \infty$, so that $h \to 0$, with $h = \max_k h_k$). Similar equivalences can be made between the Karush–Kuhn–Tucker conditions (2.18) corresponding, for instance, to Problem 5, and the first-order optimality conditions of the associated continuous problem (Problem 1) [4, 14].

2.4.1.4 Convergence

This section introduces some ideas on the convergence properties of direct collocation methods for optimal control. The error in variable $z_k \equiv z(t_k)$ with respect to a solution $z^*(t)$ is of order n if there exist $\bar{h}, c{>}0$ such that

$$\max_k ||\tilde{z}_k - z^*(t_k)|| \ \leq ch^n, \quad \forall h < \bar{h}.$$

It is well known that the convergence of the trapezoidal method for integrating differential equations is of order 2, while the convergence of the Hermite–Simpson method for integrating differential equations is of order 4 [13]. Moreover, it has been proved [12] that, provided the optimal control problem (Problem 2) satisfies certain smoothness and coercivity conditions (which are well known [12]), (1) the discretisation has a strict local minimiser and an associated adjoint variable, (2) in particular, the uncompressed trapezoidal method is of order 2 for optimal control (Problem 2) in the states x and Lagrange multipliers λ, and (3) the uncompressed Hermite–Simpson method is of order 4 for optimal control in the states and Lagrange multipliers, so that

$$\begin{aligned} \max_{k=0,\ldots N} ||\mathbf{x}_k - \mathbf{x}^*(t_k)|| &\leq ch^\kappa, \\ \max_{k=1,\ldots N} ||\lambda_k - \lambda^*(t_k)|| &\leq ch^\kappa. \end{aligned} \tag{2.56}$$

where $h = \max_k h_k$, $c > 0$, $\mathbf{x}^*$ is the continuous state in the solution of Problem 2 and λ^* is the adjoint variable in the solution to Problem 2, $\mathbf{x}_k$ is the discrete state in the NLP solution of the discretised Problem 2 and λ_k is the discrete Lagrange multiplier in the NLP solution of the discretised Problem 2, $\kappa = 2$ in the uncompressed trapezoidal case, and $\kappa = 4$ in the uncompressed Hermite–Simpson case.

In the compressed trapezoidal case, it has been proved in the case of a uniform grid with interval length h that the convergence of the state is also of order 2, but the convergence of the Lagrange multiplier is only order 2 at the midpoints but linear (order 1) at the grid points [10]:

$$\begin{aligned}\max_{k=1,\ldots N} ||\lambda_k - \lambda^*(t_k)|| &\le ch \\ \max_{k=1,\ldots N} ||\lambda_k - \lambda^*(t_k - h/2)|| &\le ch^2. \end{aligned} \tag{2.57}$$

With regard to the convergence of the controls, it has been proved that with the compressed trapezoidal discretisation method, the controls converge with order 2 in the inside grid points, these are all grid points except t_0 and t_f [10]:

$$\max_{k=1,\ldots N-1} ||\mathbf{u}_k - \mathbf{u}^*(t_k)|| \le ch^2. \tag{2.58}$$

2.4.2 Example: Simple Problem with Analytical Solution

This example from [12] serves to illustrate how a basic direct collocation method can be implemented to solve a simple problem. Consider the problem of finding $u(t)$, $t \in [0, 1]$ to minimise the cost

$$J = x_2(1) \tag{2.59}$$

subject to the dynamic constraints

$$\begin{aligned}\dot{x}_1 &= 0.5x_1 + u, \\ \dot{x}_2 &= u^2 + xu + \frac{5}{4}x^2, \end{aligned} \tag{2.60}$$

the boundary conditions:

$$\begin{aligned}x_1(0) &= 1, \\ x_2(0) &= 0, \end{aligned} \tag{2.61}$$

and the following bounds:

$$\begin{aligned} -10 \le u(t) & \le 10, \\ 10 \le x_1(t) & \le 10, \\ -10 \le x_2(t) & \le 10. \end{aligned} \tag{2.62}$$

The analytical solution of the problem is as follows:

$$\begin{aligned} x_1(t) &= \frac{\cosh(1-t),}{\cosh(1)} \\ u(t) &= -\frac{(\tanh(1-t)+0.5)\cosh(1-t)}{\cosh(1)}. \end{aligned} \tag{2.63}$$

The code shown below uses Matlab [18] commands together with a NLP function from Matlab's Optimization Toolbox [19]. The following Matlab code is used to define the right-hand side of the state equations:

```
function  [ f ] =  ex1derivs( x, u )
x1 = x(1);
x2 = x(2);
dx1 = 0.5*x1 + u;
dx2 =  u^2 + x1*u + 5.0/4.0*x1^2;
f = [dx1;dx2];
end
```

The following Matlab code is used to define the objective function. The decision vector is $y = [u(0), \ldots, u(N), x_1(0), \ldots, x_1(N), x_2(0), \ldots, x_2(N)]^T$.

```
function J = ex1obj( y, N )
N1 = N+1;
x2=y(2*N1+1:3*N1);
J = x2(end);
end
```

The following Matlab code is used to define the constraints of the NLP problem. Note that only equality constraints associated with the differential defect constraints and with the event constraints are implemented here. The discretisation employed is the compressed trapezoidal method.

```
function [g, geq] = ex1con( y, N )
t0=0; % Initial time
tf=1.0; % Final time
N1 = N+1; % Number of grid points
h = (tf-t0)/N; % Integration step
u = y(1:N1); % Control input
% States
x1=y(N1+1:2*N1);
x2=y(2*N1+1:3*N1);
% Differential defect constraints...
Z = [];
for k=1:N1-1,
 xk  = [x1(k), x2(k)]';
 xk1 = [x1(k+1),x2(k+1)]';
 fk  = ex1derivs(xk,u(k));
 fk1 = ex1derivs(xk1,u(k+1) );
 z = xk1 - xk - h/2*( fk + fk1);
 Z = [Z; z];
end;
% Event constraints ...
E = zeros(2,1);
E(1) = x1(1)- 1.0;
E(2) = x2(1)- 0.0;
geq = [Z; E];
g = [];
```

The following Matlab code is used to define an initial guess for the solution of the problem, to establish the bounds on the decision variables, and to call Matlab's SQP routine **fmincon** to search for numerical solution to the discretised problem for given tolerances and termination parameters. Note that **fmincon** is allowed to compute numerically the derivatives required for the solution of the NLP problem, which are assumed to be dense.

```
function ex1(N)
% N     = Number of intervals.
N1      = N+1; %Number of grid points.
tf      = 1.0; % Final time
% Initial guesses for states
x1_guess = linspace( 1.0, 0.0, N1 );
x2_guess = linspace( 0.0, 0.0, N1 );
% Initial guess for control
u_guess = zeros(N1,1);
% Initial guess for decision vector
y_guess = [ u_guess(:); x1_guess(:); x2_guess(:)];
x1_min = -10.0*ones(N1,1); x1_max = 10.0*ones(N1,1);
x2_min = -10.0*ones(N1,1); x2_max = 10.0*ones(N1,1);
u_min = -10.0*ones(N1,1);
u_max =  10.0*ones(N1,1);
ymin= [ u_min; x1_min; x2_min];
ymax= [ u_max; x1_max; x2_max];
options=optimset('LargeScale','off','Display', 'iter',...
'TolX',1e-5,'TolFun',1e-5,'TolCon',1e-5,'MaxIter',2000, ...
'MaxFunEvals',100000);
y = fmincon('ex1obj', y_guess, [],[],[],[], ymin,ymax,...
'ex1con',options, N );
```

Figure 2.3 shows a comparison between state and control histories obtained with the trapezoidal method and $N = 10$ equally spaced intervals, and the corresponding analytical result. The maximum absolute error in x_1 is 0.014353, while it is 0.064556 in the control u. The resulting objective function value was $J = 0.764773$, while its analytical value computed up to ten significant figures is 0.7615941557, giving an absolute error of 0.0031789 in the objective function.

While the above code is useful to illustrate the basic concepts and to solve this simple example, the code misses various key ingredients which are needed to make a robust general purpose implementation that may be employed to solve industrial scale problems. These ingredients include:

- Incorporation of other variables in the decision vector (such as initial and final time)
- Scaling of variables and constraints
- Efficient sparse differentiation
- Use of sparse NLP
- Alternative discretisations
- Handling other types of constraints
- Ability to handle multi-phase problems
- Calculation of measures of accuracy of the solution
- Automatic mesh refinement

Some of these key aspects are discussed in subsequent sections of this chapter.

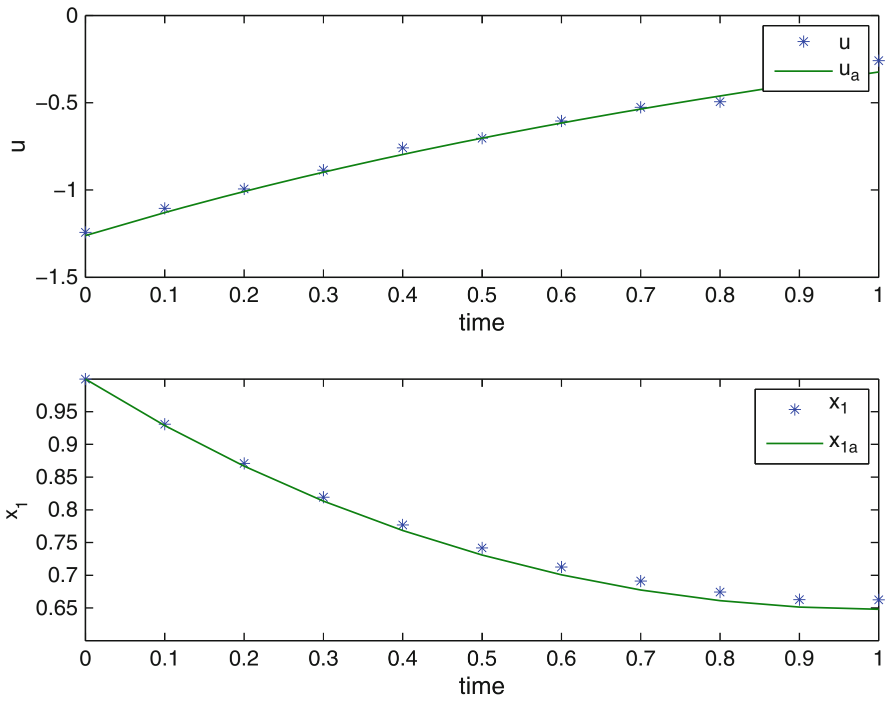


Fig. 2.3 Comparison of state and control histories obtained with the trapezoidal in example 1 using $N = 10$ intervals with the corresponding analytical result

2.5 Practical Aspects

It is worth discussing the following aspects associated to the practical use of direct collocation methods for computational optimal control.

2.5.1 Scaling

The numerical methods employed by NLP solvers are sensitive to the scaling of variables and constraints. Scaling may change the convergence rate of an algorithm, the termination tests, and the numerical conditioning of systems of equations that need to be solved.

Modern optimal control codes often perform automatic scaling of variables and constraints, while still allowing the user to manually specify some or all of the scaling factors. Scaling factors for controls, states, static parameters, and time, are typically computed on the basis of the user-supplied bounds for these variables. For finite bounds, the variables are scaled (and possibly translated) so that the translated and scaled values are within a given range, for example, $[-1, 1]$.

The scaling factor of each differential defect constraint is often taken to be equal to the scaling factor of the corresponding state [5], which helps to improve the conditioning of the Jacobian of the constraints. Scaling factors for all other constraints are often computed as follows. The scaling factor for the *i*th constraint of the NLP problem is the reciprocal of the 2-norm of the *i*th row of the Jacobian of the constraints [5]. This is done so that the rows of the Jacobian of the constraints have a 2-norm equal to one. The scaling factor for the objective function is often computed as the reciprocal of the 2-norm of the gradient of the objective function evaluated at the initial guess.

2.5.2 Sparse Nonlinear Programming

Most realistic optimal control problems result, after discretisation, in large and sparse NLP problems. "Large" means, in this context, that the underlying linear systems of equations cannot be handled efficiently or accurately by NLP implementations that use dense, direct matrix factorisations. "Sparse" means that the Jacobian of the constraints and the Hessian of the Lagrangian are sparse matrices, that is matrices with few non-zero elements compared to the total number of elements. NLP solvers need to use linear equation solvers in their iterations. Sparse NLP solvers employ sparse linear solvers. It is normally not possible to solve large sparse NLP problems using dense NLP solvers. Hence it is extremely important for an optimal control solver that the underlying NLP solver is able to solve large and sparse problems.

There are different types of large sparse NLP solvers. For instance, some implementations such as SNOPT use sparse constraint Jacobian and a reduced (dense) Hessian [11]. On the other hand, other NLP methods use a sparse Hessian [20]. Some NLP solvers use the interior-point paradigm, which employs barrier functions as part of the strategy to enforce the satisfaction of constraints [21]. Other methods, such as sequential quadratic programming (SQP) [11], use an active set strategy together with a penalty-based approach for the same purpose. It is very difficult to decide in general which method provides better results for optimal control problems, but there are indications [5] that sparse SQP methods often provide better results than interior-point methods, while methods using the sparse Hessian usually provide better efficiency and convergence than those using a reduced Hessian. It is a good idea to provide optimal control packages with options to use different sparse NLP solvers.

2.5.3 Efficient Sparse Differentiation

Modern optimal control software uses sparse finite differences [8] or sparse automatic differentiation routines to compute the derivatives needed by the NLP solver. The use of sparse differentiation is closely tied with the use of sparse NLP implementations.

Sparse finite differences exploit the structure of the Jacobian and Hessian matrices associated with an optimal control problem, such that the individual elements of these matrices can be computed very efficiently by perturbing groups of variables, as opposed to perturbing single variables.

Automatic differentiation[1] is based on the application of the chain rule to obtain derivatives of a function given as a numerical computer program. Automatic differentiation exploits the fact that computer programs execute a sequence of elementary arithmetic operations such as additions or elementary functions such as **sine()**. By applying the chain rule of differentiation repeatedly to these operations, derivatives of arbitrary order can be computed automatically and very accurately. Sparsity can also be exploited with automatic differentiation for increased efficiency.

Optimal control analysts are encouraged to use, whenever possible, sparse automatic differentiation to compute the required derivatives, as they are more accurate and faster to compute than numerical derivatives and they are free of truncation errors. The calculation of dense derivatives in automatic or numerical form is not advisable when solving optimal control problems by direct collocation methods.

When computing numerical derivatives, the structure may be exploited further by separating constant and variable parts of the derivative matrices. The constant part of the derivative is calculated once, and only its variable part is computed by sparse finite differences.

It is worth noting that the achievable sparsity is dependent on the discretisation method employed and on the problem itself [6].

2.5.4 Measures of Accuracy of the Discretisation

It is important for optimal control codes implementing direct collocation methods to estimate the discretisation error. This can be reported back to the user and employed as the basis of a mesh refinement scheme [5]. Define the error in the differential equation as a function of time:

$$\varepsilon(t) = \dot{\tilde{\mathbf{x}}}(t) - \mathbf{f}[\tilde{\mathbf{x}}(t), \tilde{\mathbf{u}}(t), \mathbf{p}, t],$$

where $\tilde{\mathbf{x}}$ is an interpolated value of the state vector given the grid-point values of the state vector, $\tilde{\mathbf{x}}$ is an estimate of the derivative of the state vector given the state vector interpolant, and $\tilde{\mathbf{u}}$ is an interpolated value of the control vector given the grid-point values of the control vector. The type of interpolation used depends on the collocation method employed. The absolute local error corresponding to state i on a particular interval $t \in [t_k, t_{k+1}]$, is defined as follows:

$$\eta_{i,k} = \int_{t_k}^{t_{k+1}} |\varepsilon_i(t)| dt,$$

[1]The following web site has a good deal of information about automatic differentiation: http://www.autodiff.org/.

where the integral is computed using an accurate quadrature method, such as the composite Simpson method. The local relative ODE error is defined as:

$$\varepsilon_k = \max_i \frac{\eta_{i,k}}{w_i + 1},$$

where

$$w_i = \max_{k=1}^{N} \left[|\tilde{x}_{i,k}|, |\dot{\tilde{x}}_{i,k}| \right].$$

The error sequence ε_k, or a global measure of the size of the error (such as the maximum of the sequence ε_k) can be analysed by the user to assess the quality of the discretisation. This information may also be useful to aid the mesh refinement process, as discussed below.

2.5.5 Mesh Refinement

Given that the solution of optimal control problems may involve periods of time where the state and control variables are changing rapidly, it makes sense to use shorter discretisation intervals in such periods, while longer intervals are often sufficient when the variables are not changing much. This is the main idea behind automatic mesh refinement methods, which detect when a solution needs shorter intervals in regions of the domain by monitoring the accuracy of the discretisation over the whole domain of the problem. Based on this observation, the solution grid is refined aiming to improve the discretisation accuracy in regions of the domain that require it. The NLP problem is then solved again over the new mesh, using the previous solution as an initial guess (which is interpolated over the new mesh), and the result is re-evaluated. Typically several mesh refinement iterations are performed until a given overall accuracy requirement, such as a specified upper bound on the discretisation error ($\max_k \varepsilon_k < \varepsilon_{\max}$), is satisfied.

It is common to start with a low-order discretisation (such as the trapezoidal method) over a uniform coarse grid and then switch to a more accurate method (such as Hermite–Simpson) after a few mesh refinement iterations. The mesh is typically refined by adding a limited number of nodes to intervals that require more accuracy, so effectively subdividing existing intervals. A well-known algorithm for mesh refinement using local discretisations was proposed by Betts [5].

2.5.6 Multi-phase Problems

Some optimal control problems can be conveniently formulated as having a number of phases. Phases may be inherent to the problem (e.g. a spacecraft drops a section and enters a new phase). Phases may also be introduced by the analyst to allow for peculiarities in the solution of the problem, such as discontinuities in the control variables, or constrained arcs (i.e. segments of time where state inequality constraints become active). Many real world problems need a multi-phase approach for their solution. Typically, the formulation for multi-phase problems involves the following elements:

1. An objective function which is the sum of individual objective functions for each phase
2. Differential equations and path constraints for each phase, noting that the dynamics need not be the same in all phases
3. Phase linkage constraints which relate the states and possibly the control variables at the boundaries between the phases
4. Event constraints which the initial and final conditions of each phase need to satisfy

2.5.7 Potential Pitfalls

There are some situations which can cause problems with direct collocation methods. These include the following:

1. *Singular arcs*. In some optimal control problems, extremal arcs satisfying Eq. (2.13) occur where the matrix $\partial^2 H/\partial \mathbf{u}^2$ is singular. These are called singular arcs. The presence of singular arcs may cause difficulties to computational optimal control methods to find accurate solutions if the appropriate conditions are not enforced a priori [5]. See [7, 17] for further details on the handling of singular arcs.
2. *Discontinuous control*. Although direct methods with local mesh refinement tend to concentrate nodes in regions where there are control discontinuities in the solution, the accuracy of the solution may be reduced in some cases if the discontinuity is not handled appropriately, for instance, by separating the problem into phases.
3. *State dependent path constraints*. This type of constraint may lead to the dynamics being a high-index differential-algebraic system, which will result in Jacobian rank deficiency and difficulties with the NLP solver, if not handled appropriately. These problems may require special treatment by differentiating the constraint function until the control input appears explicitly, and then including the resulting set of constraints into the problem. In the case of inequality constraints, additional conditions are required at the boundaries of a constrained arc. See [7, 5] for further details on handling state-dependent constraints.

2.6 Example: Space Vehicle Launch Problem

This problem consists of the launch of a space vehicle. See [16, 2] for a full description of the problem. The problem illustrates some of the aspects that are common in complex optimal control problems, including the presence of path constraints, multiple phases, free final time, and terminal constraints. The flight of the vehicle can be divided into four phases, with dry masses ejected from the vehicle at the end of phases 1, 2, and 3. The final times of phases 1, 2, and 3 are fixed, while the final time of phase 4 is free. The optimal control problem is to find the control, **u**, that minimises the cost function

$$J = -m^{(4)}(t_f). \tag{2.64}$$

In other words, it is desired to maximise the vehicle mass at the end of phase 4. Using the Earth-centred-inertial (ECI) reference, the dynamics are given by:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v}, \\ \dot{\mathbf{v}} &= -\frac{\mu}{\|\mathbf{r}\|^3}\mathbf{r} + \frac{T}{m}\mathbf{u} + \frac{\mathbf{D}}{m}, \\ \dot{m} &= -\frac{T}{g_0 I_{sp}}, \end{aligned} \tag{2.65}$$

where $\mathbf{r}(t) = \left[x(t)\ y(t)\ z(t)\right]^T$ is the position, $\mathbf{v} = \left[v_x(t)\ v_y(t)\ v_z(t)\right]^T$is the Cartesian ECI velocity, μ is the gravitational parameter, T is the vacuum thrust, m is the mass, g_0 is the acceleration due to gravity at sea level, I_{sp} is the specific impulse of the engine, $\mathbf{u} = \left[u_x\ u_y\ u_z\right]^T$ is the thrust direction, and $\mathbf{D} = \left[D_x\ D_y\ D_z\right]^T$ is the drag force, which is given by

$$\mathbf{D} = -\frac{1}{2}C_D A_{ref}\rho\|\mathbf{v}_{rel},\|\mathbf{v}_{rel}, \tag{2.66}$$

where C_{D} is the drag coefficient, A_{ref} is the reference area, ρ is the atmospheric density, and $\mathbf{v}_{rel}$ is the Earth relative velocity, where $\mathbf{v}_{rel}$ is given as

$$\mathbf{v}_{rel} = \mathbf{v} - \omega \times \mathbf{r}, \tag{2.67}$$

where ω is the angular velocity of the Earth relative to inertial space. The atmospheric density is modelled as follows:

$$\rho = \rho_0 \exp[-h/h_0], \tag{2.68}$$

where ρ_0 is the atmospheric density at sea level, $h = \|\mathbf{r}\| - R_e$ is the altitude, R_e is the equatorial radius of the Earth, and h_0 is the density scale height.

The vehicle starts on the ground at rest (relative to the Earth) at time t_0, so that the initial conditions are

$$\begin{aligned} \mathbf{r}(t_0) &= \mathbf{r}_0 = \left[5,605.2\ 0\ 3,043.4\right]^T km, \\ \mathbf{v}(t_0) &= \mathbf{v}_0 = \left[0\ 0.4076\ 0\right]^T km/s, \\ m(t_0) &= m_0 = 301,454\,kg. \end{aligned} \tag{2.69}$$

The terminal constraints define the target transfer orbit, which is defined in orbital elements as

$$\begin{aligned} a_f &= 24,361.14\,km, \\ e_f &= 0.7308, \\ i_f &= 28.5\,^\circ, \\ \Omega_f &= 269.8\,^\circ, \\ \omega_f &= 130.5\,^\circ. \end{aligned} \tag{2.70}$$

There is also a path constraint associated with this problem:

$$||\mathbf{u}||^2 = 1. \tag{2.71}$$

The following linkage constraints force the position and velocity to be continuous and also account for discontinuity in the mass state due to the ejections at the end of phases 1, 2, and 3:

$$\begin{aligned} \mathbf{r}^{(p)}(t_f) &\;-\mathbf{r}^{(p+1)}(t_0) = \mathbf{0}, \\ \mathbf{v}^{(p)}(t_f) &\;-\mathbf{v}^{(p+1)}(t_0) = \mathbf{0}, \qquad (p = 1,\ldots,3), \\ m^{(p)}(t_f) &\;-m^{(p)}_{dry} - m^{(p+1)}(t_0) = 0, \end{aligned} \tag{2.72}$$

where the superscript (p) represents the phase number, and $m^{(p)}_{dry}$ represents the ejected mass at the end of phase p.

Figure 2.4 shows the launch altitude, Fig. 2.5 shows the speed of the vehicle as a function of time and Fig. 2.6 shows the control variables as a function of time.

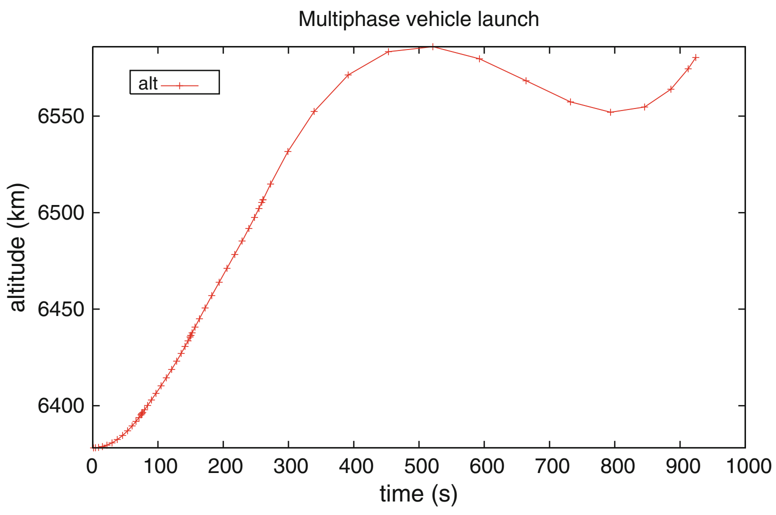


Fig. 2.4 Altitude for the vehicle launch problem

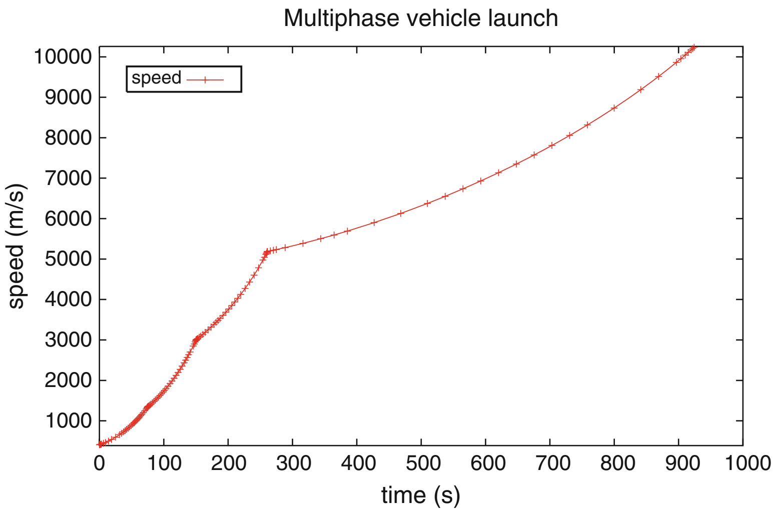


Fig. 2.5 Speed for the vehicle launch problem

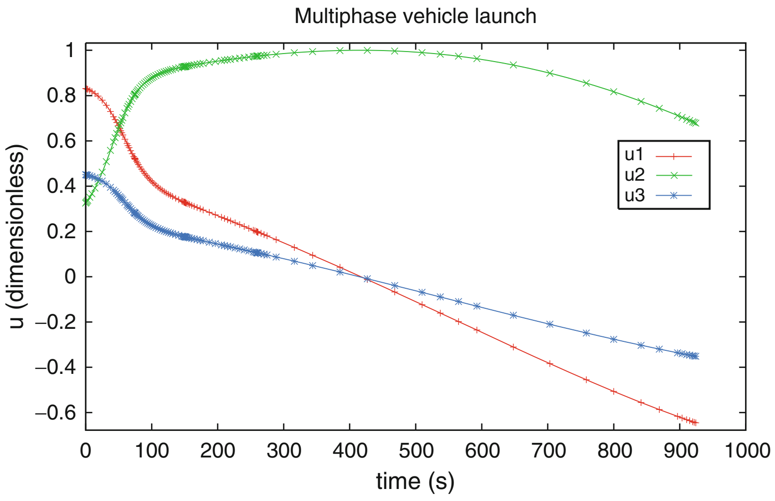


Fig. 2.6 Controls for the vehicle launch problem

This problem has been solved using the author's open source optimal control software PSOPT [1]. Note that the C++ source code associated with this problem is available for free[2]. The numerical values for the physical constants associated with the model can be found in the code. Table 2.1 summarises statistics from the

[2]See http://www.psopt.org.

Table 2.1 Mesh refinement statistics: multiphase vehicle launch

Iter	DM	M	NV	NC	OE	CE	JE	HE	RHS	$\varepsilon_{\max}$	$\mathrm{CPU_a}$
1	TRP	20	208	200	83	716	47	0	25,776	1.264e − 02	3.000e − 01
2	TRP	28	288	264	37	38	29	0	1,976	8.429e − 03	2.100e − 01
3	H-S	36	464	360	44	45	40	0	4,500	3.594e − 03	4.800e − 01
4	H-S	48	620	468	45	46	39	0	6,256	8.749e − 04	5.900e − 01
5	H-S	64	828	612	40	41	40	0	7,544	2.002e − 04	7.300e − 01
6	H-S	88	1,140	828	78	79	74	0	20,224	5.133e − 05	1.760e + 00
7	H-S	114	1,478	1,062	65	66	61	0	22,044	1.928e − 05	1.840e + 00
8	H-S	132	1,712	1,224	62	63	62	0	24,444	9.888e − 06	2.150e + 00
$\mathrm{CPU_b}$	–	–	–	–	–	–	–	–	–	–	2.419e + 01
–	–	–	–	–	454	1,094	392	0	112,764	–	3.225e + 01

Key: Iter = iteration number, DM = discretisation method, M = number of nodes, NV = number of variables, NC = number of constraints, OE = objective evaluations, CE = constraint evaluations, JE = Jacobian evaluations, HE = Hessian evaluations, RHS = ODE right-hand side evaluations, $\varepsilon_{\max}$ = maximum relative ODE error, $\mathrm{CPU_a}$ = CPU time in seconds spent by NLP algorithm, $\mathrm{CPU_b}$ = additional CPU time in seconds spent by PSOPT

mesh refinement process. The problem was solved in eight mesh refinement iterations until the maximum relative ODE error $\varepsilon_{\max}$ was below 1×10^{-5}, starting from a coarse uniform grid and resulting in a finer non-uniform grid.

References

1. Becerra, V.: Solving optimal control problems at no cost with PSOPT. Proceedings of IEEE Multi-conference on Systems and Control, Yokohama, Japan, September 7–10 (2010)
2. Benson, D.A.: A Gauss pseudospectral transcription for optimal control. Ph.D. thesis, Department of Aeronautics and Astronautics, MIT, Cambridge, MA (2004)
3. Bertsekas, D.P., Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, MA (1999)
4. Betts, J.T.: Practical Methods for Optimal Control Using Nonlinear Programming. SIAM, Philadelphia (2001)
5. Betts, J.T.: Practical Methods for Optimal Control and Estimation Using Nonlinear Programming. SIAM, Philadelphia (2010)
6. Betts, J.T., Erb, S.O.: Optimal low thrust trajectories to the moon. SIAM J. Appl. Dyn. Syst. **2**, 144–170 (2003)
7. Bryson, A., Ho, Y.C.: Applied Optimal Control. Halsted Press, Sydney (1975)
8. Curtis, A.R., Powell, M.J.D., Reid, J.K.: On the estimation of sparse Jacobian matrices. J. Inst. Math. Appl. **13**, 117–120 (1974)
9. Elnagar, G., Kazemi, M.A., Razzaghi, M.: The pseudospectral legendre method for discretizing optimal control problems. IEEE Trans. Automat. Control **40**, 1793–1796 (1995)
10. Engelsone, A., Campbell, S.: Adjoint estimation using direct transcription multipliers: Compressed trapezoidal method. Optim. Eng. **9**, 291–305 (2008)
11. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2001)
12. Hager, W.: Runge–kutta methods in optimal control and the transformed adjoint system. Numer. Math. **87**, 247–282 (2000)
13. Hairer, E., Norsett, S., Wanner, G.: Solving Ordinary Differential Equations I: Nonstiff Problems. Springer, Berlin (2002)

14. Hartl, R.F., Sethi, S.P., Vickson, R.G.: A survey of the maximum principles for optimal control problems with state constraints. SIAM Rev. **37**(2), 181–218 (1995). URL http://www.jstor.org/stable/2132823
15. Luenberger, D.: Optimization by Vector Space Methods. Wiley, New York (1997)
16. Rao, A., Benson, D., Huntington, G., Francolin, C.: User's manual for GPOPS version 1.3: A Matlab package for dynamic optimization using the Gauss pseudospectral method (2008)
17. Sethi, S., Thompson, G.: Optimal Control Theory: Applications to Management Science and Economics. Kluwer, Dordecht (2000)
18. The Mathworks: Matlab Programming Fundamentals. Natick, MA (2012)
19. The Mathworks: Optimisation Toolbox User's Guide. Natick, MA (2012)
20. Vanderbei, R.J., Shanno, D.: An interior-point method for nonconvex nonlinear programming. Comput. Optim. Appl. **13**, 231–252 (1999)
21. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**, 25–57 (2006)

Chapter 3
Formation Flying Control for Satellites: Anti-windup Based Approach

Josep Boada, Christophe Prieur, Sophie Tarbouriech, Christelle Pittet, and Catherine Charbonnel

Abstract Control theory has significantly evolved in the field of the nonlinear control. However, the methods used in the aerospace industry lie usually on linear techniques applied to linearized models. The increasing requirements in terms of operational reliability and performance ask for the development of new control techniques more complex in order to meet the new demands. Therefore, the industry is moving to the modern control theory looking for new nonlinear approaches. In particular, actuators saturation represents a nonlinear phenomenon common in almost all physical applications. This can then lead to performance degradation, limit cycle appearance, non-desired equilibrium conditions, and even system instability. The objective of this chapter is to adapt and develop the anti-windup compensator design to the control with high precision for the angular and

J. Boada
Albatros Aeronautics, Victoria, Spain
e-mail: josep.boada@albaero.com

C. Prieur (✉)
Gipsa-lab, Department of Automatic Control, Grenoble Campus, 11 rue des Mathématiques, 4638402 Saint Martin d'Hères Cedex, France
e-mail: christophe.prieur@gipsa-lab.fr

S. Tarbouriech
CNRS, LAAS, 7 avenue du Colonel Roche, F-31400 Toulouse, France

Univ de Toulouse, LAAS, F-31400 Toulouse, France
e-mail: tarbour@laas.fr

C. Pittet
Centre National d'Etudes Spatiales (CNES), Toulouse, France
e-mail: christelle.pittet@cnes.fr

C. Charbonnel
Thales Alenia Space, Cannes, France
e-mail: catherine.charbonnel@thalesaleniaspace.com

the linear axes of a satellite. In the aerospace application field, this situation meets with the drag-free or the formation flying missions. These missions use high-precision thrusters as actuators whose capacity appears to be critically low. Moreover, thrusters have a particular modeling. Allocation functions adapted to the anti-windup design are then explored. In addition considering the current state of the art of the anti-windup design, there is a strong necessity of using symmetrizing techniques for the saturation. The main objective of this work consists in applying the developed tools on an aerospace study case. As an example, a complete methodology is proposed to control a formation flying mission controlling both attitude and relative position.

Keywords Saturating thrusters • Anti-windup design • Control • Optimization

3.1 Introduction

Formation flying control problem has been an important field of research since the 1990s. Several possible applications in the space exploration domain make this field very interesting [1, 15, 24]. In these kinds of missions, one seeks to control the formation with a fine precision in both attitude and relative position. Consequently, the actuator is based on a precise propulsive system. However, these kinds of actuators have a limited capacity which cannot be exceeded.

The control limitation due to the constraints of the actuators' maximum capacity represents a nonlinear phenomenon common in almost all physical applications. Traditionally, a classical solution, at least in industry, consists in imposing important margins in order to prevent the actuators from reaching their maximum capacity, that is, to avoid saturation. In that manner, one tries to ensure the linearity of the system. However, this a posteriori validation is insufficient because, non-nominal disturbances, transitions between operational modes, and the presence of system failures can force the actuators to reach their limits. Actuator saturation can then lead to performance degradation, limit cycle appearance, non-desired equilibrium conditions, and even system instability [2, 18, 19, 26, 27].

The nonlinear techniques dealing with saturation can be classified in two main research lines. The first one seeks to introduce the nonlinear saturation in the synthesis process of the controller. The second one introduces an extra layer to the existing linear controller, accounting for the nonlinearities. This strategy, also called anti-windup design, allows the designer to keep the existing linear controller (already validated) and only to introduce a compensator which is only active when the nonlinearity arises [14, 17, 20, 29, 31, 33]. The design of such a compensator is generally cast in a static optimization problem of the controller parameters. Accompanying the development of semi-definite programming and convex optimization [10], the synthesis problem can be formulated as the optimization of a multi-objective criterion (corresponding to closed-loop stability and performance specifications) subject to matrix inequalities constraints associated to the dynamical system.

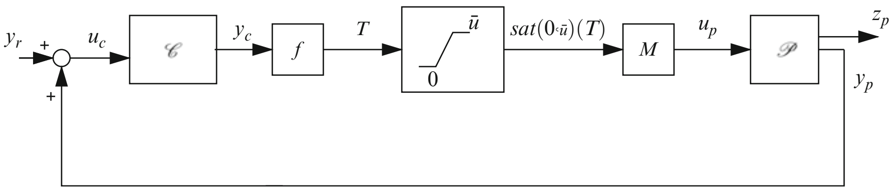


Fig. 3.1 Control loop block diagram

The objective of this chapter is to depict how the anti-windup compensator design can be adapted to high-precision control of the linear and angular displacements of a satellite. Consequently, in formation flying control problem, the introduction of the anti-windup becomes an interesting technique to ensure the mission requirements and its reliability. Moreover, let us point out that space missions involving thrusters as actuators are modeled in a particular way. The control variables of the physical systems and the action provided by thrusters are not the same. An allocation function is included in the modeling of the actuators to distribute the control among the actuators [4, 11, 22]. In the classical linear approach, this function can be omitted; however, when the saturation of the actuators arises, its behavior has to be considered in the control design [4]. Allocation functions adapted to the anti-windup design have to be explored. Moreover the presence of thrusters introduces a peculiar formulation of the saturation function. The saturation presents asymmetric bounds with the minimum equal to zero. Considering the current state of the art of the anti-windup design, there is a strong necessity of using symmetrizing techniques for the saturation. The symmetrizing procedure and the allocation function defined are put along with the anti-windup design to be applied in a spacecraft mission configuration.

For anti-windup study purposes, the formation flying problem can be described by a block diagram as presented in Fig. 3.1. If y_p appears as attitude and/or relative position, the control loop would be illustrative of a formation flight configuration.

Remark 1. $y_r = 0$ for consistency reasons without loss of generality.

In this chapter a formation flying control problem is modeled and solved. This is the relative position of a two-satellite formation. We prefer to focus on this problem to ease the presentation of the anti-windup methods. However, this reduction is not a necessary simplification, and a more complete multi-variable system with a coupling of the attitude and the relative position of two satellites could be tackled. For more details on this last case, see [6], where the saturation effects are studied and some anti-windup methods are developed. Let us emphasize that the anti-windup design problem considered corresponds to a multiobjective control optimization problem, which consists in minimizing some performance criteria (closed-loop $\mathscr{L}_2$-gain, fuel consumption) and maximizing the size (via the choice of optimal criteria) of the state space region of safe operation (guaranteed stability around the origin, attainability of the performance level), in presence of two hard nonlinear phenomena as the presence of both actuator saturation and allocation function.

Thus, this kind of mathematical programming approach can be viewed as complementary to on-line optimization-based methods, which require in general a high computational effort.

The chapter is organized as follows. First, in Sect. 3.2, the model of the relative position between two satellites is presented. Only the control of an axis is considered. Then, in Sect. 3.3, the following anti-windup techniques are applied to the relative position control: the Direct Linear Anti-Windup (DLAW), the Model Recovery Anti-Windup (MRAW), and the Extended Model Recovery Anti-windup (EMRAW). Simulations illustrate the benefits provided by the anti-windup compensators. Some concluding remarks end the chapter.

3.2 Relative Position Control

3.2.1 Relative Position Plant Model

The first relative dynamics to be described is the relative position between two satellites along the z-axis. Let us consider two satellites and two frames fixed to each satellite. $\mathscr{F}_{\text{sat1}}$ is the first satellite associated frame and $\mathscr{F}_{\text{sat2}}$ the second satellite associated frame. From the third theorem of the rigid body dynamics, and assuming that the satellite is a point-mass system, the acceleration of a rigid body is proportional to the sum of external forces:

$$\ddot{z}_i = m_i^{-1} \sum(F_i), \tag{3.1}$$

where z_i is the displacement on the z-axis of satellite i. Hence, $\ddot{z}_i$ denotes the acceleration on this axis. m_i denotes the mass of satellite i, and $\sum(F_i)$ stands for the sum of external forces on satellite i.

The control objective is to cancel the lateral position error on the z-coordinate between the satellites (see Fig. 3.2). Therefore, the relative dynamics can be described, applying (3.1) to the difference of the z_i coordinate with $i = 1, 2$, that is, $\Delta z = z_1 - z_2$. The control objective is then to satisfy $\Delta z = 0$. Denoting $\sum(F_i)$ by F_i for consistency reasons, it yields $\Delta\ddot{z} = -m_2^{-1}F_2 + m_1^{-1}F_1$.

The state space representation reads

$$\begin{cases} \dot{x}_p = A_p x_p + B_p u_p = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ m_1^{-1} & -m_2^{-1} \end{bmatrix} u_p \\ y_p = C_p x_p = [1 \quad 0] x_p = \Delta z \\ z_p = C_z x_p = [1 \quad 0] x_p = \Delta z, \end{cases} \tag{3.2}$$

where the state variables included in the state vector x_p are the relative position (Δz) and the relative velocity ($\Delta\dot{z}$), $u_p = [F_1 \; F_2]'$ is the control input, $y_p = \Delta z$ is the measured output, and $z_p = \Delta z$ is the regulated output.

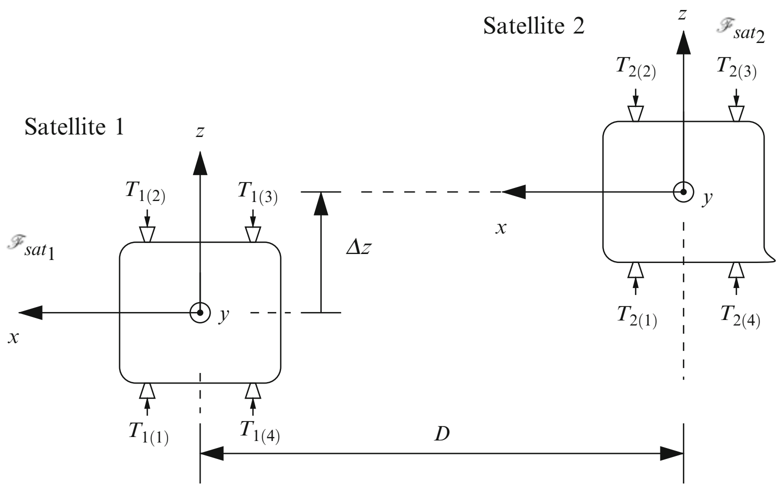


Fig. 3.2 Relative position control configuration

3.2.2 Relative Position Controller

A centralized controller is used. This means that a unique controller takes the measurements from all satellites in the formation and computes a vector y_c which contains the control output of each satellite. The controller is described through a state space representation:

$$\begin{cases} \dot{x}_c = A_c x_c + B_c u_c \\ y_c = C_c x_c + D_c u_c. \end{cases} \tag{3.3}$$

In the relative position problem, the controller is an 1-input 2-outputs (Single Input Multiple Output or SIMO) linear system with a 5 dimension state vector. The controller input is $u_c = y_p = \Delta z$ and its outputs are $y_c = [y_{c1} \quad y_{c2}]' = [F_{c1} \quad F_{c2}]'$. F_{c1} (resp. F_{c2}) stands for the controller output for the first (resp. second) satellite.

3.2.3 Relative Position Actuator Model

The satellite formation is actuated by a propulsive system composed of four proportional thrusters on each of the two satellites. To apply the required control forces (F_1 and F_2) using this propulsive system, thruster management functions have to be introduced in the control loop. These functions are composed by an allocation function that transforms the required control efforts (F_{c1}, F_{c2}) into thrusters forces, and an influence matrix that transforms the thruster outputs into forces applied on the system. Moreover, the actual forces delivered by each thruster are saturated. The general expression for the actuator is given as follows:

$$u_p = Msat_{(0,\bar{u})}(f(y_c)), \tag{3.4}$$

where M is the influence matrix for each satellite, *sat* is the thruster saturation, and f is the allocation function. Let us briefly describe all these elements separately in the next sections.

3.2.3.1 The Influence Matrix

The influence matrix describes the geometric distribution of the thrusters. The physical distribution of the thrusters is presented in Fig. 3.2. The influence matrices are then described as follows:

$$M_1 = [1 \quad -1 \quad -1 \quad 1]; \quad M_2 = [1 \quad -1 \quad -1 \quad 1]. \tag{3.5}$$

Influence matrices M_1 and M_2 are associated to satellite 1 and satellite 2, respectively.

3.2.3.2 Thruster Saturation

The saturation function is modeled by

$$sat_{(0,\bar{u})}(T_{(i)}) = \begin{cases} \bar{u}_{(i)} & \text{if } T_{(i)} > \bar{u}_{(i)} \\ T_{(i)} & \text{if } 0 \leq T_{(i)} \leq \bar{u}_{(i)} \\ 0 & \text{if } T_{(i)} < 0 \end{cases}, i = 1, \ldots, m. \tag{3.6}$$

The saturation bounds for the relative position control problem are $\underline{u} = 0$ and $\bar{u} = 1\,mN$, and $T = [T_{(1)}, \ldots, T_{(m)}]'$ is the thrust applied on each satellite.

3.2.3.3 Allocation Function

The allocation function (AF) used on real applications is a highly nonlinear optimized function. Indeed, finding an AF is a dynamic nonlinear optimization problem, which can be solved using non-linear optimization techniques [22], for example, quadratic programming. Another option is to manipulate a simple nonlinear AF based on a switching structure. This nonlinear allocation function lies on the fact that the control output y_c is treated component-wise, whereas the AF computes a set of thrust T for each component of the control output. They are denoted by $y_{c(k)}$, and the set of thrusts associated is denoted by T^k. Finally, the thrust vector applied is the sum of all T^k with $k = 1, \ldots, m_c$. The switching structure can be described by the following expression:

$$f(y_c) = \begin{cases} T^k_{(i)} = \begin{cases} 0 & \text{if } sign(M_{(k,i)}) \neq sign(y_{c(k)}) \\ \dfrac{y_{c(k)}}{\tau_{(k)} M_{(k,i)}} & \text{if } sign(M_{(k,i)}) = sign(y_{c(k)}) \end{cases} , k = 1, \ldots, m_c. \\ T_{(i)} = \sum_{k=1}^{m_c} T^k_{(i)}, i = 1, \ldots, m. \\ T = [T_{(1)} \quad \ldots \quad T_{(m)}]', \end{cases} \tag{3.7}$$

where sign$(\cdot)$ stands for the function sign and $\tau_{(k)}$ stands for the number of thrusters generating an effort of the same sign as $y_{c(k)}$. $\tau_{(k)}$ is described as follows:

$$\tau_{(k)} = \sum_{i=1}^{m} \left\{ sign(M_{(k,i)}) = sign(y_{c(k)}) \right\}, \tag{3.8}$$

where $\{\text{sign}(M_{(k,\ i)}) = \text{sign}(y_{c(k)}))\}$ is a boolean function that returns 1 if both elements are equal or 0 if they are not.

The relative position control problem presents $m = 4$ thrusters and $k = 1$ control output. Then the switching AF (3.7) has the following form for both satellites:

$$f(y_c) = \begin{cases} T_{(1)} = T_{(4)} = \begin{cases} 0 & \text{if } y_c < 0 \\ \dfrac{y_c}{2} & \text{if } y_c \geq 0 \end{cases} \\ T_{(2)} = T_{(3)} = \begin{cases} 0 & \text{if } y_c \geq 0 \\ \dfrac{y_c}{2} & \text{if} y_c < 0. \end{cases} \end{cases} \tag{3.9}$$

In that case, the actuator is modeled as follows:

$$u_p = M sat_{(0,\bar{u})}(f(y_c)) = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} sat_{(0,\bar{u})} \left(\begin{bmatrix} f(y_{c1}) \\ f(y_{c2}) \end{bmatrix} \right). \tag{3.10}$$

Another AF is based on the pseudo-inverse matrix M^* of the influence matrix M and is given by

$$T = f(y_c) = M^* y_c, \tag{3.11}$$

In this case, the control input u_p reads

$$u_p = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} sat_{(0,\bar{u})}\left(\begin{bmatrix} M_1^* y_{c1} \\ M_2^* y_{c2} \end{bmatrix}\right). \tag{3.12}$$

See also [8] where a multi-saturation-based allocation function is suggested (together with an anti-windup approach that used [28]).

3.2.4 Relative Position Closed-Loop Model

With the previously presented plant (3.2), controller (3.3) and actuator (3.10), the closed-loop system describing the relative position control reads:

$$\begin{cases} \dot{x}_p = A_p x_p + B_p u_p = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ m_1^{-1} & -m_2^{-1} \end{bmatrix} u_p \\ \dot{x}_c = A_c x_c + B_c C_p x_p \\ y_p = C_p x_p = [1 \quad 0] x_p = \Delta z \\ z_p = C_z x_p = [1 \quad 0] x_p = \Delta z \\ u_p = M sat_{(0,\bar{u})}(f(y_c)) = \begin{bmatrix} M_1 sat_{(0,\bar{u})}(f(y_{c1})) \\ M_2 sat_{(0,\bar{u})}(f(y_{c2})) \end{bmatrix} \\ y_c = C_c x_c + D_c C_p x_p \end{cases} \tag{3.13}$$

with $f(y_c)$ defined by (3.9).

System (3.13) provides a benchmark for further simulations. However, the nonlinearity introduced by the AF (3.9) makes complex the characterization of constructive conditions[1] to compute the anti-windup compensator for this system. For this reason an alternative formulation can be used based on Eq. (3.12) leading to the closed-loop system:

[1]In the sense that tractable numerical procedures can be associated to exhibit the matrices of the anti-windup compensator.

$$\begin{cases} \dot{x}_p = A_p x_p + B_p u_p = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ m_1^{-1} & -m_2^{-1} \end{bmatrix} u_p \\ \dot{x}_c = A_c x_c + B_c C_p x_p \\ y_p = C_p x_p = [1 \;\; 0] x_p = \Delta z \\ z_p = C_z x_p = [1 \;\; 0] x_p = \Delta z \\ u_p = M sat_{(0,\bar{u})}(M^* y_c) = \begin{bmatrix} M_1 sat_{(0,\bar{u})}(M_1^* y_{c1}) \\ M_2 sat_{(0,\bar{u})}(M_2^* y_{c2}) \end{bmatrix} \\ y_c = C_c x_c + D_c C_p x_p . \end{cases} \tag{3.14}$$

3.3 Anti-windup on the Relative Position Control

The relative position control problem has been modeled. Before dealing with the anti-windup computation purposes, let us consider the saturation function in (3.14). It is asymmetric. Since most of the results on anti-windup synthesis consider symmetric saturations, the saturation has to be symmetrized in order to adapt some of these results. To do that the symmetrizing technique of [7] is applied. More precisely, introducing the so-called symmetrizing vector $N\zeta_{sym} = \bar{u}/2$, consider the scheme depicted in Fig. 3.3.

Hence, we define

$$N\zeta_{var} = min\big(max_{(i)}(|f(y_c)|), N\zeta_{sym(i)}\big) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \quad i = 1, \ldots, m \tag{3.15}$$

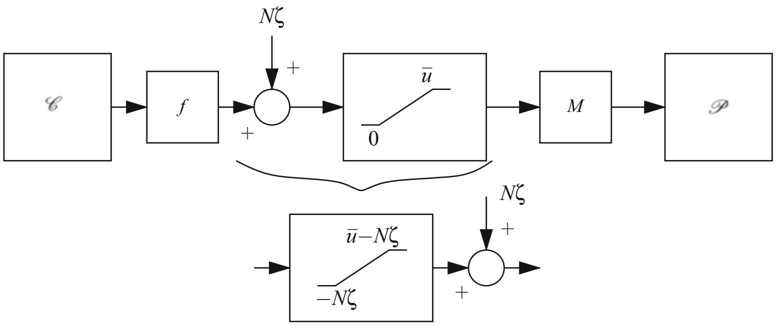


Fig. 3.3 Saturation bounds modification

which verifies that $MN\zeta_{\text{var}} = 0$. That corresponds to a variable kernel function (VKF) and solves the problems of extra consumption as exposed in [7]. Assuming the consumption to be proportional to the integral of all the thrust responses one can check that the actual consumption can be reduced by adequately defining the VKF. For more detail, the reader can consult [6]. With the saturation symmetrized, the closed-loop system (3.14) we want to study in order to design anti-windup loops can be written as:

$$\begin{cases} \dot{x}_p = A_p x_p + B_p u_p = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ m_1^{-1} & -m_2^{-1} \end{bmatrix} u_p \\ \dot{x}_c = A_c x_c + B_c y_p + v_x \\ y_p = C_p x_p = [1 \quad 0] x_p = \Delta z \\ z_p = C_z x_p = [1 \quad 0] x_p = \Delta z \\ u_p = M sat_{(u_0)}(M^* y_c) = \begin{bmatrix} M_1 sat_{(u_0)}(M_1^* y_{c1}) \\ M_2 sat_{(u_0)}(M_1^* y_{c2}) \end{bmatrix} \\ y_c = C_c x_c + D_c C_p x_p + v_y \end{cases} \tag{3.16}$$

with symmetric bounds $u_0 = \frac{1}{2}\, mN$ and where v_x and v_y are the output signals of the anti-windup compensator to be designed.

3.3.1 Anti-windup Compensator Synthesis

Three types of anti-windup compensators are investigated: the DLAW, the MRAW and the EMRAW.

All the results developed in the DLAW context are based upon the use of the dead-zone nonlinearities and associated modified sector conditions. Indeed, any system with saturation may be rewritten with dead-zone nonlinearity. Let us recall that the dead-zone function is defined by $\phi_{(u_0)}(y_c) = y_c - sat_{(u_0)}(y_c)$. In this context, the anti-windup compensator is defined as follows:

$$\mathscr{AW} \begin{cases} \dot{x}_{aw} = A_{aw} x_{aw} + B_{aw} \phi_{(u_0)}(y_c) \\ v_x = [I_{n_c} \quad 0]\Big(C_{aw} x_{aw} + D_{aw} \phi_{(u_0)}(y_c)\Big) \\ v_y = [0 \quad I_m]\Big(C_{aw} x_{aw} + D_{aw} \phi_{(u_0)}(y_c)\Big), \end{cases} \tag{3.17}$$

where $x_{aw} \in \Re^{n_{aw}}$ is the anti-windup state, $n_{\text{aw}} \geq 0$, $\phi_{(u_0)}(y_c)$, the dead-zone output is the anti-windup input, $v = [v_x' \quad v_y']' \in \Re^{n_c+m}$ is the anti-windup output and A_{aw}, B_{aw}, C_{aw},and D_{aw} are matrices of appropriate dimensions. Figure 3.4 presents the block diagram describing the DLAW structure [14, 27].

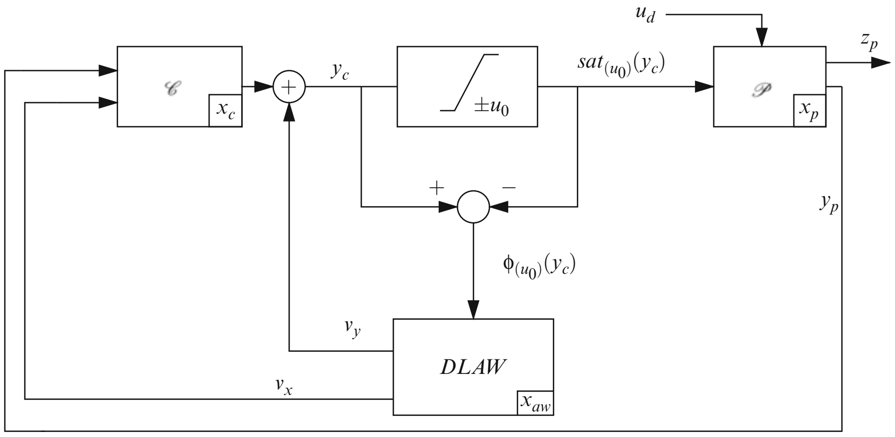


Fig. 3.4 Direct linear anti-windup structure

In the sequel, we consider by simplicity that $v_y = 0$ to avoid implicit loop, and therefore we consider a dynamic DLAW into the form

$$\mathscr{AW}\begin{cases} \dot{x}_{aw} = A_{aw}x_{aw} + B_{aw}\phi_{(u_0)}(y_c) \\ v_x = C_{aw}x_{aw} + D_{aw}\phi_{(u_0)}(y_c), \end{cases} \tag{3.18}$$

where $x_{aw} \in \Re^{n_{aw}}$ is the anti-windup state, $n_{\mathrm{aw}} \geq 0$, $v_x \in \Re^{n_c}$ is the anti-windup output, and A_{aw}, B_{aw}, C_{aw}, and D_{aw} are matrices of appropriate dimensions.

3.3.1.1 Static DLAW Synthesis

In the static case, one chooses $n_{\mathrm{aw}} = 0$, and anti-windup matrices are null except D_{aw} ($A_{\mathrm{aw}} = 0$, $B_{\mathrm{aw}} = 0$, $C_{\mathrm{aw}} = 0$). Thus, the anti-windup output is given by $v_x = D_{aw}\phi_{(u_0)}(y_c)$ and $v_y = 0$ [16].

The DLAW static gain D_{aw} can be computed by using Lyapunov theory. In particular, in order to characterize a domain of asymptotic stability for the closed-loop system (including the plant, the controller with the static anti-windup loop and the linear closed-loop system without saturation), a quadratic Lyapunov function can be used to express some conditions in terms of linear matrix inequalities (LMIs). The main advantages of using quadratic Lyapunov function resulting into LMIs reside in the fact that the synthesis of the anti-windup loop can then be carried out through the solution of convex optimization problems, which can be solved by efficient software packages [3, 21]. Hence, the stability domain can be optimized in some direction of interest, namely in the Δz direction. In this case, by using the conditions developed in [6] (see Theorem 3.1 in Chapter 3 of [6]), the static anti-windup gain D_{aw} obtained is:

$$D_{aw} = \begin{bmatrix} -0.32 & -0.32 & 0.32 & 0.32 & -1.65 & -1.65 & 1.65 & 1.65 \\ 0.28 & 0.28 & -0.28 & -0.28 & 3.5 & 3.5 & -3.5 & -3.5 \\ 2.95 & 2.95 & -2.95 & -2.95 & -6.39 & -6.39 & 6.39 & 6.39 \\ 0.88 & 0.88 & -0.88 & -0.88 & -1.81 & -1.81 & 1.81 & 1.81 \\ 376.78 & 376.78 & -376.78 & -376.78 & -665.5 & -665.5 & 665.5 & 665.5 \end{bmatrix}. \tag{3.19}$$

Regarding the structure of the static DLAW (3.19), we can observe the difference between the first four columns and the last four ones. That is due to the relation of the first four columns with the first satellite thrusters, and the relation of the last four ones with the second satellite thrusters. In addition, it is important to remark that there is a row (the fifth one) whose values outstand in comparison with the others. Actually this line affects the state of the controller related to the integration. The anti-windup mission is to attenuate the integral state of the controller which is the more sensible state to the saturation effects. Therefore, it is normal to find a more important effect on this controller state than on the others. With this anti-windup technique, we thus recover the usual method of de-charging the integral state, when a saturation occurs.

3.3.1.2 Dynamic DLAW Synthesis

Using the results in [25], the matrices A_{aw}, B_{aw}, C_{aw}, and D_{aw} of the dynamic anti-windup compensator Eq. (3.18) can be computed by solving some LMIs when we choose the order of the anti-windup compensator $n_{aw} = n_p + n_c + n_l$. For more details see Propositions 3.1 and 3.2 in Chapter 3 of [6]. Nevertheless, in some cases the order of this anti-windup compensator can be too large to be implemented in a real on-board calculator. Thus, the goal now is to find a low-order anti-windup compensator. In this case, the main difficulty resides in the choice of matrices A_{aw} and C_{aw}, the matrices B_{aw} and D_{aw} remaining linearly computationable in the conditions. A way to do this is to use the full order anti-windup compensator computation as a guide on the choice of the poles for the fixed-order anti-windup synthesis. The computation is decomposed in two steps. The first one consists in computing the full-order DLAW. From all the poles of the computed full-order compensator, only the poles of A_{aw} sharing the same magnitude order as those of the linear closed-loop system without saturation (named A_l) are retained (see also [5, 9]). The second step consists in testing the conditions to obtain B_{aw} and D_{aw}, A_{aw} and C_{aw} being fixed from the first step (see, e.g, [20, 27]).

As in the static anti-windup case, the stability domain is maximized in the relative position direction Δz. Table 3.1 shows the poles of the full-order dynamic DLAW and the poles of the linear closed-loop system A_l. The selected poles for the fixed order DLAW synthesis are marked with $*$.

Table 3.1 Full-order DLAW eigenvalues in relative position control

$\mathrm{eig}(A_{\mathrm{aw}})$($\ast \equiv$ selected)	$\mathrm{eig}(A_l)$
$-8.28 \cdot 10^6$	$(-2.61 \pm j2.88) \cdot 10^{-1}$
$-6.21 \cdot 10^6$	$(-1.62 \pm j2.02) \cdot 10^{-1}$
$-4.74 \cdot 10^4$	$(-8.23 \pm j8.23) \cdot 10^{-3}$
$-1.93 \cdot 10^2$	$-2.73 \cdot 10^{-3}$
-1.61	
$(-9.11 \pm j27.5) \cdot 10^{-2}$	
-0.13	
$-4.38 \cdot 10^{-2} (\ast)$	
$(-8.97 \pm j5.17) \cdot 10^{-3} (\ast)$	
$-8.61 \cdot 10^{-3} (\ast)$	
$-7.46 \cdot 10^{-3} (\ast)$	
$-4.11 \cdot 10^{-3}$	

3.3.1.3 MRAW and EMRAW Synthesis

The MRAW consists in selecting the anti-windup compensator as a dynamical filter, incorporating a model of the plant [14]. The aim of this filter is to recover the unconstrained closed-loop dynamics. The plant control is limited by the saturation nonlinearity thus, recovering the missing control and filtering it throughout the anti-windup compensator, we can recover the missing dynamics of the plant. This recovered dynamics allows the system to keep tracking what the closed-loop response would be in the absence of saturation. The equations describing filter dynamics are stated by

$$\begin{cases} \dot{x}_{aw} = A_p x_{aw} + B_p(sat_{u_0}(y_c + v_1) - y_c) \\ y_{aw} = C_p x_{aw} \\ v_1 = g(x_{aw}), \end{cases} \tag{3.20}$$

where $x_{aw} \in \Re^{n_p}$ is the anti-windup compensator state, $y_{\mathrm{aw}} \in \Re^q$ and $v_1 \in \Re^m$ are its outputs, and g is nonlinear function suitably designed [30, 32]. The anti-windup compensator structure is represented in Fig. 3.5. Notice that, by using the DLAW framework, the anti-windup loop is injected to the controller by considering v_x and v_y as follows:

$$\begin{aligned} v_x &= -B_c y_{aw} \\ v_y &= -D_c y_{aw} + v_1, \end{aligned} \tag{3.21}$$

where v_x and v_y are described by Fig. 3.4.

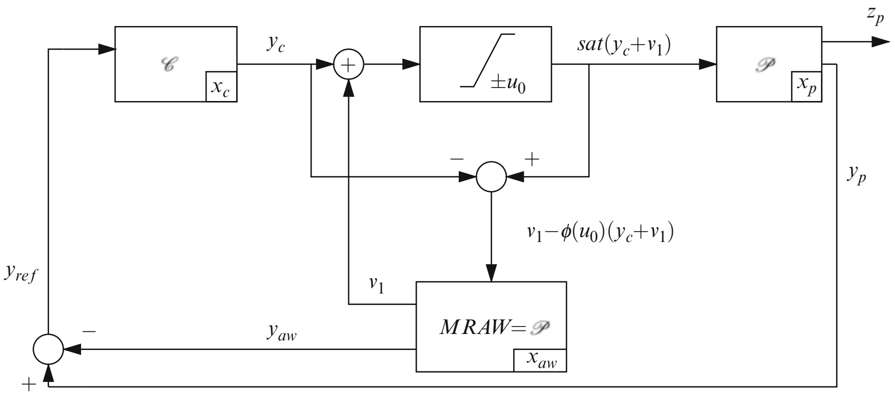


Fig. 3.5 MRAW control block diagram

A first possible solution to the MRAW problem is to select v_1 as a linear state feedback from x_{aw} designed completely disregarding the saturation effects. These solutions are associated to local stability properties but, for exponentially stable plants, the global stability is possible [13]. Another type of solutions that one can propose within the MRAW compensation is to select v_1 as a nonlinear function of the anti-windup compensator state x_{aw}. This type of solution is certainly the most difficult to design and to implement, but it is the most advanced scheme within this framework. Reference [12] gives constructive conditions to find such a stabilizing law v_1. See also [14, 27, 33], and references therein.

The EMRAW techniques is a combination of both the DLAW and MRAW structures, as depicted in Fig. 3.6.

The EMRAW strategy is then described by the following equations:

$$\mathcal{AW}: \begin{cases} \dot{x}_{aw} = (A_p x_{aw} + B_p v_1) + B_p \phi_{(u_0)}(y_c + v_1) \\ y_{aw} = C_p x_{aw} \\ v_1 = F_{aw} x_{aw} \\ v_e = E_{aw} \phi_{(u_0)}(y_c + v_1), \end{cases} \tag{3.22}$$

where $x_{aw} \in \Re^{n_{aw}}$ is the anti-windup compensator state with $n_{\text{aw}} = n_p$, $y_{\text{aw}} \in \Re^q$ and $v_1 \in \Re^m$ are the outputs generated by MRAW stage, and $v_e \in \Re^{n_c}$ is the output issued by the static DLAW stage (gain E_{aw}). Notice E_{aw} is a static DLAW as it feedbacks the dead-zone function $\phi_{(u_0)}$ directly into the controller dynamics x_c through the signal v_e. First, an objective-based algorithm, as in [5], (or a coordinate-descending algorithm as in [23]), is applied to compute some gains, and then some stability conditions are solved to compute the anti-windup loop. See [6, 9] for more details, and for the numerical data of this design.

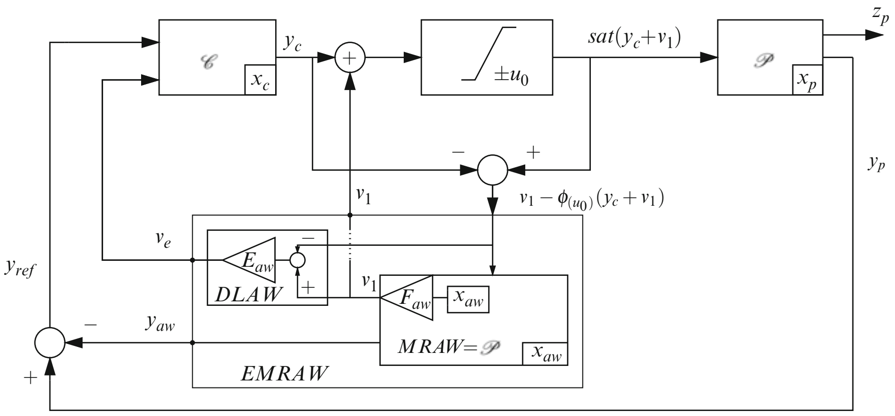


Fig. 3.6 Extended MRAW block diagram

3.3.2 Simulations on Relative Position Control

First, the closed-loop system (3.13) is simulated without anti-windup. System (3.13) is simulated from an initial condition of $x_p(0) = [\Delta z \quad \Delta\dot{z}]' = [-1\ 0]'$ and[2] $x_c = 0_{1\times5}$. Let us remind that Eq. (3.13) describes the relative position closed-loop system with the switching AF (3.9).

In Fig. 3.7, the solid line presents the Δz response of the system (3.13) (i.e., nonlinear response). The dot-dashed line shows the response of the system without saturation (i.e., linear response). One can realize the effects of the saturation as oscillations are induced in the nonlinear response of Δz. The control output and the performed thrust are also shown in Fig. 3.7. The thrustresponse is saturated and the control output oscillates. The effects of the saturation on the system (3.13) fully justify the introduction of the anti-windupcompensator.

In Fig. 3.8, the time evolution of the relative position of both satellites using a compensator developed by Thales Alenia Space (TAS), Cannes, France, is shown. It can be observed that this ad hoc compensator succeeds in avoiding the oscillations, even in presence of the nonlinearities in the closed loop. However, whereas this control law performs very well in single-input single-output control problems, it cannot be easily generalized to multivariable systems, and thus to the non-simplified formation flying control problem for which relative position and attitude are coupled. This is an additional motivation to develop the anti-windup techniques presented in this chapter.

The closed-loop system (3.16) is then simulated with the following anti-windup compensators:

- Static DLAW compensator (solid line)
- Full order dynamic DLAW compensator (dashed line)

[2]Throughout the simulations, the controller state is considered to be always initialized at the origin.

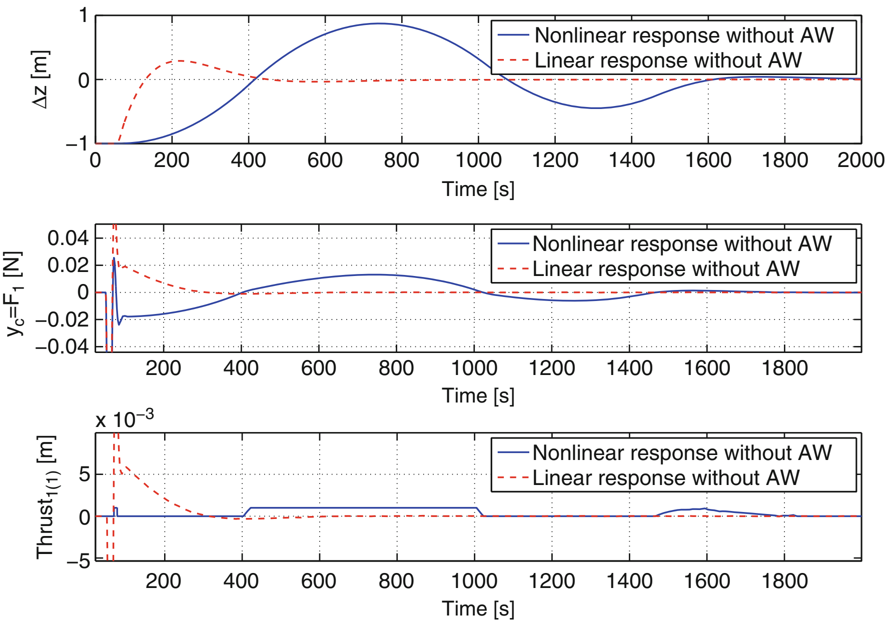


Fig. 3.7 Responses of the relative position without anti-windup compensator

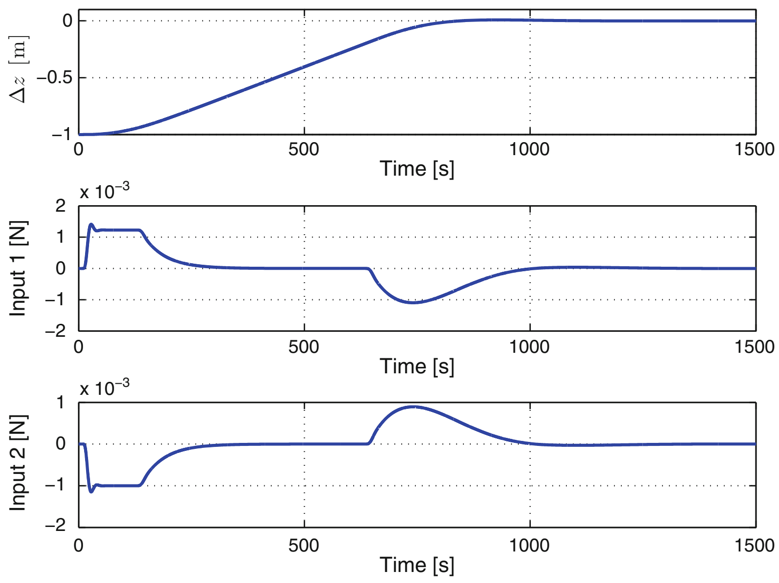


Fig. 3.8 Responses of the relative position with compensator developed by TAS. *Top*: relative position of both satellites. *Middle*: input of the first satellite. *Down*: input of the second satellite

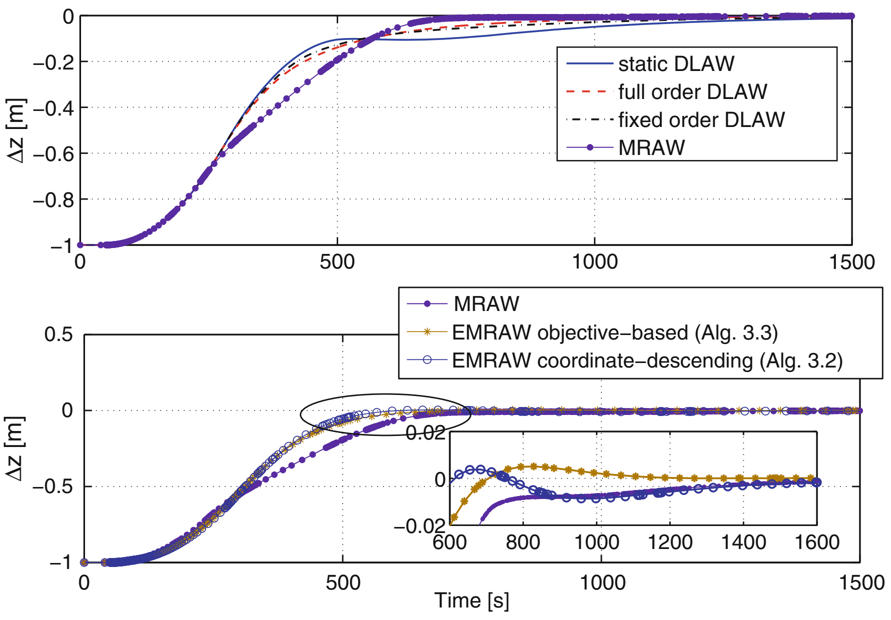


Fig. 3.9 Response of the relative position with several anti-windup compensators

- Fixed order dynamic DLAW compensator (dot-dashed line)
- MRAW compensator (line with dots)
- EMRAW compensator computed with a coordinate-descending algorithm that is denoted Algorithm 3.2 (line with circles)
- EMRAW compensator computed with an objective-based algorithm, that is denoted Algorithm 3.3 (line with stars)

Figure 3.9 shows the response of the relative position with the different anti-windup compensators in the control loop. The Δz responses have been split in two small figures for clarity purposes. From a general overview of the Fig. 3.9 one can conclude that the oscillation observed in Fig. 3.7 has disappeared with the introduction of the anti-windup compensator.

Let us start analyzing the response with DLAW. In Fig. 3.9 the response with a dynamic DLAW is smoother than in the static DLAW case which presents a drop on its slope. The anti-windup action, in the static case, ends as soon as there is no saturation. Therefore, the drop appears. On the contrary, the dynamic DLAW keeps modifying the controller action even without actuator saturation. In Fig. 3.10 the thrust response suddenly falls with the static DLAW case while it decreases progressively with the dynamic one. The advantage of a dynamic DLAW is then proven.

Also in Fig. 3.10, the thrust response with a full-order DLAW is noisy. The fast dynamics in the full-order case induces the high-frequency oscillation. The presence of fast and slow modes in the full-order DLAW generates numerical problems for both LMI computation step (bad conditioning effects) and simulation step (numerical precision effects). On the contrary, the fixed-order DLAW does not

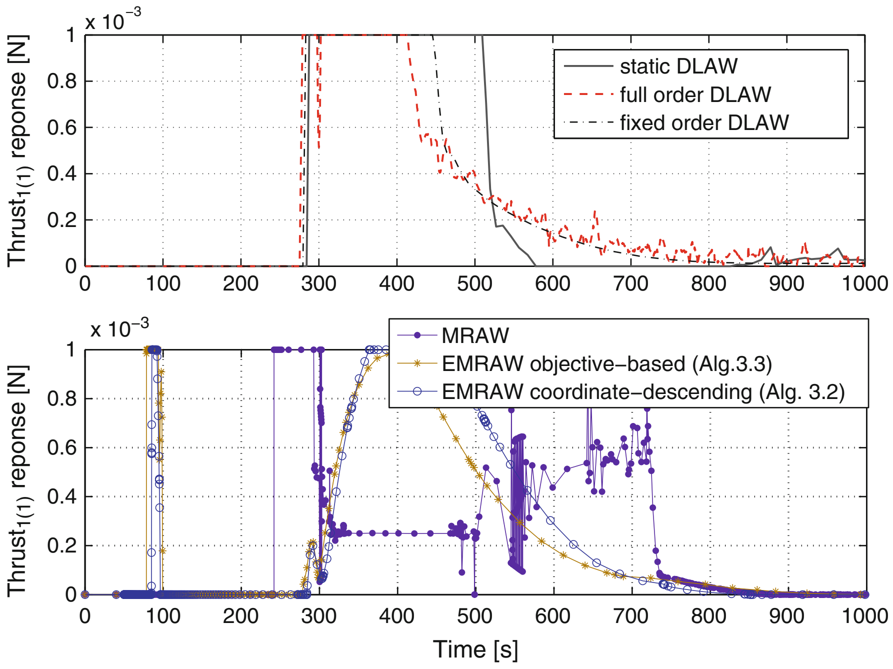


Fig. 3.10 Response of the first thruster with several Anti-windup compensators

present this oscillation because the fast dynamics were not chosen in the synthesis process. Therefore, the fixed-order provides a smooth response without high-frequency oscillations.

Let us analyze in Fig. 3.9 the Δz response with a MRAW/EMRAW: actually, fast responses are attained. Moreover, the response Δz of the system (3.16) is faster with the EMRAW. An explanation to this behavior is found in Figs. 3.11 and 3.12 where the anti-windup output y_{aw} and the reference signal y_{ref} are compared. The system response with the MRAW attains the reference, that is, $y_p = y_{\text{ref}}$ and $y_{\text{aw}} = 0$, before the response with the EMRAW. However, the reference y_{ref} for the MRAW is slower than the one for the EMRAW. Therefore, the actual response $y_p = \Delta z$ of the system (3.16) is faster with the EMRAW.

Another comment on the MRAW approach can be done by the analysis of the thrust response. In Fig. 3.10 large oscillations on the MRAW approach can be noticed. However the closer it gets to the origin, the higher the bang-bang frequency and hence the stronger the noise induced into the thrust action.

Finally, Fig. 3.13 presents the stability domain estimation for the different anti-windup compensators. The MRAW case has been omitted.

The EMRAW computed with the coordinate-descending algorithm (based on [23]) provides a larger estimation. Despite slight differences, all the compensators ensure (more or less) the same estimated maximal admissible $\Delta z_{\max}(0)$ (cutting point with $\Delta\dot{z} = 0$ axis). However, the estimation obtained with the full-order

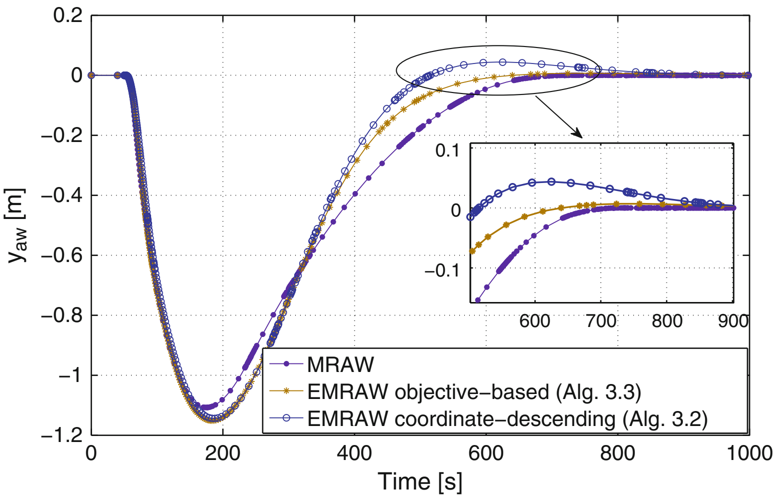


Fig. 3.11 Anti-windup output response for MRAW/EMRAW approaches

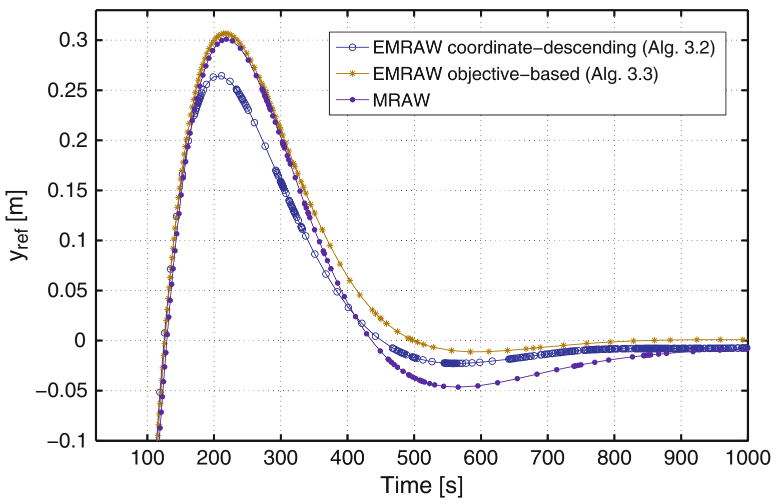


Fig. 3.12 Reference response for MRAW/EMRAW approaches

DLAW is smaller than the one obtained with the other compensators. This is due to some numerical problems. The difference between the maximum and the minimum eigenvalue of A_{aw} results on a bad conditioning of the LMIs conditions. As a consequence, the stability domain analysis for the full-order DLAW is unfeasible.

Let us make two final remarks on the stability domain estimation. First, notice that the size of the stability domain without anti-windup (dots line) is clearly

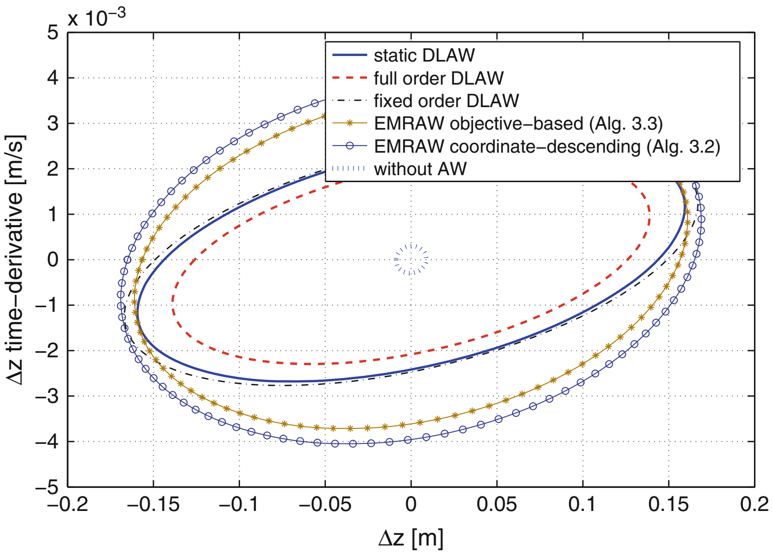


Fig. 3.13 Stability domain with several anti-windup compensators

Table 3.2 Summary values on the relative position control

	Static DLAW	Full-order DLAW	Fixed-order DLAW
$\Delta z_{\max}(0) = \frac{1}{\sqrt{\rho}}$ [m]	0. 14	0. 13	0.15
Time of response [s]	1, 570	1, 160	1, 390
Consumption [Ns]	2. 017	1. 829	1. 82
AW order	0	14	5
	MRAW	EMRAW Algorithm 3.2	EMRAW Algorithm 3.3
$\Delta z_{\max}(0) = \frac{1}{\sqrt{\rho}}$ [m]		0. 17	0. 16
Time of response [s]	750	580	630
Consumption [Ns]	1. 679	1. 894	1. 8192
AW order	2	2	2

smaller. The second point is the conservatism of the estimation. Simulations show that the system is stable (even without anti-windup) for an initial condition $x_p(0) = [-1 \;\; 0]'$. However, considering the estimated domain the maximal admissible initial condition is around $x_p(0) = [-0.15 \quad 0]'$.

Table 3.2 summarizes the main values characterizing the anti-windup compensators. These values are:

- The maximal admissible initial condition for the relative position $\Delta z_{\max}(0)$.
- The *time evolution* for Δz in seconds.
- The integral of the thrust response (value related to the *consumption*).
- The *order* of the anti-windup (AW) compensator.

A certain trade-off has to be done in the optimized compensator design between performance (consumption, speed of convergence) and size of the stability domain (ρ).

The time evolution appears as a remarkable value for the comparison. It has been defined as the time when the Δz response has reached the 99 % of the gap between its initial condition and the origin. In these simulations the initial relative position is $\Delta z = -1\, m$. It has been understood as $\Delta z = \pm\, 0.\, 01$ m.

Note the important gap between the time evolution of the system with a DLAW or with a MRAW/EMRAW. With these last approaches the responses are certainly faster. In addition, this improvement does not come with an undesirable increase of the consumption.

Let us remind that the value representing the consumption is just the integral of the thrusters response with relation to the time, and not the actual consumption. These values are however proportional.

Remark 2. Because of its simplicity, its efficiency in terms of time evolution, and its guaranteed stability domain, the EMRAW structure arises as an interesting architecture for the anti-windup computation. However, the initialization process for the associated algorithms is not trivial. In addition, the anti-windup compensator order is restricted to the order of the plant.

Remark 3. The full-order DLAW presents fast and slow modes. These modes generate numerical problems for both LMI computation step (bad conditioning effects) and simulation step (numerical precision effects). Therefore, the fixed-order DLAW is an interesting alternative to the full-order one. Although some strategies are proposed in the literature, the choice of the A_{aw} poles is not strictly formalized.

Remark 4. Even if in Table 3.2 the static DLAW presents the worst values of all compensators, these values are not too much far from those obtained with other approaches. On the other hand, the static DLAW is easy to compute because of the associated LMIs simplicity. It is also easy to implement because it is simply a static gain. Therefore, a systematic procedure can be designed for the static DLAW computation.

3.4 Conclusion

Anti-windup compensator design represents the core of this chapter. For ease of understanding and methods comparisons, the relative position control system has been taken as a simplified study case. Different anti-windup compensators have been evaluated for this problem. The closed-loop system modeling this control problem has been initially provided. The simplicity of the system has allowed the characterization of the main features of the anti-windup compensators. The simulations have allowed to point out the interest of the EMRAW approach. Similarly to the study presented in this chapter, the attitude and relative position control system has also been studied. This application presents the particularity of being a multi-variable system with a coupling of the attitude over the relative position. For more details on this last case, see [6].

Interesting point for further research, regarding the EMRAW, concerns on the anti-windup order reduction. Indeed, one main drawback of the EMRAW is that the plant and the anti-windup compensator are forced to have the same order. In large-order models, this constraint is unacceptable for future physical implementations. Therefore, it would be interesting to develop constructive methods to reduce the anti-windup order, for example, eliminating secondary dynamics of the considered plant. In this scenario, the results could be related with the anti-windup synthesis on an uncertain system.

References

1. Absil, O.: Science with pegase. In: Proceedings of 2nd TPF/Darwin International Conference, San Diego, CA, USA (2004)
2. Aström, K.J., Rundqwist, L.: Integrator windup and how to avoid it. In: Proceedings of the American Control Conference, Pittsburgh, PA, USA (1989)
3. Balas, G., Chiang, R., Packard, A., Safonov, M.: Robust Control Toolbox user's guide. The MathWorks Inc., Natick, MA (2007)
4. Berg, J.M., Hammet, K.D., Schwartz, C.A., Banda, S.S.: Analysis of the destabilizing effect of daisy chained rate-limited actuators. IEEE Trans. Contr. Syst. Tech. **4**, 171–176 (1996)
5. Biannic, J.M., Roos, C., Tarbouriech, S.: A practical method for fixed-order anti-windup design. In: Proceedings of 7th IFAC Symposium on Nonlinear Control Systems (NOLCOS), Pretoria, South Africa (2007)
6. Boada, J.: Satellite control with saturating inputs. Ph.D. thesis, ISAE, Toulouse, France (2010)
7. Boada, J., Prieur, C., Tarbouriech, S., Pittet, C., Charbonnel, C.: Anti-windup design for satellite control with microthrusteurs. In: Proceedings of AIAA GN&C conference, Chicago, IL, USA (2009)
8. Boada, J., Prieur, C., Tarbouriech, S., Pittet, C., Charbonnel, C.: Multi-saturation anti-windup structure for satellite control. In: Proceedings of the American Control Conference, Baltimore, MD, USA (2010)
9. Boada, J., Prieur, C., Tarbouriech, S., Pittet, C., Charbonnel, C.: Extended model recovery anti-windup for satellite control. In: Proceedings of the 16th IFAC Symposium on Automatic Control in AeroSpace, Nara, Japan (2010)
10. Boyd, S., Vandenberghe, L.S.P.: Convex Optimization. Cambridge Univesity Press, Cambridge (2004)
11. Durham, D.C.: Constrained control allocation. J. Guid. Contr. Dynam. **16**, 717–725 (1993)
12. Galeani, A.R., Teel, A.R., Zaccarian, L.: Constructive nonlinear anti-windup design for exponentially unstable linear plants. Syst. Contr. Lett. **56**(5), 357–365 (2007)
13. Galeani, S., Onori, A.R., Teel, A.R., Zaccarian, L.: Regional, semiglobal, global nonlinear anti-windup via switching design. In: Proceedings European Control Conference, Kos, Greece (2007)
14. Galeani, S., Tarbouriech, S., Turner, M.C., Zaccarian, L.: A tutorial on modern anti-windup design. Eur. J. Contr. **15**(3–4), 418–440 (2009)
15. Gaulocher, S.: Commande boucle fermée multivariable pour le vol en formation de vaisseaux spatiaux. Ph.D. thesis, Université de Toulouse, Toulouse, France (2007)

16. Gomes da Silva, J.M. Jr., Tarbouriech, S.: Anti-windup design with guaranteed regions of stability: An LMI-based approach. IEEE Trans. Automat. Contr. **50**(1), 106–111 (2005)
17. Grimm, G., Hatfield, J., Postlethwaite, I., Teel, A.R., Turner, M.C., Zaccarian, L.: Anti-windup for stable linear systems with input saturation: An LMI-based synthesis. IEEE Trans. Automat. Contr. **48**(9), 1509–1525 (2003)
18. Hu, T., Lin, Z.: Control Systems with Actuator Saturation. Analysis and Design. Birkhauser, Basel (2001)
19. Kapila, V., Grigoriadis, K. (eds.): Actuator Saturation Control. Marcel Dekker, New York (2002)
20. Kerr, M.L., Turner, M.C., Postlethwaite, I.: Practical approaches to low-order anti-windup compensator design: A flight control comparison. In: IFAC World Congress, Seoul, Korea (2008)
21. Löfberg, J.: Yalmip: A toolbox for modeling and optimization in matlab. In: CACSD Conference, Taipei, Taiwan (2004)
22. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
23. Peaucelle, D., Arzelier, D.: An efficient numerical solution for H_2 static output feedback synthesis. In: Proceedings of European Control Conference, Porto, Portugal (2001)
24. Pirson, L., Charbonnel, C., Udrea, B., Rennie, M., McGuinness, P., Palomo, P.: Darwin precursor demonstration mission: The ICC2 study, from GNC design to real-time test bench validation. In: Proceedings of 6th GNC ESA, Loutraki, Greece (2005)
25. Roos, C., Biannic, J.M.: A convex characterization of dynamically-constrained anti-windup controllers. Automatica **44**(8), 2449–2452 (2008)
26. Stein, G.: Bode lecture: Respect the unstable. In: Proceedings of Conference on Decision and Control, Tampa, USA, December (1989)
27. Tarbouriech, S., Turner, M.C.: Anti-windup synthesis: An overview of some recent advances and open problems. IET Control Theory Appl. **3**(1), 1–19 (2009)
28. Tarbouriech, S., Prieur, C., Gomes da Silva, J.M. Jr.: Stability analysis and stabilization of systems presenting nested saturations. IEEE Trans. Automat. Contr. **51**(8), 1364–1371 (2006)
29. Tarbouriech, S., Garcia, G., Gomes da Silva, J.M. Jr., Queinnec, I.: Stability and stabilization of linear systems with saturating actuators. Springer, London (2011)
30. Teel, A.R., Kapoor, N.: The $\mathcal{L}_2$ antiwindup problem: Its definition and solution. In: Proceedings of the 4th European Control Conference, Brussels, Belgium (1997)
31. Turner, M.C., Postlethwaite, I.: A new perspective on static and low order anti-windup synthesis. Internat. J. Control **77**(1), 27–44 (2004)
32. Zaccarian, L., Teel, A.R.: A common framework for anti-windup, bumpless transfer and reliable designs. Automatica **38**(10), 1735–1744 (2002)
33. Zaccarian, L., Teel, A.R.: Modern Anti-Windup Synthesis. Princeton University Press, Princeton (2011)

Chapter 4
The ESA NLP Solver WORHP

Christof Büskens and Dennis Wassel

Abstract **W**e **O**ptimize **R**eally **H**uge **P**roblems (WORHP) is a solver for large-scale, sparse, nonlinear optimization problems with millions of variables and constraints. Convexity is not required, but some smoothness and regularity assumptions are necessary for the underlying theory and the algorithms based on it. WORHP has been designed from its core foundations as a sparse sequential quadratic programming (SQP) / interior-point (IP) method; it includes efficient routines for computing sparse derivatives by applying graph-coloring methods to finite differences, structure-preserving sparse named after *Broyden*, *Fletcher*, *Goldfarb* and *Shanno* (BFGS) update techniques for Hessian approximations, and sparse linear algebra. Furthermore it is based on reverse communication, which offers an unprecedented level of interaction between user and nonlinear programming (NLP) solver. It was chosen by ESA as the European NLP solver on the basis of its high robustness and its application-driven design and development philosophy. Two large-scale optimization problems from space applications that demonstrate the robustness of the solver complement the cursory description of general NLP methods and some WORHP implementation details.

Keywords Nonlinear optimization • Large-scale • Mathematical optimization • NLP

4.1 Introductory Remarks

Nonlinear optimization has grown to a key technology in many areas of aerospace industry, especially for solving discretized optimal control problems with differential equation systems, like ordinary differential equations (ODEs),

C. Büskens (✉) • D. Wassel
Center for Industrial Mathematics, University of Bremen, Bremen, Germany
e-mail: bueskens@worhp.de; dwassel@worhp.de

differential-algebraic equations (DAEs), and partial differential equations (PDEs). Applications are satellite control, shape-optimization, aerodynamics, trajectory planning, reentry problems, and interplanetary flights. One of the most extensive areas is the optimization of trajectories for aerospace applications; this book is proof to that notion. Nonlinear optimization problems arising from these applications typically are large and sparse. Previous methods for solving nonlinear optimization problems were originally developed for small- to medium-sized[1] and dense problems. Many challenging optimization problems cannot be tackled by these solvers due to their size.

Solving large-scale problems requires an NLP solver to exploit as much of the problem structure as possible. This implies efficient storage of sparse vectors and matrices, special linear algebra methods for solving sparse, large-scale linear systems of equations and efficient methods for approximation of sparse first and second derivatives.

Most of the available optimization methods employ the *Broyden*, *Fletcher*, *Goldfarb* and *Shanno* (BFGS) update technique for the Hessian of the Lagrange function. This update technique has several advantages, such as guaranteeing the positive definiteness of the Hessian approximation, which has algorithmic benefits, and the ability to achieve superlinear convergence of the overall algorithm. Its most important advantage is that BFGS-type updates are very cheap to evaluate, while its most important drawback is the fact that BFGS approximations are dense in general, even if the underlying optimization problem is sparse, which precludes their use in large-scale optimization. **W**e **O**ptimize **R**eally **H**uge **P**roblems (WORHP) has therefore been designed to use exact Hessians, if available, or to approximate them numerically efficient by sophisticated, graph-coloring-based finite-difference methods or by various sparse variations of the classical BFGS update; the structure-preserving *sparse BFGS* (SBFGS) update is a unique feature of WORHP.

This chapter is organized as follows: We will first give an overview about nonlinear optimization and some background about methods for solving such problems. Then we introduce the general methodology of the new solver WORHP and present its advantages and techniques in more detail. Numerical results of some large-scale optimization problems in space applications demonstrate the capabilities of the method.

4.2 Nonlinear Optimization

WORHP solves nonlinear optimization problems of the form

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \quad f(x) \\ &\text{subject to} \quad \begin{pmatrix} l \\ L \end{pmatrix} \le \begin{pmatrix} x \\ g(x) \end{pmatrix} \le \begin{pmatrix} u \\ U \end{pmatrix} \end{aligned} \tag{OP}$$

[1] We regard problems with less than 1,000 variables and constraints as small and problems with up to 5,000 variables and constraints as medium-sized, although this is not a sharp definition.

with bounds

$$
\begin{aligned}
l, u &\in (\mathbb{R} \cup \{\pm\infty\})^n, \\
L, U &\in (\mathbb{R} \cup \{\pm\infty\})^m,
\end{aligned}
$$

and user-defined functions $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$. If either x or g are unbounded, the corresponding bound is $\pm\infty$ (in practice, ∞ is replaced by a finite, large constant, such as 10^{20}). One should note that each g_i needs to have at least one finite bound, since it is redundant otherwise and should be removed from the model.

Both the **objective function** f and the **constraints** g may be linear, quadratic, or nonlinear (no convexity is assumed). The theoretical foundation requires certain smoothness and regularity assumptions, which unfortunately are next to impossible to ascertain for most problems of practical relevance. In practice, WORHP often finds solutions even if the differentiability requirement is not satisfied.

The constraints g in Eq. (OP) may have mixed linear and nonlinear components, i.e., it may be of the form

$$g(x) = \begin{pmatrix} Ax \\ g_{\mathrm{nl}}(x) \end{pmatrix}$$

with a matrix $A \in \mathbb{R}^{k \times n}$. Some NLP solvers impose this ordering of the constraints into linear and nonlinear constraints on the user, while WORHP accepts constraints in arbitrary order, without any requirement to sort or classify them (though future versions are planned to offer an optional mechanism for tagging constraints as linear, which can be used to evaluate the constraint Jacobian in a more efficient way).

Problem (OP) may be used to *maximize* an objective function by replacing the objective f by its negative $-f$.

Special cases of Eq. (OP) are linear optimization problems (linear programming), quadratic optimization problems (quadratic programming, QP), discretized optimal control problems, trajectory optimization problems, or constrained nonlinear least-squares problems.

Internally, WORHP solves problems in standard mathematical form

$$
\begin{aligned}
\min_{x \in \mathbb{R}^N} \quad & F(x) \\
\text{subject to} \quad & G_i(x) = 0, \quad i = 1, \dots, M_e \\
& G_j(x) \le 0, \quad j = M_e + 1, \dots, M,
\end{aligned}
\tag{NLP}
$$

which is equivalent to the initial problem formulation by a simple transformation; usually $F = f$ and $N = n$, but in general $G := (G_1, \dots, G_M) \neq g$ and $M > m$, except for special cases.

Formulation (NLP) is convenient for expressing mathematical properties, while formulation (OP) is more convenient for the implementation of practical optimization problems.

The aim is to find a vector $x^* \in \mathbb{R}^N$, which satisfies the constraints G and uses the remaining degrees of freedom to minimize the given objective function F. The sets

$$\begin{aligned} I(x^*) &:= \left\{ i \in \{M_e + 1, \ldots, M\} \middle| G_i(x^*) = 0 \right\}, \\ J(x^*) &:= I(x^*) \cup \{1, \ldots, M_e\}, \end{aligned} \tag{AS}$$

are called set of active indices. To find x^*, we introduce the Lagrangian

$$L(x, \lambda) := F(x) + \lambda^T G(x),$$

where $\lambda \in \mathbb{R}^M$ is the vector of the Lagrange multipliers. The necessary first-order optimality conditions, also called KKT conditions, guarantee that if x^* is a local minimum of Eq. (NLP) and moreover x^* is regular (compare Mangasarian-Fromowitz [14]), then there exists $\lambda^* \in \mathbb{R}^M$ such that the following holds:

$$\begin{aligned} \nabla_x L(x^*, \lambda^*) &= \nabla_x F(x^*) + (\lambda^*)^T \nabla_x G(x^*) = 0, \\ \lambda_i^* &\geq 0, \; i \in I(x^*), \\ \lambda_j^* &= 0, \; j \notin J(x^*), \\ (\lambda^*)^T G(x^*) &= 0. \end{aligned} \tag{KKT}$$

If additionally $\nabla_x G_i(x^*)$, $i \in J(x^*)$ are linearly independent, then λ^* is unique. This criterion is called linear independence constraint qualification (LICQ) and is used to search for optimal solutions of Eq. (NLP).

4.2.1 Sequential Quadratic Programming

WORHP is a mixed SQP-IP solver: it uses sequential quadratic programming (SQP) on NLP level and an Interior-Point method on QP level, i.e., the quadratic subproblems [see below, Eq. (QP)] generated by the SQP method. The general idea of SQP methods was first introduced by Wilson [17] in 1963 and revived by Han [10] in 1976. Since then they belong to the most frequently used algorithms for the solution of practical optimization problems due to their robustness and their good convergence properties. SQP methods can be proven to achieve global convergence and locally superlinear convergence rate; even locally quadratic convergence can be achieved if the Hessian of the Lagrangian $\nabla_{xx}^2 L(x, \lambda)$ is available.

The fundamental principle of SQP methods is to find KKT points, i.e., points that satisfy the *necessary* optimality conditions. While *sufficient* optimality conditions can be formulated, they are usually not tested, since this is computationally costly. In rare cases, this can mislead NLP solvers to consider local maxima or saddle points as minima, but this is mostly limited to constructed academic examples and hardly observed on practical problems.

SQP methods essentially apply Newton's method (compare [7]) to find zeros of $\nabla_x L(x, \lambda)$ that satisfy the constraints G, using a local linearization. In this basic form, quadratic subproblems of the form

$$\begin{aligned} \min_{d \in \mathbb{R}^N} \quad & \frac{1}{2} d^T \nabla^2_{xx} L(x, \lambda) d + \nabla_x F(x) d, \\ \text{subject to} \quad & G_i(x) + \nabla_x G_i(x) d = 0, \; i = 1, \dots, M_e \\ & G_j(x) + \nabla_x G_j(x) d \leq 0, \; j = M_e + 1, \dots, M \end{aligned} \tag{QP}$$

are solved to obtain a *search direction d*.

Often, the Hessian of the Lagrangian $\nabla^2_{xx} L(x, \lambda)$ is replaced by an update of BFGS type, which has the additional benefit that only strictly convex quadratic programs have to be solved. This strategy works well for small- to medium-sized problems, but it turns out to be infeasible for large-scale problems, since the update formulae generally yield dense matrices, which in turn lead to dense Hessian approximations. To solve large-scale problems, one is thus forced to fall back to the exact Hessian of the Lagrangian. This matrix may be indefinite and thus lead to non-convex quadratic subproblems, which makes a regularization step (compare Sect. 4.2.1.6) necessary. On the upside, it facilitates solution of optimal control problems (OCPs) when using full discretization—one of WORHP's intended major application areas—because discretization invariance may be achieved with the Hessian, i.e., the number of iterations needed to solve a given OCP remains constant irrespective of the discretization grid.

Computation of derivatives is a crucial element in nonlinear optimization. Basically first derivatives, i.e., the gradient of the objective function and the Jacobian of the constraints, are necessary in order to find a descent direction.

The overall SQP algorithm is an iterative process that defines a series of points $x^{[k]}$ (called *iterates*) that (ideally) converge to an optimal point. It consists very roughly of following steps:

1. Check termination criteria (see Eq. (KKT) and Sect. 4.2.2.3).
2. Locally approximate Eq. (NLP) by Eq. (QP) and solve it (see Sects. 4.2.1.1, 4.2.1.2 and 4.2.1.6).
3. Use the solution from 2 to define the new iterate, employing either a merit function (Sect. 4.2.1.3) or a filter method (Sect. 4.2.1.5) to find a suitable step size (Sect. 4.2.1.4).

4.2.1.1 Interior-Point Methods

Interior-point (IP) methods were originally designed to solve linear optimization problems, but where later generalized to solve other classes of problems as well (even NLPs).

Applied to problems of the form Eq. (QP) whose solution is needed by the NLP solver, IP methods transform problems of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \quad Ax = b, \\ Cx \leq d \end{aligned}$$

with $Q = \nabla^2_{xx} L(x^{[k]},\ \lambda^{[k]}),\ c = \nabla_x F(x),\ A = \nabla_x(G_i)(x^{[k]}),\ i = 1,\dots,M_e,$ and $C = \nabla_x(G_j)(x^{[k]}),\ j = M_e+1,\dots,M$ into the form

$$\begin{aligned} \min_{(x,s)\in\mathbb{R}^{n+m}} \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \quad Ax = b, \\ Cx + s = d, \\ s \geq 0. \end{aligned} \tag{IP}$$

The fundamental concept in IP methods is to eliminate the inequality constraints $Cx \leq d$ by introducing (strictly positive) slack variables s, and adding them to the objective function using a log-barrier term, where $\nu > 0$ weights the penalty term:

$$\begin{aligned} \min_{(x,s)\in\mathbb{R}^{n+m}} \frac{1}{2} x^T Q x + c^T x - \nu \sum_{i=1}^{m} \log s_i \\ \text{s.t.} \quad Ax = b, \\ Cx + s = d. \end{aligned} \tag{IP$_{\log}$}$$

Then a sequence of equality constrained nonlinear programs is solved while simultaneously the weight parameter $\nu > 0$ in the objective function tends to zero. The optimal solution fulfills $s > 0$, because the objective function would become infinite otherwise. This property of IP methods has coined their name, since the log-barrier term forces the iterates to stay strictly in the interior of the feasible region defined by the inequality constraints. This also implies one major technical restriction of IP method, namely the requirement that the initial guess has to be feasible.

The KKT conditions for the barrier problem are

$$\begin{aligned} Qx + A^T y + C^T z &= -c, \\ Ax &= b, \\ Cx + s &= d, \\ -\frac{\nu}{s_i} + z_i &= 0, \quad i = 1, \dots, m, \end{aligned} \tag{IP$_{\text{KKT}}$}$$

where y and z are Lagrange multipliers for the constraints in Eq. (IP$_{\log}$). The KKT conditions (IP$_{\text{KKT}}$) can also be interpreted as perturbed KKT conditions of problem (IP). The last condition can be transformed to

$$\nu = \frac{1}{m} z^T s,$$

the so-called complementarity measure, by multiplying the equation with $s_i > 0$ and summation afterwards. The complementarity measure quantifies the violation of the complementarity condition in Eq. (IP).

The approach of an IP method can be described now: The iterative solutions remain in the strict interior of the feasible region, since the barrier term would become too large otherwise. A decrease of ν allows solutions near the boundary of the feasible region.

Primal-dual interior-point methods generate primally feasible slack variables $s^{[k]} > 0$ and dually feasible multipliers $z^{[k]} > 0$ such that the complementarity measure $\nu_k = \left(z^{[k]}\right)^T s^{[k]}/m$ tends to zero. Additionally, $x^{[k]}$ and $y^{[k]}$ must be computed such that the residuals

$$\begin{aligned} r_Q^{[k]} &= Qx^{[k]} + A^T y^{[k]} + C^T z^{[k]} + c, \\ r_A^{[k]} &= Ax^{[k]} - b, \\ r_C^{[k]} &= Cx^{[k]} + s^{[k]} - d \end{aligned}$$

converge to zero. In the event of convergence, we obtain $\lim z^{[k]} \geq 0$, $\lim s^{[k]} \geq 0$, $\lim \nu_k = 0$, $\lim r_*^{[k]} = 0$, $* \in \{Q, A, C\}$, such that the KKT conditions for Eq. (IP) are fulfilled. The initial guess of Eq. (IP$_{\text{KKT}}$) does not need to be feasible. A complete QP solver implementation is described in Gertz and Wright [6]. They use a heuristic of Mehrotra which was successfully applied to linear programs. Two linear equation systems have to be solved for one iteration: The first linear equation system basically represents a Newton step for the nonlinear equation system (IP$_{\text{KKT}}$) (with $\nu = 0$). This is a so-called affine-scaling step. The second step, the so-called corrector step, produces an improvement of the complementarity measure.

The solution $x \in \mathbb{R}^n$ of Eq. (IP) is used as search direction $d^{[k]}$ by the SQP method, and the multipliers y and z are used to update the Lagrange multipliers λ in the NLP.

4.2.1.2 Constraint Relaxation

The problem (QP) may have no solution at all, if there are inconsistencies in the linearized constraints. A simple example is illustrated by the constraint

$$G(x) = 1 - x^2 \leq 0.$$

Linearizing this constraint at $x^{[0]} = 0$, we arrive at the inequality

$$G(x^{[0]}) + \nabla_x G(x^{[0]}) \cdot d = 1 + 0 \cdot d \leq 0,$$

which obviously cannot be satisfied. For the practical implementation a relaxed formulation of Eq. (QP) is solved with respect to $z = (d, \delta) \in \mathbb{R}^{N+1}$:

$$\begin{aligned} \min_{z \in \mathbb{R}^{N+1}} \quad & \frac{1}{2} d^T \nabla_{xx}^2 L(x, \lambda) d + \nabla_x F(x) d + \frac{1}{2} \eta_r \delta^2, \\ \text{subject to} \quad & G_i(x)(1-\delta) + \nabla_x G_i(x) d = 0, \; i = 1, \ldots, M_e \\ & G_j(x)(1-\sigma_j \delta) + \nabla_x G_j(x) d \leq 0, \; j = M_e + 1, \ldots, M, \end{aligned} \qquad (\mathrm{QP_r})$$

where δ denotes the relaxation variable, $\eta_r > 0$ is a penalty weight, and

$$\sigma_j = \begin{cases} 0, & \text{if } G_j^{[k]} < 0, \\ 1, & \text{otherwise} \end{cases}, \; j = M_e + 1, \ldots, M.$$

If $\delta = 1$, the problem always has $d = 0$ as feasible (though somewhat useless) solution, and conversely, if $\delta = 0$, then Eq. ($\mathrm{QP_r}$) is equivalent to Eq. (QP). The *constraint relaxation penalty* term η_r that strives to minimize δ via the $\frac{1}{2}\eta_r \delta^2$ term in the objective of Eq. ($\mathrm{QP_r}$) is therefore chosen to achieve sufficiently small values of δ.

4.2.1.3 Merit Functions

Merit functions are needed to obtain a scalar measure of "goodness" of the trial point

$$x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}, \qquad (4.1)$$

which is an update of the current iterate $x^{[k]}$ by the result $d^{[k]}$ of Eq. (QP) and a scalar step size $\alpha^{[k]}$ that is introduced to achieve global convergence (since SQP is essentially derived from Newton's method that also depends on line search for globalization). This measure consists of a scalar quantification of both the objective and the constraints. Well-known merit functions are the L_1-penalty function

$$L_1(x;\eta) := F(x) + \sum_{i=1}^{M_e} \eta_i |G_i(x)| + \sum_{i=M_e+1}^{M} \eta_i \max\{0, G_i(x)\},$$

and the augmented Lagrangian

$$L_a(x,\lambda;\eta) := F(x) + \sum_{i=1}^{M_e} \lambda_i G_i(x) + \frac{1}{2}\sum_{i=1}^{M_e} \eta_i G_i^2(x) + \frac{1}{2\eta_i}\sum_{i=M_e+1}^{M} \Big(\big(\max\{0, \lambda_i + \eta_i G_i(x)\}\big)^2 - \lambda_i^2 \Big),$$

where $\eta \in \mathbb{R}^M$, with $\eta_i \geq 0$, $i = 1, \ldots, M$, is a penalty vector. The *penalty parameters* η used in both variants are updated iteratively, depending on derivative, multiplier, and progress information. For more details compare Schittkowski [16], Gill et al. [7], and Gill et al. [8].

4.2.1.4 Line Search

After determining the search direction $d^{[k]}$ from the QP, we have to find a suitable step size $\alpha^{[k]}$. To this end we use a line-search method based on the Armijo rule, a standard technique described for instance in [7]. We define

$$\phi(\alpha) := M(x^{[k]} + \alpha d^{[k]}, \Psi(\lambda^{[k]}, \alpha); \eta),$$

where $\Psi(\lambda^{[k]}, \alpha)$ is an update of the multipliers depending on α, the current multipliers $\lambda^{[k]}$, and the new multiplier approximation λ_{QP} from Eq. (QP), for instance $\Psi(\lambda^{[k]}, \alpha) = (1-\alpha)\lambda^{[k]} + \alpha\lambda_{QP}$, and M is one of the merit functions introduced in Sect. 4.2.1.3.

In general, a good choice would be an α which exactly minimizes $\phi(\alpha)$, but unfortunately, this defines another nonlinear optimization problem. Although the original problem is reduced to a one-dimensional line search, it is too expensive to solve, especially with large-scale problems. A more realistic goal is thus to find the largest step size α that satisfies

$$\phi(\alpha) < \phi(0) = M(x^{[k]}, \lambda^{[k]}; \eta),$$

by using an inexact line search.

Starting with a maximum step size $\alpha_{\max} \in (0, 1]$ and a decrease factor $\beta \in (0, 1)$, candidates for the step size are

$$\{\alpha_j = \beta^j \alpha_{\max} \mid j = 0, 1, 2, \ldots, l_{\max}\},$$

where $l_{\max} = \max\{l \in \mathbb{N} \mid \beta^l \alpha_{\max} \geq \alpha_{\min}\}$ defines the smallest step size allowed. As the first trial step size we choose $\alpha_0 = \alpha_{\max}$. If the Armijo condition

$$\phi(\alpha_j) \leq \phi(0) + \sigma \alpha_j \phi'(0), \tag{4.2}$$

where $\sigma \in \mathbb{R}^+$ is a suitable factor and $\phi'(0)$ the derivative of ϕ with respect to α, is not satisfied, $\alpha_1 = \beta^1 \alpha_0$ is calculated and Eq. (4.2) is checked again. This is done iteratively until either a suitable α is found, or the line search fails when $i > l_{\max}$.

4.2.1.5 Filter

An alternative to merit functions is the recently developed filter method for determining the step size α in Eq. (4.1).

A trial iterate $x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}$ is accepted by the filter if it either improves the objective function F or the constraint violation $H(x) := ||G^+(x)||$, where

$$G_i^+(x) := \begin{cases} G_i(x), & \text{for } i = 1, \ldots, M_e, \\ \max\{0, G_i(x)\}, & \text{for } i = M_e + 1, \ldots, M. \end{cases} \tag{4.3}$$

To facilitate notation, we will now index iteration-dependent function values with the iteration index, e.g., $F^{[k]} := F(x^{[k]})$

The point $(F^{[k]}, H^{[k]})$ is said to be dominated by the point $(F^{[j]}, H^{[j]})$ if both

$$F^{[j]} \leq F^{[k]} \quad \text{and} \quad H^{[j]} \leq H^{[k]} \tag{4.4}$$

A filter $\mathscr{F}$ is a list of pairs $(F(x), H(x))$ such that no entry dominates any other. A trial iterate $x^{[k+1]}$ is accepted by the filter if the corresponding pair $(F^{[k+1]}, H^{[k+1]})$ is not dominated by any filter entry. The pair $(F^{[k+1]}, H^{[k+1]})$ can then be added to the filter, and all entries that are dominated by this new pair have to be deleted. Otherwise a new trial iterate with a smaller step size is tested.

The filter can be initialized as $\mathscr{F} = (\infty, \infty)$ or with an upper bound $H_{\max} > 0$ for the constraint violation as $\mathscr{F} = (\infty, H_{\max})$.

For an actual implementation of a filter method, additional criteria are needed to prevent convergence to infeasible points or insufficient progress between iterations.

4.2.1.6 Hessian Regularization

To guarantee that Eq. (QP) has a unique solution, one has to ensure that $H_L = \nabla^2_{xx} L(x, \lambda)$ is positive definite on the kernel of the Jacobian of the active constraints.

While there exist NLP solvers that operate on the reduced Hessian (e.g., SNOPT [9]), this approach is infeasible for large-scale problems, since it requires a QR-factorization of the Jacobian. Instead the whole Hessian is updated via

$$H = H_L + \tau(|\sigma| + 1)I. \tag{4.5}$$

The parameter $\tau \in [0, 1]$ is used to damp the Hessian update, and σ is the (lower) Gerschgorin bound on the most negative eigenvalue of $H_L = (h_{ij})_{ij} \in \mathbb{R}^{N\times N}$, which is determined by

$$\sigma = \min_{1\leq i\leq n}\left\{h_{ii} - \sum\nolimits_{i\neq j}^{n} |h_{ij}|\right\}.$$

The original idea was suggested by Levenberg [13]. He used the matrix $\overline{\tau}I$ as an approximation of the Hessian for least-squares problems.

The choice of τ is crucial for the rate of convergence of the overall algorithm. The setting $\tau = 1$ guarantees positive definiteness of the Hessian but impairs convergence, since it may cause a large perturbation of the original Hessian. On the other hand, $\tau = 0$ may cause problems in the QP solver, since in this case the original Hessian without regularization is used. The idea of Betts [1] is to reduce τ when the predicted reduction in the merit function coincides with the actual one, and increase the parameter otherwise.

4.2.2 WORHP Implementation

While WORHP is based on an SQP method, it has a number of design choices that set it apart from textbook or research-oriented implementations. Some of its architectural features are driven by practical considerations and experience, since WORHP's design goal is to provide a robust tool for solving demanding optimization problems in practical applications.

4.2.2.1 Derivative Approximations

Derivatives play a crucial role for solving NLP problems. As a rule of thumb, NLP solvers in general, and WORHP in particular, work most reliably if *analytic* derivatives are supplied to it. Unfortunately, this is challenging or outright impossible for complex optimization models, especially for discretized optimal control problems with complex dynamics; some models in (aero)space optimization even involve interpolated tabular data.

Finite Difference: To facilitate practical use of mathematical optimization tools, WORHP offers finite-difference and BFGS modules: Naïve finite-difference approaches such as element-wise perturbations with "small" $\varepsilon > 0$ and the i-th unit vector e_i

$$\frac{\partial \phi}{\partial x_i}(x) \approx \frac{\phi(x + \varepsilon e_i) - \phi(x)}{\varepsilon}, \quad i = 1, \dots, n,$$

to compute $\nabla \phi$ are infeasible for large-scale problems, since n is large, and evaluations of ϕ are often computationally expensive. If $\nabla \phi$ is sparse, not all variables need to be perturbed, so ∇F can mostly be approximated quite efficiently, if the sparsity structure is exploited; this is particularly common in discretized optimal control problems, where F often depends on a single optimization variable only.

An equally simple approach is not possible for the constraint Jacobian ∇G, since G usually depends on *all* optimization variables, and in most cases cannot be evaluated component wise. To reduce the computational cost, WORHP applies graph-coloring-based methods that allow to approximate several partial derivatives of a vector-valued with a single function evaluation.

Example 1. Let the Jacobian of G have sparsity structure

$$\nabla G = \begin{pmatrix} * & * & & \\ & * & * & \\ & & * & * \end{pmatrix} = \begin{pmatrix} \bullet & \circ & & \\ & \circ & \bullet & \\ & & \bullet & \circ \end{pmatrix}.$$

Entries $*$ denote Jacobian entries that are structurally nonzero, although they may attain the numeric value 0, depending on x, while missing entries denote structural zeros, i.e., entries that are always zero, independent of x. Typical large-scale problems have sparsity, i.e., the ratio between total elements and structural nonzeros, of $< 1\%$.

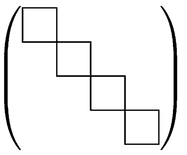
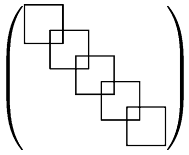
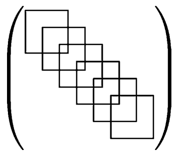
We can group variables if all rows of ∇G depend on at most one variable per group. One possible grouping is denoted by $\circ$ and $\bullet$ in the example above. To calculate a finite-difference approximation of the Jacobian, we can now perform multiple perturbations in a single evaluation, and calculate the whole Jacobian approximation by evaluating

$$\frac{G\big(x + \varepsilon(e_1 + e_3)\big) - G(x)}{\varepsilon} \quad \text{and} \quad \frac{G\big(x + \varepsilon(e_2 + e_4)\big) - G(x)}{\varepsilon},$$

i.e., two evaluations of G (omitting the unperturbed one, which can be cached) in contrast to four evaluations for the naïve approach. □

BFGS-Type Update Methods: Since most—if not all—large-scale optimization problems are highly structured, this grouping approach can drastically reduce the computational cost of finite-difference approximations of the Jacobian of G and with some extensions of the concept also of the Hessian of L. It is not uncommon for the grouping approach to require only a handful of evaluations of G, even if the problem has millions of variables and constraints. The grouping approach has the additional benefit of discretization-invariance: When solving OCPs using a common transcription method, the number of groups, and thus the number of function evaluations, remains constant irrespective of the grid size.

Since finite-difference approximations, especially the second-order ones, may still be expensive despite the grouping approach, and are inherently ill-conditioned, particularly difficult or large problems are not admissible to second-order derivative approximations. This is the realm of BFGS-type approximations. As mentioned earlier, the standard BFGS update causes 100% fill-in[2] and is thus inadmissible for problems with more than a few thousand variables. Due to the favorable properties of the BFGS update, it was desirable to find sparse extensions of the standard technique. A first generalization is to perform non-intersecting, intersecting, or even multiply intersecting block-BFGS updates of the form

$B_{ni} =$ , $B_i =$ , $B_{mi} =$ .

The non-intersecting case is straightforward because the classic BFGS method can be applied to the individual blocks, while the intersecting cases need a technique called *convexity shifts* to maintain the BFGS properties. By choosing appropriate block sizes and overlapping, the structure of Hessian matrices with small bandwidths can be approximated or reconstructed using BFGS matrices of B_i or B_{mi} structure. If, however, the Hessian has (many) far off-diagonal entries, none of these approaches can cover them without sacrificing a substantial degree of sparsity (which is essential for adequate performance in large-scale optimization). Even though the solver is able to cope with missing elements and will even converge if the Hessian is replaced by the identity matrix, this adversely affects convergence order. A worse (lower) convergence order in turn causes the NLP solver to require more—possibly *many* more—iterations to converge to an optimal point. For these cases, WORHP introduced the so-called SBFGS method. One central idea behind this method is illustrated by Example 2.

Example 2. Consider the sparse matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & a_{32} & a_{33} & 0 \\ 0 & a_{42} & 0 & a_{44} \end{pmatrix}.$$

[2] Operations on sparse matrices that destroy structural zeros are said to cause *fill-in*. Since fill-in adversely affects the performance of sparse matrix operations, numerical algorithms take (often extensive) measures to prevent or reduce it.

By selecting and rearranging the elements of A, we can form three dense sub-matrices

$$A_1 = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad A_2 = \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix}, \quad A_3 = \begin{pmatrix} a_{22} & a_{24} \\ a_{42} & a_{44} \end{pmatrix}$$

and perform the standard BFGS update on them, to obtain $\tilde{A}_i$. An updated form $\tilde{A}$ of the original matrix A can then be reassembled from the $\tilde{A}_i$, since all elements are accounted for. However, a_{22} is present in all three sub-matrices A_i, resulting in *three* potentially different elements $\tilde{a}_{22[i]}$ after the dense BFGS updates on $\tilde{A}_i$. The updated element $\tilde{a}_{22}$ can be obtained from a convex combination $\sum_i \lambda_i \tilde{a}_{22[i]}$ with suitably chosen λ_i.

The SBFGS method uses the above idea to decompose the Hessian into smaller dense sub-matrices, perform the classic BFGS update on them, and then reassemble them appropriately. The mathematical challenge is to preserve certain properties of the update to admit a convergence proof and maintain positive definiteness, while the technical challenge is to find the minimum number of dense sub-matrices of maximum size. Details can be found in [11].

4.2.2.2 Technical Features

WORHP has a number of non-standard technical features that are meant to improve usability.

Reverse Communication: The architecture of WORHP is based on the *reverse communication* (RC) paradigm—also dubbed *single step* design—in contrast to the traditional direct communication, more colorfully described as "Fire and Forget." A routine that performs reverse communication requests the caller to perform an action, instead of doing it itself.

In the context of WORHP, a reverse communication interface does not evaluate the user-defined functions that make up the problem formulation, but instead instructs the user to do so. Technically, the solver *returns* repeatedly to the calling program, instructing it to update a function or derivative value, print an iteration output, or to perform no action at all. The calling program will then flag that it has performed all requested actions, and either call the solver again to continue, or terminate the optimization process. The solver thus "outsources" the major iteration loop to the caller.

The two most important benefits of using reverse communication are the facts that users are not restricted to provide their functions in a solver-specific way[3] and

[3]In fact, the caller does not even need to provide the problem functions as routines in the technical sense.

that the caller can monitor and influence the solver between any two major iterations.[4]

Interactive Mode: An experimental extension of the RC is the *interactive mode*. This allows the user to pause the solver and presents a rudimentary terminal which allows inspection and manipulation of the solver data and parameters. To the best of the authors' knowledge, this makes WORHP the first NLP solver that allows its users to interact with it in real-time.

Data Management and XML Facilities: In contrast to most NLP solvers, or most mathematical software in general, WORHP does not keep any internal state, but manages its data in rich data structures that are used to communicate with the caller (through reverse communication, as outlined above). This enables partial or full serialization of the solver state, which is implemented through XML files. This solver-state serialization enables the so-called *Hotstart*, i.e., a facility to freeze the current progress of the solver, save it in an XML file, and either continue later, or even on another machine. Due to the fact that WORHP has no internal state, this Hotstart behaves as if the solver had not been interrupted, i.e., it aims to achieve bit-precise correspondence between uninterrupted and Hotstarted optimization runs. This solver feature can be used for logging or check-pointing, which is especially interesting for long-running optimization problems. The generated log files can furthermore be used for data exchange, export, documentation, or analysis tools, since XML is an ubiquitous standard.

Weak-Active Set: Textbook SQP methods use an *active set* to handle inequality constraints. The active set (AS) method tries to determine the set J of active indices, in analogy to Eq. (AS). Problem (NLP) is then replaced by

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & F(x) \\ \text{subject to} \quad & G_i(x) = 0, \quad i \in J \subset \{1, \dots, M\}, \end{aligned} \tag{NLP$_{\text{AS}}$}$$

i.e., the inactive inequality constraints are identified and dropped from the problem. The task of identifying the active set $J(x^{[k+1]})$ is nontrivial, since this set may change between iterations (although it stabilizes when the SQP method converges to a solution), and because the solver essentially has to make a prediction for the active set at the *next* iterate $x^{[k+1]}$ that is computed using $J(x^{[k+1]})$. Since G is nonlinear, the prediction can be inaccurate; thus, an iterative update procedure is needed, with a worst-case performance of $O(2^M)$.

While interior-point methods do not suffer from this complication, introducing the slack variables increases the problem size; this effect is strongest if the problem has many inequalities, an effect that is very pronounced in discretized optimal control problems with path constraints. To take advantage of the benefits of both methods, the *weak-active set* method uses a conservative estimate to remove

[4] WORHP's concept of splitting major iterations into so-called stages allows still finer resolution.

inequality constraints, and thus reduce the size of the quadratic subproblems (QP). This method does not share the difficulties of its strict precursor, since the estimate of the active set may be fuzzy, and any "suspicious" inequality constraints can be left untouched, since there is no need to transform them into equality constraints.

4.2.2.3 Termination Criteria

For testing an iterate $x^{[k]}$, the first-order necessary optimality conditions (KKT) have to be evaluated. First-order derivatives are required for this test. Using the notation from Eqs. (NLP) and (AS), the iterate $x^{[k]}$ is said to be optimal if and only if it satisfies

$$\nabla_x F(x^{[k]}) + \lambda^{[k]T} \nabla_x G(x^{[k]}) \leq \varepsilon_{\text{opti},} \tag{4.6}$$

$$-\lambda_i^{[k]} \leq \varepsilon_{\text{comp}}, \; i \in I(x^{[k]}), \tag{4.7}$$

$$|\lambda_j^{[k]}| \leq \varepsilon_{\text{comp}}, \; j \notin J(x^{[k]}). \tag{4.8}$$

$$\begin{aligned} |G_i(x^{[k]})| &\leq \varepsilon_{\text{feas}}, \; i = 1, \ldots, M_e, \\ G_j(x^{[k]}) &\leq \varepsilon_{\text{feas}}, \; j = M_e + 1, \ldots, M, \end{aligned} \tag{4.9}$$

with user-specified tolerances $\varepsilon_{\text{opti}}$ for optimality, $\varepsilon_{\text{comp}}$ for complementarity, and $\varepsilon_{\text{feas}}$ for feasibility, which are commonly chosen between 10^{-6} and 10^{-8}.

WORHP also supports a scaled version of conditions (4.6)–(4.9). These scaled conditions are motivated by the idea that the meaning of conditions (4.6) through (4.8) for the underlying optimization problem is difficult to interpret for users, who are actually interested in a feasible point that satisfies

$$|F(x^*) - F(x^{[k]})| \leq \varepsilon.$$

In addition to the scaled KKT conditions, special low-pass-filter techniques account for numerical inaccuracies, large multipliers, and generally possible bad scaling in the optimization model, and terminate the optimization if no more progress is possible through numerical means.

4.3 Numerical Examples

We will present two OCPs in this section, with the aim to demonstrate WORHP's capabilities to solve large-scale nonlinear optimization problems. The focus of this section is on showcasing the solver performance with respect to problem size, using two uncomplicated models.

The examples will be OCPs of the form

$$\begin{aligned}
\text{Minimize} \qquad & F(y,u) = \varphi(y(t_0), y(t_f)) + \int_{t_0}^{t_f} \bar{f}_0(y(t),u(t))\,dt \\
\text{subject to} \qquad & \dot{y}(t) = \bar{f}(y(t),u(t)), \\
& y(t_0) = y_0, \quad \Psi(y(t_f)) = 0, \\
& C(y(t),u(t)) \le 0, \quad t \in [t_0, t_f].
\end{aligned} \tag{OCP}$$

The functions $\varphi : \mathbb{R}^{2n} \to \mathbb{R}, \bar{f}_0 : \mathbb{R}^{n+m} \to \mathbb{R}, \bar{f} : \mathbb{R}^{n+m} \to \mathbb{R}^n,\ \Psi : \mathbb{R}^n \to \mathbb{R}^r, 0 \le r \le n,$ and $C : \mathbb{R}^{n+m} \to \mathbb{R}^k$ are assumed to be sufficiently smooth on appropriate open sets. The admissible class of control functions is that of piecewise continuous controls. The final time t_f is either fixed or free.

OCPs can be understood as infinite-dimensional optimization problems, since the states y and the controls u have to be optimal for every point of time $t \in [t_0, t_f]$.

Choose a natural number $\bar{N}$ and let $\tau_i \in [t_0, t_f]$, $i = 1, \ldots, \bar{N}$, be mesh or grid points with

$$t_0 = \tau_1 < \ldots < \tau_{\bar{N}-1} < \tau_{\bar{N}} = t_f. \tag{4.10}$$

For notational simplicity we assume that the discretization in Eq. (4.10) is equidistant:

$$h := \frac{t_f - t_0}{\bar{N} - 1}, \qquad \tau_i = t_0 + (i-1)\cdot h, \quad i = 1, \ldots, \bar{N}. \tag{4.11}$$

Let the vectors $y^i \in \mathbb{R}^n$ and $u^i \in \mathbb{R}^m$, $i = 1, \ldots, \bar{N}$, be the approximations of the state variable $y(\tau_i)$, resp. control variables $u(\tau_i)$ at the grid points. Then Euler's approximation applied to the differential equation in Eq. (OCP) yields

$$y^{i+1} = y^i + h \cdot \bar{f}(y^i, u^i), \quad i = 1, \ldots, \bar{N} - 1, \tag{4.12}$$

or alternatively the trapezoidal method

$$y^{i+1} = y^i + \tfrac{h}{2} \cdot \left(\bar{f}(y^i, u^i) + \bar{f}(y^{i+1}, u^{i+1})\right), \quad i = 1, \ldots, \bar{N} - 1. \tag{4.13}$$

Choosing the optimization variable

$$x := (y^1, \ldots, y^{\bar{N}}, u^1, \ldots, u^{\bar{N}}) \in \mathbb{R}^N, \qquad N := (n+m)\bar{N}, \tag{4.14}$$

the optimal control problem (OCP) is replaced by the following *discretized control problem* **DOC** in the form of a nonlinear programming problem with equality and inequality constraints:

NLP-problem **DOC:**

$$\text{Minimize} \qquad F(x) = \varphi(y^1, y^{\bar{N}}) + h \sum_{i=1}^{\bar{N}-1} \bar{f}_0(y^i, u^i) \tag{4.15}$$

$$\begin{aligned} \text{subject to } -y^{i+1} + y^i + h \cdot \bar{f}(y^i, u^i) &= 0, \quad i = 1, \ldots, \bar{N}-1, \\ y_0 - y^1 &= 0, \\ \Psi(y^{\bar{N}}) &= 0, \\ C(y^i, u^i) &\leq 0, \quad i = 1, \ldots, \bar{N}. \end{aligned} \tag{4.16}$$

A *free* final time t_f can be handled as an additional optimization variable in Eq. (4.14). When using trapezoidal discretization instead of Euler's approximation, Eq. (4.13) is used in Eq. (4.16) in place of Eq. (4.12). The nonlinear optimization problem **DOC** has a large number of optimization variables N and constraints $M := (n+k)\bar{N} + r$. Nevertheless, it can be solved in an efficient way by taking into account the sparse structure in the Jacobian of the constraints and the Hessian of the associated Lagrangian.

Let $M_e = n\bar{N} + r$ be the number of *equality* constraints and $M_i = k\bar{N}$ the number of *inequality* constraints in Eq. (4.16). Furthermore, let $G(x) = (G_1(x), \ldots, G_M(x))$ denote the collection of functions defining the equality and inequality constraints (4.16). Then the nonlinear programming problem **DOC** has the form Eq. (NLP).

Compare Büskens and Maurer [4] for details on direct methods for solving OCPs.

The numerical examples presented hereinafter were solved on an Intel Q6600 system, i.e., a 2.4 GHz quad-core, running Linux.

4.3.1 Low-Thrust Planar MEO–GEO Transfer

This case describes a simple planar transfer of a satellite with a low-thrust engine from a circular orbit in roughly 7,000 km height to GEO (Figs. 4.1–4.3).

The problem—posed by Kluever [12]—is modeled using the radial position y_1 (in Earth radii, hereinafter referred to as $R_\oplus$), the radial and circumferential velocity y_2 and y_3 (both in km/s), and the polar angle y_4 (in radians). The gravitational parameter for Earth is taken as $\mu_r = \frac{G}{R_\oplus} = 62.5$. The aim of this task is to find a thrust direction control $u(t)$, $0 \leq t \leq t_f$ that minimizes the final time $F(y, u) = t_f$, which can be implemented in Eq. (OCP) by $\varphi = 0$ and $\bar{f}_0 \equiv 1$, subject to

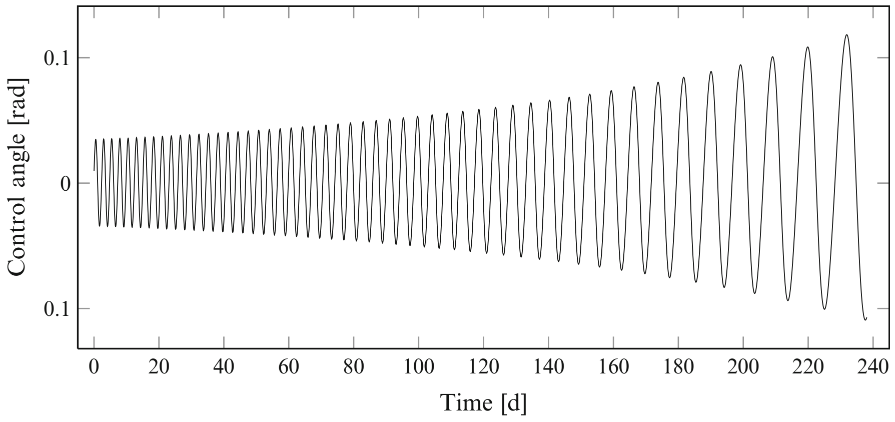


Fig. 4.1 Control angle of the planar MEO–GEO transfer plotted against flight time. The oscillatory behavior can be represented to relatively high precision due to the fine grid

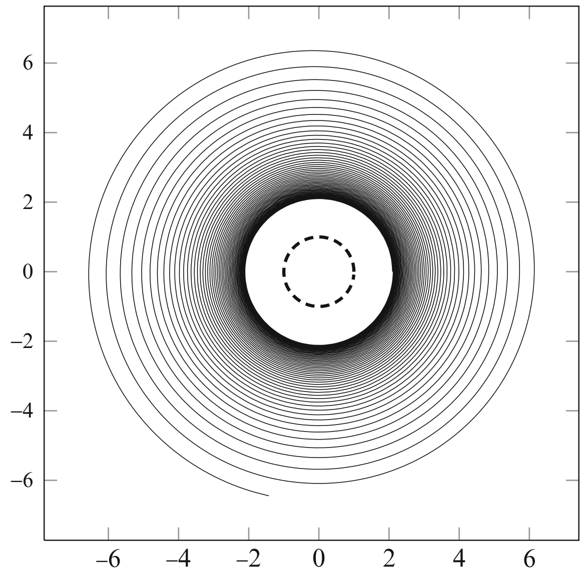


Fig. 4.2 Phase diagram of the planar MEO–GEO transfer. The Earth radius is marked by the *dashed unit circle* all distances are given in Earth radii. The very slow ascent during the initial revolutions is numerically difficult to handle for our chosen (simple) transcription method, and it requires higher precisions. Using higher-order integrators and an adaptive grid, also LEO orbits can be considered as initial condition for this planar transfer

$$
\begin{aligned}
\dot{y}_1 &= y_2,\\
\dot{y}_2 &= \frac{y_3^2}{y_1} - \frac{\mu_r}{y_1^2} + 0.01\sin(u),\\
\dot{y}_3 &= -\frac{y_2 y_3}{y_1} + 0.01\cos(u),\\
\dot{y}_4 &= \frac{y_3}{y_1},
\end{aligned}
$$

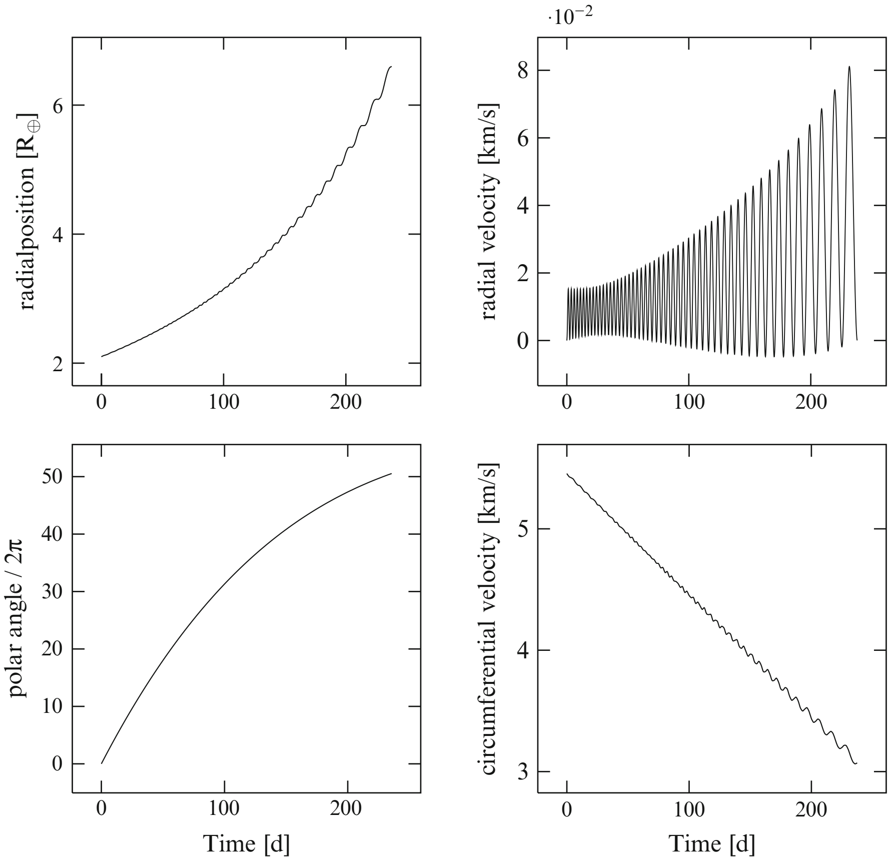


Fig. 4.3 States of the planar MEO–GEO transfer

with the initial and terminal conditions

$$\begin{aligned}
y_1(0) &= 2.1, & y_1(t_f) &= 6.6\\
y_2(0) &= 0.0, & y_2(t_f) &= 0.0\\
y_3(0) &= \sqrt{\frac{\mu_r}{y_1(0)}}, & y_3(t_f) &= \sqrt{\frac{\mu_r}{y_1(t_f)}}\\
y_4(0) &= 0.0,
\end{aligned}$$

which can be formulated with ψ, and control constraints

$$-\pi \le u \le \pi$$

which can be formulated through C in Eq. (4.16).

The OCP is of the form Eq. (OCP) and was discretized using the trapezoidal rule (4.13). AMPL [5] was used to formulate the discretized problem. The problem with n discretization points in time has $N = 5n - 6$ variables and $M = 6n - 4$ constraints. Using $n = 200{,}005$ discrete points, we arrive at 1,000,019 variables and 1,200,026 constraints.

The problem is hard to solve for NLP solvers if no good initial guess is provided. Our approach was to "bootstrap" an initial guess: The problem is first solved for a low number of discrete points and a higher initial orbit. This solution is then interpolated to a finer mesh and a lower initial orbit and used as initial guess to the solver. Iterative bootstrapping quickly leads to good initial guesses for large-scale instances of this problem.

The final solution takes roughly 230 days and needs 50 revolutions to reach GEO and was found by WORHP after 127.2 s.

4.3.2 *Range-Maximal Flight of a Hypersonic Vehicle After Engine Failure*

The "hypersonic vehicle" in this example is the Sänger reentry vehicle that was developed by MBB in the 1980s, but scrapped some years later, because it failed to be significantly cheaper than Ariane 5. The Sänger reentry vehicle is actually the upper stage of two-stage space transport vehicle, with a first stage that lifts off horizontally.

The scenario considered by Mayrhofer and Sachs [15] or Büskens and Gerdts [2, 3] is an engine ignition failure of the upper stage after separation. The vehicle does not have any thrust and is thus limited to using its initial speed and height for reaching a safe landing site, but is still able to maneuver.

For the description of the dynamic of the flight system, a mass point model with six states and two control functions is used. The Earth is assumed to be perfectly spherical and rotating. The equations of motion can then be formulated as

$$\begin{aligned}
\dot{v} &= -D(v,h;C_L)\frac{1}{m} - g(h)\sin\gamma \\
&\qquad + \omega^2\cos\Lambda(\sin\gamma\cos\Lambda - \cos\gamma\sin\chi\sin\Lambda + \cos\gamma\cos\Lambda)\frac{r(h)}{v}, \\
\dot{\gamma} &= L(v,h;C_L)\frac{\cos\mu}{mv} - \left(\frac{g(h)}{v} - \frac{v}{r(h)}\right)\cos\gamma + 2\omega\cos\chi\cos\Lambda \\
&\qquad + \omega^2\cos\Lambda(\sin\gamma\sin\chi\sin\Lambda + \cos\gamma\cos\Lambda)\frac{r(h)}{v},
\end{aligned}$$

$$\begin{aligned}
\dot{\chi} &= L(v,h;C_L)\frac{\sin\mu}{mv\cos\gamma} - \cos\gamma\cos\chi\tan\Lambda\frac{v}{r(h)} \\
&\quad + 2\omega(\sin\chi\cos\Lambda\tan\gamma - \sin\Lambda) - \omega^2\cos\Lambda\sin\Lambda\cos\chi\frac{r(h)}{v\cos\gamma}, \\
\dot{h} &= v\sin\gamma, \\
\dot{\Lambda} &= \cos\gamma\sin\chi\frac{v}{r(h)}, \\
\dot{\Theta} &= \cos\gamma\cos\chi\frac{v}{r(h)\cos\Lambda}.
\end{aligned}$$

The above mentioned functions are defined as

$$\begin{aligned}
r(h) &= r_0 + h, & g(h) &= g_0\left(\frac{r_0}{r(h)}\right)^2, \\
q(v,h) &= \frac{1}{c}\rho(h)v^2, & \rho(h) &= \rho_0 e^{-\beta h}, \\
c_D(C_L) &= c_{D_0} + kC_L^2, \\
L(v,h;C_L) &= q(v,h)FC_L, \\
D(v,h;C_l) &= q(c,h)Fc_D(C_L).
\end{aligned}$$

The constants are chosen as

$$\begin{aligned}
c &= 2, & c_{D_0} &= 0.017, & r_0 &= 6.371\cdot 10^6 \\
F &= 305, & g_0 &= 9.80665, & k &= 2, \\
\omega &= 7.270\cdot 10^{-5}, & \beta &= 1/6900, & \rho_0 &= 1.249512.
\end{aligned}$$

The state variable y consists of the velocity v in meters per second, the flight path angle γ and course angle χ in radians, the altitude h in meters, and the longitude Λ and latitude Θ in degree. The control function u consists of C_L (dimensionless lift coefficient) and μ (bank angle in radians) that are restricted by

$$0 \le C_L \le 1, \qquad 0 \le \mu \le 1,$$

which can be formulated through C in Eq. (OCP).

The mass is supposed to be constant $m = 115{,}000$ kg. The initial values are

$$\begin{pmatrix} v(0) \\ \gamma(0) \\ \chi(0) \\ h(0) \\ \Lambda(0) \\ \Theta(0) \end{pmatrix} = \begin{pmatrix} 2150.5452900 \\ 0.1520181770 \\ 2.2689279889 \\ 33900.000000 \\ 0.9268828079 \\ 0.1544927057 \end{pmatrix},$$

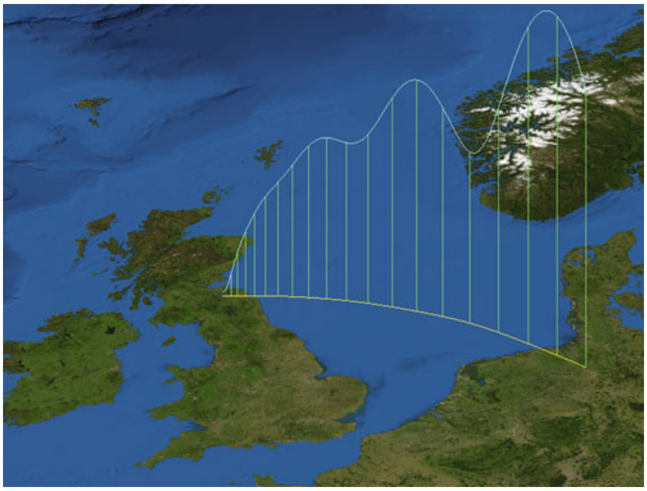


Fig. 4.4 3D plot of the trajectory

which correspond to a separation point roughly 34 km over Bremen and an initial velocity of roughly 2,150 m/s (almost Mach 7 at this height).

To maximize the choice of potential landing sites, it is necessary to find a trajectory with maximum distance to the starting point (Figs. 4.4 and 4.5):

$$F(\mu, C_L, t_f) = \left(\frac{\Lambda(t_f) - \Lambda(t_0)}{\Lambda(t_0)}\right)^2 + \left(\frac{\Theta(t_f) - \Theta(t_0)}{\Theta(t_0)}\right)^2.$$

As a final constraint a final altitude of 500 m is required:

$$\Psi\big(y(t_f)\big) = h(t_f) - 500 = 0.$$

The problem was discretized using a simple Euler discretization, which generates very sparse optimization problems. Besides its property of being more challenging and realistic than the MEO–GEO one, this problem can also serve as a solver benchmark to ascertain that WORHP essentially scales as $O(N + M)$ for large-scale sparse problems. To provide good initial guesses, we used the same bootstrap process as in Sect. 4.3.1, except for the first run with 201 discrete points, which therefore needed more NLP iterations than the successive runs (Table 4.1).

To empirically verify that WORHP scales linearly with the problem size, the dimensions were doubled (after a “warm-up” phase with 2001 discrete points). Since (usually) over 95% of the computational time is spent solving the QP sub-problem, the sum of QP iterations was recorded, and the user time divided by it. This provides a simple, yet rather significant measure of time/problem size.

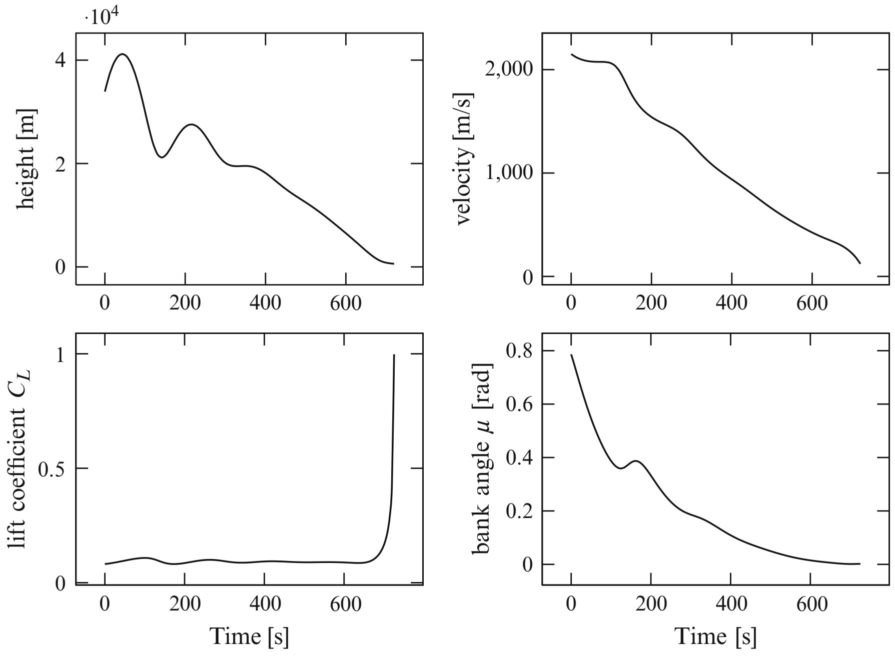


Fig. 4.5 Height, velocity, and controls of the Sänger landing after engine failure

Table 4.1 Timing and iteration results for different discretization of the thrustless flight problem

n	N	M	NLP	QP	utime[s]	utime/QP
201	1,608	2,012	151	1,652	15.8	0.01
2,001	16,008	20,012	21	136	15.9	0.12
5,001	40,008	50,012	2	36	8.0	0.22
10,001	80,008	100,012	2	22	10.3	0.47
20,001	160,008	200,012	1	12	11.3	0.94
40,001	320,008	400,012	2	20	58.2	2.91
80,001	640,008	800,012	1	11	41.2	3.74
160,001	1,280,008	1,600,012	1	11	90.3	8.21

Note that the first run was a "cold start," while all other runs were supplied with an interpolated solution of the previous one as initial guess to ensure comparability.

4.4 Summary and Conclusion

In this chapter, we have outlined the basic mathematical concepts and numerical methods required for mathematical optimization of general nonlinear problems: SQP and IP methods have been introduced as the two fundamental methods for solving nonlinear or quadratic problems; constraint relaxation and Hessian regularization are added as necessary modifications to ensure solvability of the generated subproblems; merit functions or the filter method are used together with line search to achieve global convergence and improve the convergence behavior.

We have further presented some distinctive features of WORHP that set it apart from textbook or research-focused implementations: WORHP offers advanced methods for approximation of first and second derivatives, most noticeably the group-theoretical approach for cheap finite-difference computations, and the novel structure-preserving sparse BFGS method; technical features like the XML module, the reverse communication architecture and the interactive mode are intended to allow a more flexible use of mathematical optimization in practical applications.

Two discretized optimal control problems, a MEO–GEO low-thrust transfer and a complex aerodynamic model, both exceeding one million variables and constraints, have been used to demonstrate WORHP's ability to solve relevant large-scale optimization problems, and to ascertain that WORHP's computational time scales linearly with the problem size over the whole observed range—this is a drastic improvement over dense methods, which are intrinsically bounded by cubic scaling of dense linear algebra. Sparse solvers like WORHP therefore do not have fundamental bounds to the problem dimensions, but are essentially only limited by the available hardware and the patience of their users.

Acknowledgments The authors are indebted to Florian Wolff for his support in preparing the numerical results and for various suggestions on the manuscript and to Dr. Matthias Knauer for guiding the numerical evaluation with first TransWORHP results.
Development of WORHP has been supported by BMWi (German Federal Ministry of Economics and Technology) grants 50RL0722 and 50JR0688, the TEC-EC Control Division of the European Space Agency (ESA) in the projects eNLP (GSTP-4 G603-45EC) and eNLPext (GSTP-5 G517-045EC), and Steinbeis Research Center (SFZ) Optimization and Optimal Control.

References

1. Betts, J.T.: Practical Methods for Optimal Control Using Nonlinear Programming. SIAM Press, Philadelphia, Pennsylvania (2001)
2. Büskens, C., Gerdts, M.: Numerical solution of optimal control problems with DAE systems of higher index. In: Optimalsteuerungsprobleme in der Luft- und Raumfahrt, Workshop in Greifswald des Sonderforschungsbereichs 255: Transatmospärische Flugsysteme, pp. 27–38. München (2000)
3. Büskens, C., Gerdts, M.: Emergency landing of a hypersonic flight system: A corrector iteration method for admissible real–time optimal control approximations. In: Optimalsteuerungsprobleme in der Luft- und Raumfahrt, Workshop in Greifswald des Sonderforschungsbereichs 255: Transatmospärische Flugsysteme, pp. 51–60. München (2003)
4. Büskens, C., Maurer, H.: SQP-methods for solving optimal control problems with control and state constraints: Adjoint variables, sensitivity analysis and real-time control. J. Comput. Appl. Math. **120**(1–2), 85–108 (2000)
5. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Mathematical Programming Language. Brooks/Cole Publishing Company / Cengage Learning, Florence (2002)
6. Gertz, M., Wright, J.W.: Object-oriented software for quadratic programming. ACM Trans. Math. Software **29**, 796–813 (2003)
7. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic, London (1981)

8. Gill, P.E., Murray, W., Saunders, M., Wright, M.H.: Model building and practical spects of nonlinear programming. In: Schittkowski, K. (ed.) Computational Mathematical Programming, pp. 209–47. Springer, Berlin (1985)
9. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM J. Optim. **12**, 979–1006 (1997)
10. Han, S.P.: Superlinearly convergent variable metric algorithms for general nonlinear programming problems. Math. Program. **11**(3), 263–282 (1976/77)
11. Kalmbach, P.: Effiziente Ableitungsbestimmung bei hochdimensionaler nichtlinearer Optimierung. Ph.D. thesis, Universität Bremen (2011)
12. Kluever, C.A.: Optimal feedback guidance for low-thrust orbit insertion. Optim. Contr. Appl. Meth. **16**, 155–173 (1995)
13. Levenberg, K.: A method for the solution of certain non-linear problems in least-squares. Q. Appl. Math. **2**(2), 164–168 (1944)
14. Mangasarian, O.L., Fromowitz, S.: The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. J. Math. Anal. Appl. **17**, 37–47 (1967)
15. Mayrhofer, M., Sachs, G.: Notflugbahnen eines zweistufigen Hyperschall-Flugsystems ausgehend vom Trennmanöver. In: Seminar des Sonderforschungsbereichs 255: Transatmospärische Flugsysteme, pp. 109–118. München (1996)
16. Schittkowski, K.: On the convergence of a Sequential Quadratic Programming method with an augmented Lagrangian line search function. Mathematische Operationsforschung und Statistik, Series Optimization **14**, 197–216 (1983)
17. Wilson, R.B.: A Simplicial algorithm for concave programming. Ph.D. thesis, Harvard University (1963)

Chapter 5
Global Optimization Approaches for Optimal Trajectory Planning

Andrea Cassioli, Dario Izzo, David Di Lorenzo, Marco Locatelli, and Fabio Schoen

Abstract Optimal trajectory design for interplanetary space missions is an extremely hard problem, mostly because of the very large number of local minimizers that real problems present. Despite the challenges of the task, it is possible, in the preliminary phase, to design low-cost high-energy trajectories with little or no human supervision. In many cases, the discovered paths are as cheap, or even cheaper, as the ones found by experts through lengthy and difficult processes. More interestingly, many of the tricks that experts used to design the trajectories, like, e.g., traveling along an orbit in fractional resonance with a given planet, naturally emerge from the computed solutions, despite neither the model nor the solver have been explicitly designed in order to exploit such knowledge. In this chapter we will analyze the modelling techniques that computational experiments have shown to be most successful, along with some of the algorithms that might be used to solve such problems.

Keywords Global optimization • Basin hopping • Trajectory planning

A. Cassioli • D.D. Lorenzo • F. Schoen (✉)
Università di Firenze, Via di S.Marta 3, 50139 Firenze, Italy
e-mail: cassioliandre@gmail.com; dilorenzo@dsi.unifi.it; fabio.schoen@unifi.it

D. Izzo
Advanced Concepts Team, European Space Agency, Noordwijk, The Netherlands
e-mail: dario.izzo@esa.int

M. Locatelli
Università di Parma, Viale G.P.Usberti 181/A, 43100 Parma, Italy
e-mail: locatelli@ce.unipr.it

5.1 Introduction

In this chapter we deal with the application of global optimization techniques to the design of interplanetary trajectories. This subject has received attention in the last years as it becomes increasingly evident that such a framework introduces a high level of automation in a process that is otherwise still heavily relying on expert engineering knowledge. Thanks to initiatives such as the global trajectory optimization competition (GTOC) [14] and the global trajectory optimization problems database (GTOP) [17] of the European Space Agency, the attention of communities not traditionally linked to aerospace engineering research has increased, bringing a beneficial influx of new ideas and solutions, thus advancing the field considerably. For chemically propelled spacecrafts the chemical multiple gravity assist (MGA) and the MGA with one deep space maneuver (MGA-1DSM) models have been efficiently tackled by global optimization algorithms (see, e.g., [2, 3, 6, 15, 16, 22, 23, 29, 30, 32]). It has been proven that efficient computer algorithms are able to produce, for these types of spacecrafts, competitive trajectory designs. Much less work has been published on the multiple gravity assist—low-thrust (MGA-LT) problems, mainly because they are transcribed as nonlinearly constrained rather than box-bounded constrained problems. In this chapter we will present results on some instances of the MGA-LT and the MGA-1DSM problems showing how in all cases it is possible to efficiently explore the solution space. In Sect. 5.2 we derive the mathematical model we use for the LT-MGA problem (for the MGA and MGA-1DSM problems the models used can be found in [15]). The model, inspired by the work of Sims [26], is explained here for the optimization practitioners via a series or progressively more complex problems. Eventually we will reduce the problem to maximizing a simple linear function over a feasible region defined by a set of nonlinear and nonconvex constraints. Non-convexity, directly linked in this context to planetary phasing as well as to the orbital mechanics of the spacecraft, implies that the problem exhibits many local minimizers, which makes the detection of the global ones more complicated. Indeed, the available local search techniques easily get trapped into local and not global optima so that the detection of a global minimizer requires the application of global search techniques. To be more precise, for space trajectory problems, it usually makes sense not only to search for the global solution but also for a whole set of good local minimizers, in order to have a selection of different reasonable options among which the mission designer can choose. This, however, does not change our perspective: what we need is still some global search technique leading to good local solutions, since local search techniques easily get trapped in bad minimizers. Some recent works about global optimization approaches for the LT-MGA problem are [31, 36]. In Sect. 5.3 we shortly describe three different global search techniques applied to this problem: Multistart, monotonic basin hopping (MBH) and Simulated Annealing with Adaptive Neighborhood. These, of course, do not cover all the possible approaches, but the computational experiments performed with these three methods already lead to some interesting indications about the structure of the problem and the strategies to tackle it. Two case studies of LT-MGA missions are described in Sect. 5.4, for which extensive results and analysis are provided. In Sect. 5.5 an extensive series of tests on instances of the MGA-1DSM problem are presented. Conclusions and directions for future research are discussed in Sect. 5.6.

5.2 Building a Mathematical Model for the LT-MGA Problem

In this section we derive a mathematical model for the LT-MGA problem. We start to provide a model for a simple version, and then we progressively extend the model. Before deriving any model, we first need to consider the problem of Keplerian orbital propagation. In other words, we need to compute, given a starting position and velocity, the position and velocity of a point mass subject to a central force field describing the gravitational attraction to a massive object located in the origin of the coordinate system. Although the shape of the resulting trajectory is known to be a conic, the analytical relation between position, velocity, and time is more complex and requires the solution of a transcendental equation called the Kepler's equation. Different methods exist, and a description of those is outside the scope of this chapter. Many known algorithms first compute if the orbit is elliptic or hyperbolic and then execute slightly different instructions to handle the two cases. The parabolic trajectory is usually not handled explicitly and is considered as a degenerate case. However, if the starting conditions are such that the trajectory is nearly parabolic, such methods often present numerical difficulties, and the loss of accuracy strongly affects the behavior of the solvers. Here we will briefly describe a method, which uses a reformulation of the problem, called the universal variable formulation, which allows us to treat in the same way the parabolic, hyperbolic, and elliptic cases and prevents such numerical problems from arising in the first place. For more information about the method, we refer to [5]. The algorithm is also efficient, but somewhat at the expense of clarity, as many of the variables used inside the method do not represent any obvious physical property.

Let $r_0 \in \mathbb{R}^3$ and $v_0 \in \mathbb{R}^3$ be the starting position and velocity of the orbiting body. Let μ be the gravitational parameter of the central body

$$\mu = GM,$$

where $\{G\} = 6.67384(80)\ 10^{-11}\ \mathrm{m^3kg^{-1}s^{-2}}$ is the gravitational constant and M is the mass of such body. Our goal is to compute the position $r_t \in \mathbb{R}^3$ and the velocity $v_t \in \mathbb{R}^3$ of the orbiting body after t units of time. First, we compute α as

$$\alpha = \frac{2}{\|r_o\|} - \frac{\|v_0\|^2}{\mu}.$$

The sign of α gives the shape of the trajectory, which is hyperbolic if $\alpha < 0$, parabolic if $\alpha = 0$, and elliptic if $\alpha > 0$. The parabolic case is a degenerate one; if $\alpha \neq 0$, α^{-1} is the conic semimajor axis, with a negative sign in case of the hyperbolas.

Then, we find a value of θ such that

$$\frac{r_0' v_0 \theta^2 C(\alpha\theta^2)}{\sqrt{\mu}} + (1 - \alpha\|r_0\|)\theta^3 S(\alpha\theta^2) + \|r_0\|\theta = \sqrt{\mu}t, \tag{5.1}$$

where $S(z)$ and $C(z)$ are the Stumpff functions defined as

$$S(z) = \begin{cases} \left(\sqrt{z} - \sin(\sqrt{z})\right) z^{-\frac{3}{2}} & \textit{if } z>0 \\ \left(\sinh(\sqrt{-z}) - \sqrt{-z}\right)(-z)^{-\frac{3}{2}} & \textit{if } z<0 \\ \frac{1}{6} & \textit{if } z = 0, \end{cases}$$

$$C(z) = \begin{cases} \frac{1 - \cos(\sqrt{z})}{z} & \textit{if } z>0 \\ \frac{\cosh(\sqrt{-z}) - 1}{z} & \textit{if } z<0 \\ \frac{1}{2} & \textit{if } z = 0. \end{cases}$$

Next, we compute the coefficients F and G (called the Lagrangian coefficients) as

$$F = 1 - \frac{\theta^2 C(\alpha\theta^2)}{\|r_0\|} \qquad G = t - \frac{\theta^3 S(\alpha\theta^2)}{\sqrt{\mu}},$$

and find r_t as

$$r_t = Fr_0 + Gv_0.$$

Similarly, we compute the remaining two Lagrangian coefficients F_t and G_t as

$$F_t = \sqrt{\mu}\theta \frac{\alpha\theta^2 S(\alpha\theta^2) - 1}{\|r_0\|\|r_t\|} \qquad G_t = 1 - \frac{\theta^2 C(\alpha\theta^2)}{\|r_t\|}$$

to finally express v_t as

$$v_t = F_t r_0 + G_t v_0.$$

By using the equations above, since Eq. (5.1) always admits a unique solution, we can define the propagation function $\text{Prop}_\mu : \mathbb{R}^7 \to \mathbb{R}^6$ that, given the initial conditions, propagates a body for a given period of time under the influence of a mass with gravitational parameter μ. Then, $[r_t, v_t] = \text{Prop}_\mu(r_0, v_0, t)$. Such function cannot be computed analytically because neither Eq. (5.1) can. However, Eq. (5.1) is numerically well behaved and can easily be solved to machine precision in a few steps of the Newton algorithm. Apart from the degenerate points where $r_0 = 0$ or $r_t = 0$, the function $\text{Prop}_\mu(r_0, v_0, t)$ is smooth. When such degenerate cases occur, the function is undefined. However, this function will be used to propagate a vehicle in the solar system, and if either $r_0 = 0$ or $r_t = 0$, the spacecraft must be located at the center of the Sun, so we will treat those cases with generous disregard. Finally, we note that this function can be used to propagate any body in the solar system, be it a spacecraft or a planet. We are now ready to introduce the first simple model.

5.2.1 A Single Ballistic Leg

To show a simple usage of the Prop_μ function, we can write a very simple model, in which a spacecraft travels from the Earth to Mars and is allowed to thrust the engines at departure and at arrival. We model the engine thrust as an impulsive velocity change. A typical objective function we wish to minimize is the total fuel usage or some function which is correlated with fuel consumption. Let ${r_0}^{\text{E}}$, ${v_0}^{\text{E}}$, ${r_0}^{\text{M}}$, and ${v_0}^{\text{M}}$ be the Earth and Mars position and velocities at a given epoch t_0. We introduce the variables

- $t_1 \in \mathbb{R}$, the time of departure as the number of units of time from t_0
- ${r_1}^{\text{E}}$, ${v_1}^{\text{E}}$, ${r_1}^{\text{S}}$, and ${v_1}^{\text{S}}$, all belonging to $\mathbb{R}^3$, the positions and velocities of the Earth and the spacecraft at time t_1
- $t_2 \in \mathbb{R}$, the time of arrival as the number of units of time from t_0
- ${r_2}^{\text{M}}$, ${v_2}^{\text{M}}$, ${r_2}^{\text{S}}$, and ${v_2}^{\text{S}}$, all belonging to $\mathbb{R}^3$, the positions and velocities of Mars and the spacecraft at time t_2

The model can be written as

$$\begin{aligned}
\min\ & \|v_1^{\text{S}} - v_1^{\text{E}}\| + \|v_2^{\text{S}} - v_2^{\text{M}}\| \\
& [r_1^{\text{E}}, v_1^{\text{E}}] = \text{Prop}_\mu(r_0^{\text{E}}, v_0^{\text{E}}, t_1) \\
& [r_2^{\text{M}}, v_2^{\text{M}}] = \text{Prop}_\mu(r_0^{\text{M}}, v_0^{\text{M}}, t_2) \\
& [r_2^{\text{S}}, v_2^{\text{S}}] = \text{Prop}_\mu(r_1^{\text{S}}, v_1^{\text{S}}, t_2 - t_1) \\
& r_1^{\text{S}} = r_1^{\text{E}} \\
& r_2^{\text{S}} = r_2^{\text{M}} \\
& t_2 \geq t_1.
\end{aligned} \tag{5.2}$$

The first three equations enforce the gravitational laws on the Earth, Mars, and on the spacecraft. The other two equalities constrain the spacecraft to start at the Earth and end on Mars, while the last one prevents from doing the trip backward, from Mars to the Earth. The effect of the gravitational fields of the planets on the spacecraft is neglected. We minimize the sum of all velocity changes because, through the well-known rocket equation [Tsiolkovsky formula, see Eq. (5.8)], this can be directly related to the net mass loss. We note that for a working implementation, many variables might be eliminated by substitution to reduce the problem size, in particular the position and velocity of the planets could be computed and substituted by the first two constraints. Model (5.2) can be easily extended. It is trivial to enforce simple constraints like bounds on the departure or arrival times. Additional constraints on the departure can be easily imposed. For example, the angle between $v_1{}^{\mathrm{S}}$ and the Earth rotation plane could be required to be small, as the launch systems always exploit the increased initial speed given for free by the Earth's rotation. More complex extensions are discussed in what follows.

5.2.2 *Adding Deep Space Maneuvers*

The main limitation of model (5.2) is that it forbids the engine from thrusting during the flight. To allow deep space maneuvers (modeled here as instantaneous velocity changes), additional variables must be introduced regarding the position and velocities of the spacecraft at the maneuver epoch. We will call a leg the fraction of the trajectory between two planets, and a segment a fraction of the leg, during which the spacecraft can perform a space maneuver. A leg can be composed of more than one segment. Inside a leg, between two deep space maneuvers, no force, apart from the gravitational pull of the Sun, affects the spacecraft. Rewriting the previous model to account for this case, we obtain

$$\begin{aligned}
\min\ & \|v_1^{\mathrm{S}} - v_{1^+}^{\mathrm{E}}\| + \sum\nolimits_{i=1}^{n} \|v_{i^+}^{\mathrm{S}} - v_{i^-}^{\mathrm{S}}\| + \|v_{(n+1)^-}^{\mathrm{S}} - v_{n+1}^{\mathrm{M}}\| \\
& [r_1^{\mathrm{E}}, v_1^{\mathrm{E}}] = \mathrm{Prop}_\mu(r_0^{\mathrm{E}}, v_0^{\mathrm{E}}, t_1) \\
& [r_{n+1}^{\mathrm{M}}, v_{n+1}^{\mathrm{M}}] = \mathrm{Prop}_\mu(r_0^{\mathrm{M}}, v_0^{\mathrm{M}}, t_{n+1}) \\
& [r_{i+1}^{\mathrm{S}}, v_{(i+1)^-}^{\mathrm{S}}] = \mathrm{Prop}_\mu(r_i^{\mathrm{S}}, v_{i^+}^{\mathrm{S}}, t_{i+1} - t_i) \quad \forall i \in \{1, \ldots, n\} \\
& r_1^{\mathrm{S}} = r_1^{\mathrm{E}} \\
& r_{n+1}^{\mathrm{S}} = r_{n+1}^{\mathrm{M}} \\
& t_{i+1} \geq t_i \quad \forall i \in \{1, \ldots, n\}.
\end{aligned} \tag{5.3}$$

For the i-th deep space maneuver, we introduce the following variables:

- $t_i \in \mathbb{R}$ is the time at which the deep space maneuver is performed.
- $r_i \in \mathbb{R}^3$ is the position of the spacecraft at the time the deep space maneuver is performed.
- v_i- $\in \mathbb{R}$ is the velocity of the spacecraft just before the deep space maneuver is performed.
- $v_i{}^{+} \in \mathbb{R}^3$ is the velocity of the spacecraft immediately after the deep space maneuver is performed.

5.2.3 Planetary Swingby

A further refinement of the model can be obtained by taking into account planetary swingbys, also called flybys. A planetary swingby happens when the spacecraft gets so close to a planet that the planet's gravitational pull becomes much stronger than the Sun's, and such force can be used to alter the speed direction. Since the amount of time spent in the neighborhood of a planet is so small with respect to the total time of a mission, we can assume in our model that swingbys happen instantly. At a given time t, let r^{P} and v^{P} be the planet position and velocity, r^{S} be the spacecraft position, v_{-}^{S} be the spacecraft velocity just before the swingby, and v_{+}^{S} be the spacecraft velocity just after the swingby. The following constraints must be imposed:

$$r^{\mathrm{P}} = r^{\mathrm{S}}, \tag{5.4}$$

$$\|v^{\mathrm{P}} - v_{-}^{\mathrm{S}}\| = \|v_P - v_{+}^{\mathrm{S}}\|. \tag{5.5}$$

Equation (5.4) is obvious and forces the spacecraft and planet positions to coincide at the moment of the swingby. Equation (5.5) derives from the fact that gravity is a conservation law, and it cannot be used to gain energy with respect to the attractor body. The only reason why planets can actually be used to change the velocity of the spacecraft with respect to the Sun is that planets are not stationary. Under the assumption that planets are point mass objects, such equations describe all the possible swingbys that can happen. In practice, however, many of such orbits must pass so close to the center of the planet that the spacecraft trajectory would intersect the planet sphere. Assuming that Eq. (5.5) holds, the angle θ between v_{-}^{S} and v_{+}^{S} and the minimum distance from the center of the planet $r^{\min}$ are related by the following equation [5]:

$$\theta = 2 \arcsin\left(\frac{1}{1 + r^{\min}\|v_P - v_{-}^{\mathrm{S}}\|^2/\mu^{\mathrm{P}}}\right), \tag{5.6}$$

where μ^{P} is the gravitational constant of the planet. $r^{\min}$ must obviously be greater than the radius of the planet, plus a safety margin.

5.2.4 Low-Thrust Engine

The trajectory model that is to be used to transcribe an LT-MGA trajectory optimization into a nonlinear programming problem (to be solved by global optimization methods) is crucial to the success of the overall algorithm one wants to define and apply. Criteria to be accounted for include accuracy in the description of

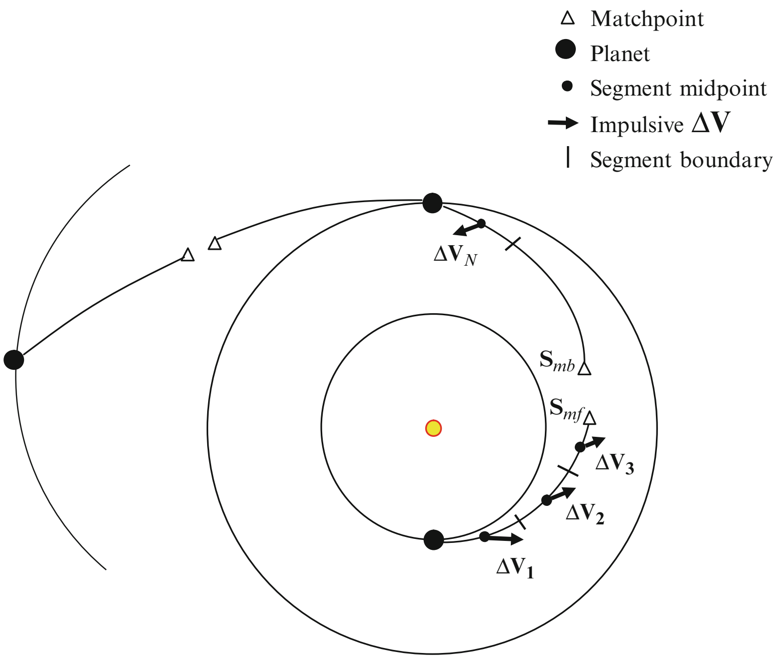


Fig. 5.1 Impulsive $\varDelta$V transcription of a low-thrust trajectory, after Sims and Flanagan [26]

the spacecraft dynamics, computational efficiency in the objective function and constraint evaluation, problem dimension, and number of nonlinear constraints produced. Bearing these issues in mind, we complete the models presented above by introducing one more constraint per segment, reaching a model similar to that proposed by Sims and Flanagan [26] and that, as will be shown later, allows for an efficient global search.

Figure 5.1 illustrates the trajectory model. Low-thrust arcs on each leg are modeled as sequences of impulsive maneuvers which provide a change of velocity that we will call $\varDelta V_i$. Such arcs are connected, using the Prop_μ function, by a conic curve. We denote the number of impulses for a given arc (which is the same as the number of segments) with N. The $\varDelta \mathrm{V}_i$ at each segment equal to $\|v^{\mathrm{S}}_{i^-} - v^{\mathrm{S}}_{i^+}\|$ should not exceed a maximum magnitude, $\varDelta V_{\max}$, where $\varDelta V_{\max}$ is the velocity change accumulated by the spacecraft when it is operated at full thrust during that segment

$$\Delta V_{\max} = (T_{\max}/m)(t_{\mathrm{f}} - t_0)/N, \tag{5.7}$$

where $T_{\max}$ is the maximum force that the low-thrust engine can apply, m is the mass of the spacecraft, t_0 and t_{f} is the initial and final time of a leg.

The spacecraft mass is propagated using the Tsiolkovsky rocket equation [28]

$$m_{i+1} = m_i e^{\left(-\Delta V_i/(g_0 I_{\mathrm{sp}})\right)}, \tag{5.8}$$

where the subscript i refers to the i-th segment, g_0 is the standard gravity (9.80665 m/s^2), and I_{sp} is the specific impulse of the low-thrust engine.

At each leg, the trajectory is propagated using the Prop_{μ} equation as seen in Sect. 5.2.2. Swingby maneuvers are modeled using the equations in Sect. 5.2.3.

Using all the elements previously introduced, we can finally write an optimization model for a low-thrust trajectory with multiple swingbys. Let L be the number of legs, and let S_l be the number of segments of the l-th leg. For every leg, the following variables are used:

- $r_l^i \in \mathbb{R}^3$, $r_l^f \in \mathbb{R}^3$ are the Cartesian positions of the spacecraft at the beginning and at the end of the leg.
- $v_l^i \in \mathbb{R}^3$, $v_l^f \in \mathbb{R}^3$ are the spacecraft velocities at the beginning and at the end of the leg.
- $m_l^i \in \mathbb{R}$, $m_l^f \in \mathbb{R}$ are the spacecraft masses at the beginning and at the end of the leg.
- t_l is the time spent in leg l.
- $\Delta v_{l,\,s} \in \mathbb{R}^3$, for every segment s in the leg, is the change in velocity provided by the spacecraft engine.
- Except for the last leg, where no swingby is executed, r_l^{swingby} is the minimum distance from the spacecraft to the center of the planet during the maneuver.

For the sake of convenience, in the model we define a function $\mathrm{Propleg}_{\mu}$ that applies, in sequence, a propagation to a whole leg for all of its segments. $\mathrm{Propleg}_{\mu}$ takes as arguments the starting position, velocity, and mass, the total time t of the leg, one $\Delta v \in \mathbb{R}^3$ vector for each segment, and computes the final position, velocity, and mass of the spacecraft after the leg. Starting from the initial position, for every segment we must propagate the body up to the middle, then apply a DSM and propagate for the remaining time. Let r, v, m denote the current position, velocity, and mass. Those variables are initially set to the starting values. For every segment, whose duration is assumed to be constant and equal to t/S_l, we do the following:

1. We propagate the spacecraft up to half the segment time using $[r, v] = \mathrm{Prop}_{\mu}(r, v, t/(2S_l))$. The mass m is kept constant.
2. We instantaneously apply the DSM. This does not change the spacecraft position r. The velocity is modified by $v = v + \Delta v_i$, where Δv_i is the change in velocity for segment i. The mass is updated using Eq. (5.8) to account for the spent fuel.
3. We propagate the spacecraft to the end of the segment, using again equation $[r, v] = \mathrm{Prop}_{\mu}(r, v, t/(2S_l))$.

Finally, the updated values of r, v, m are returned. As an implementation related note, the Prop_{μ} function satisfies the following property:

Property 1. Let each of $r_1, r_2, r_3, v_1, v_2, v_3$ belong to $\mathbb{R}^3$, and let $t_1, t_2 \in \mathbb{R}^+$. Then, if

$$[r_2, v_2] = \mathrm{Prop}_{\mu}(r_1, v_1, t_1)$$

$$[r_3, v_3] = \mathrm{Prop}_{\mu}(r_2, v_2, t_2),$$

it holds that

$$[r_3, v_3] = \mathrm{Prop}_\mu(r_1, v_1, t_1 + t_2),$$

which obviously states that propagating a body for a time equal to $t_1 + t_2$ is equivalent to propagating the same body twice, first for t_1 and then for t_2 units of time. Such property can be used to reduce the number of times the Prop_μ function is computed from $2S_l$ to $S_l + 1$ times, by computing simultaneously the last half of every segment with the first half of the following one.

Note that the Propleg function does not limit the thrust of the spacecraft. For low-thrust engines this is a very unrealistic assumption, so the following additional constraint must be imposed for every segment:

$$\|\Delta v_i\| \leq \frac{T_{\max} t}{m_i N}, \tag{5.9}$$

where $T_{\max}$ is the maximum power of the engine, t is the total time of the leg, m_i is the current mass of the spacecraft during the i-th segment, and N is the number of segments for the leg. Such constraints are cheaper to compute while executing the Propleg function, as this avoids to evaluate twice the values of m_i.

Our objective is to minimize the total fuel consumption or, equivalently, to maximize the final mass of the spacecraft:

$$\max m_f \tag{5.10}$$

$$\begin{gathered}\text{subject to, for every leg } l\\ [r_l^f, v_l^f, m_l^f] = \mathrm{Propleg}_\mu(r_l^i, v_l^i, m_l^i, t_l, \Delta v_{l,1}, \dots, \Delta v_{l,S_l})\end{gathered} \tag{5.11}$$

$$\|\Delta v_{l,s}\| \leq \frac{T_{\max} t_l}{m_{l,s} N} \qquad \forall s \in \{1, \dots S_l\} \tag{5.12}$$

$$r_{l+1}^i = r_l^f = r_l^{\mathrm{P}} \tag{5.13}$$

$$m_{l+1}^i = m_l^f \tag{5.14}$$

$$\|v_l^f - v_l^{\mathrm{P}}\| = \|v_{l+1}^i - v_l^{\mathrm{P}}\| \tag{5.15}$$

$$\arccos\left(\frac{(v_l^f - v_l^{\mathrm{P}})'(v_{l+1}^i - v_l^{\mathrm{P}})}{\|v_l^f - v_l^{\mathrm{P}}\|\|v_{l+1}^i - v_l^{\mathrm{P}}\|}\right) = 2 \arcsin\left(\frac{1}{1 + r_l^{\mathrm{swingby}}\|v_l^{\mathrm{P}} - v_l^f\|^2/\mu_l^{\mathrm{P}}}\right) \tag{5.16}$$

$$r_l^{\mathrm{swingby}} \geq r_l^{\min}. \tag{5.17}$$

Equation (5.11) imposes the trajectory constraints from the beginning to the end of every leg, Eq. (5.12) forbids the engine to thrust more than what its thrust capacity allows. Equations (5.13) and (5.14) simply state that at every swingby the position of the spacecraft must coincide with the one of the planet and that the spacecraft mass is not affected by the maneuver. Equations (5.15)–(5.17) put some constraints on the swingby, and can be disregarded for the last leg, if no swingby is performed. The model is usually extended including specific constraints like, for example, bounds on the time of flight or bounds on the spacecraft starting velocity relative to the Earth's velocity.

5.3 Some Global Optimization Algorithms

The proposed transcription of the LT-MGA problem is a continuous, constrained, nonlinear optimization problem. Since such class of problems usually present local minimizers that are not global, they are often unsolvable using only local optimization algorithms. Practical experience shows that this is usually the case with trajectory optimization problems, regardless of the propulsion type. Thus, local solvers must be used inside a global optimization strategy in order to achieve solutions which are as close as possible to the optimal ones. Here we describe three different approaches that have been tried on such problem. Of course, these are preliminary experiments, and more approaches could and should be tested. However, such preliminary results already allow to draw some interesting conclusions about the problem and good strategies to tackle it. Before detailing each approach, we first define some procedures that the algorithms will use:

- $\mathcal{G}()$ is a procedure that randomly generates a starting point. Ideally, we would like the point to be uniformly distributed over the feasible region, but since our problem's feasible set is of a very small size, and generating a point inside such region is a hard problem by itself, we used a procedure that uniformly generates points in a box containing the feasible region.
- $\mathcal{S}(x)$ is a procedure that, given a point x, computes a local minimizer of the objective function, taking x as an initial guess. For our tests we will use SNOPT [10, 9], which is based on sequential quadratic programming (SQP) [24].
- $Best(x, y)$ is a procedure that, given two solutions x and y, returns the best one according to a fixed rule. Since we are dealing with a constrained optimization problem, $Best(x, y)$ chooses the point with the lower constraint violation norm value. When both points are feasible, then the point with the lower objective function value is chosen instead.

5.3.1 *Multistart*

The first and simplest algorithm is called Multistart (MS), which directly optimizes the points obtained by a generator. Multistart can be described by the following pseudo code.

Although for N big enough Multistart will eventually converge to the global solution, the convergence rate might be extremely slow, in particular if the number of local (and not global) minimizers is very high. However, because of its simplicity, this algorithm can be effectively used as a baseline to compare other solvers to.

Algorithm 1: Multistart

```
1  $x^\star \leftarrow \mathcal{G}()$;
2  for $i \in \{1,\ldots,N\}$ do
3      $x \leftarrow \mathcal{G}()$;
4      $y^\star \leftarrow \mathcal{S}(x)$;
5      $x^\star \leftarrow Best(x^\star, y^\star)$;
6  end
```

5.3.2 Monotonic Basin Hopping

The second algorithm is MBH (see, e.g., [33]). In addition to the procedures previously used by Multistart, MBH needs a procedure $\mathcal{P}(x)$ that, given a point x, returns another point randomly generated in a conveniently defined neighborhood of x. Such algorithm can then be described as in Algorithm 2.

Algorithm 2: Monotonic Basin Hopping

```
1  $x_{best} \leftarrow \mathcal{G}()$;
2  for $i \in \{1,\ldots,N\}$ do
3      $x \leftarrow \mathcal{G}()$;
4      $x^\star \leftarrow \mathcal{S}(x)$;
5      while $k < MNI$ do
6          $y \leftarrow \mathcal{P}(x^\star)$;
7          $y^\star \leftarrow \mathcal{S}(y)$;
8          if $Best(y^\star, x^\star) = y^\star$ then
9              $x^\star \leftarrow y^\star$;
10         end
11     end
12     $x_{best} \leftarrow Best(x_{best}, x^\star)$;
13 end
```

In practice, after a local optimization, instead of generating a new point inside the whole feasible region like in Multistart, we restrict the generation inside a small region centered on the current best point. The procedure $\mathcal{P}$ is called a perturbation procedure and should generate points in a small neighborhood of the current point. It usually includes some random component, so that its output is a random point "close" to the current one. This might appear as a small and not particularly relevant modification with respect to Multistart, but, if $\mathcal{P}$ is properly chosen, it can really enhance the performance of Multistart even by order of magnitudes. This has been observed in different fields such as molecular conformation problems (see, e.g., [20, 33]) and packing problems (see, e.g., [1, 12]). The reason of these impressive improvements is briefly exposed here, while we refer, e.g., to [21] for a more detailed explanation. The procedure $\mathcal{P}$, together with the local search $\mathcal{S}$, allows to impose a graph structure over the set of local minimizers. Indeed, we are able to define a graph G whose node set is the set $X^\star$ of local minimizers, and such that for each $x_i, x_j \in X^\star$, a directed arc (x_i, x_j) exists if $x_j = \mathcal{S}(\mathcal{P}(x_i))$ with a strictly positive probability and $x_j = Best(x_i, x_j)$. In other words, a directed arc exists if we are able to move from x_i to x_j in a single step with a strictly positive probability. We can now reformulate the global optimization problem as a combinatorial optimization problem over the graph G. We denote by $X^{\star\star}$ the set of the local minimizers over the graph G (basically, all local minimizers with no outgoing arc). For reasonable choices of $Best$, such set certainly includes the global minimizer of the problem. It obviously holds that $X^{\star\star} \subseteq X^\star$.

Since the While loop of the algorithm is nothing but a descending local search over the graph G starting from some randomly generated initial local minimizer and stopped as soon as no improvement has been made for a number of times equal to a fixed parameter called max no improve (*MNI*, set equal to 500 during our experiments), we can reasonably believe that the outcome $x^\star$ of this loop is a member of $X^{\star\star}$. If the cardinality of $X^{\star\star}$ is small, then we might hope that a few repetitions of the While loop will finally allow to detect the member of $X^{\star\star}$ coinciding with the global minimizer. For this reason, the While loop is repeated different times, each time starting from a new local minimizer randomly generated over the whole feasible set. In conclusion, if the structure of our problem is such that:

- We are able to define a perturbation procedure $\mathcal{P}$ which generates points in a small neighborhood of the current point and, in particular, the number of local minimizers which can be generated with strictly positive probability by applying $\mathcal{P}$ followed by $\mathcal{S}$ to the current point, is small with respect to the cardinality of $X^\star$.
- The cardinality of $X^{\star\star}$ is also small with respect to the cardinality of $X^\star$.

then MBH turns out to be a quite efficient method for the solution of such problem. As already previously remarked this turns out to be the case for different kinds of problems like the molecular conformation and packing ones. In the field of space trajectory planning, the MGA and MGA-DSM problems appear to display such structure and have been efficiently tackled by MBH or by other methods inspired by MBH [2, 30]. The computational experiments in Sect. 5.4 will reveal that MBH performs quite well also when applied to the LT-MGA problem. The discussion above clarifies that the choice of the procedure $\mathcal{P}$ is of primary importance. A typical perturbation rule, rather simple and quite general, is to choose a new point inside a small box or sphere centered on the current point. Problem knowledge, if available, can be used to better tune the perturbation. As an example, the synodic period between two planets is the minimum interval of time that it takes for such two bodies to appear twice in the same position relative to each other. Then, when only two different planets are considered, we can expect that by shifting a solution in time by a period equal to the synodic period of such two planets, solutions that are similar (and maybe better) than the current one could be found. So the perturbation rule we used is composed by the following two steps:

1. For each variable x_i and its lower and upper bounds l_i and u_i, add to x_i a value uniformly chosen in the interval $[-r(u_i - l_i), r(u_i - l_i)]$ with a small given value of r.
2. With a low probability p, shift the solution in time, either forward or backward with equal probability, by a time length equal to the synodic period.

In our experiments, we used $r = 0.05$ and $p = 0.1$.

5.3.3 Simulated Annealing with Adaptive Neighborhood

Finally, simulated annealing with adaptive neighborhood (SA-AN) has been considered. The algorithm structure is similar to MBH, with some important differences in key parts of the algorithm, which are quickly described here. For a more complete description we refer, e.g., to [4, 19].

First of all, the comparison in line 8 is modified to allow for non-monotonicity. The problem is transformed to an unconstrained optimization problem by using a penalty function f to account for constraint violations. Then, $Best(x, y)$ returns x with probability 1 if $f(x) \leq f(y)$ or with probability $e^{(f(y) - f(x))/T}$ if $f(x) > f(y)$. T is the temperature parameter, which is exponentially decreased every fixed number of iterations.

Furthermore, the perturbation procedure $\mathcal{P}$ is adaptive, meaning that the radius r of the perturbation is adjusted at every iteration. Such r value is increased each time the generated point is accepted and decreased each time the point is refused.

Finally, the local optimization $\mathcal{S}$ is not executed at each step of the algorithm, but just once at the end, starting from the point returned by SA-AN.

In our tests, the temperature parameter T starts at 10 and is multiplied by 0.8 every 100 iterations. The starting perturbation radius r^0 is equal for each variable to 0.05 times the width of the box, and every K iterations a new value r^{k+1} is obtained from r^k using the following rule. Let ρ be the number of accepted steps in the last K iterations divided by K. Then,

$$r^{k+1} = \begin{cases} r^k(1 + \rho - 0.62) & \text{if } \rho > 0.6 \\ r^k 1 + \dfrac{0.4 - \rho}{2} & \text{if } \rho < 0.4 \\ r^k & \text{otherwise.} \end{cases}$$

We used $K = 10$, and the maximum number of iterations was set equal to 2,000.

5.3.4 *On the Results Representation*

The huge amount of tests performed required the use of a compact representation in order to visualize and compare at glance the algorithms' behavior.

For each algorithm we plot a function $f(t)$, $t > 0$ that represents the percentage of times in which the final value reported by the algorithm was within $t\%$ of the currently known putative optimum. In other words, let $f_1, f_2, \ldots, f_{\mathtt{NRuns}}$ the outcome of each algorithm execution and $f^\star$ an estimation of the global minimum[1]. Then,

$$f(t) = \frac{|\{i \in 1..\text{NRuns} : f_i \leq f^\star \times (1 + t/100)\}|}{\text{NRuns}}.$$

For instance, $f(0)$ indicates the percentage of times (if any) the method obtained the (putative) global optimal value $f^\star$, while $f(100)$ the fraction of runs giving a value which is at most twice the optimum, i.e., no more than 100% worse and so on. This way, it is quite easy to read from the figure which algorithm gave the best approximation to the optimum and which was capable of producing a larger quantity of good results. These graphical representations are closely related to performance profiles (see [7] for details), from which they differ only by the fact that here we compare different independent runs of some algorithms on a single problem, while performance profiles are usually employed to show the behavior of single runs of different methods on different test problems.

5.4 Experiments on LT-MGA Problem Instances

In this section we consider two case studies of LT-MGA problems, a mission to Jupiter and a mission to Mercury, and perform some experiments with the three algorithms previously introduced. The relative merits of the algorithms are discussed, and the solutions found are briefly discussed.

5.4.1 *Nuclear-Electric Propulsion Mission to Jupiter*

We demonstrate the application of our global optimization framework to perform preliminary design of trajectories for a mission that employs nuclear-electric propulsion (see Table 5.1). The spacecraft is assumed to have a thruster with a constant maximum thrust and a constant specific impulse, with similar hardware parameters to the Jupiter Icy Moons Orbiter [11, 25, 34, 35, 37] (a canceled mission

[1] In fact, we use the putative global minimum in order to define $f^\star$.

Table 5.1 Parameters for a nuclear-electric propulsion mission

Parameters	Values
Initial mass of the spacecraft	20, 000 kg
Maximum thrust	2.26 N
Specific impulse	6, 000 s
Launch date	Jan. 1, 2020–Jan. 1, 2030
Launch V_∞	$\leq$ 2.0 km/s
Maximum time of flight	10 years
Minimum flyby radius	7, 000 km

Table 5.2 Algorithm statistics for the E–E–J (Earth–Earth–Jupiter) mission

Algorithm	Basin hopping	Simulated annealing	Multistart
Best (kg)	17, 102	17, 019	16, 961
Worst (kg)	10, 913	11, 924	11, 900
Mean (kg)	16, 235	16, 302	15, 930
Mean best 10 (kg)	17, 039	16, 912	16, 715
Std (kg)	1, 345	1, 160	1, 320
No. of sol. above 95% best (16, 246 kg)	137	19	11
No. of sol. above 90% best (15, 391 kg)	155	22	14
Total no. of solutions	183	24	18

originally proposed by NASA in 2003). In our example scenario, we consider a planetary encounter sequences of Earth–Earth–Jupiter (i.e., one Earth flyby) which rendezvous at Jupiter. We use ten segments ($N = 10$) for each leg, and the dimension of this problem (i.e., the number of optimization variables) is 75 with 35 nonlinear constraints. We first let the global optimizers run for a fixed time (~5 h for each algorithm) on a PC (AMD Turion@2.1 GHz with 3 GB of RAM) which produced hundreds of trajectories. Then, we selected solutions for which the norm of the constraint violation vector was less than 10^{-6}. Table 5.2 summarizes the results found by the three global optimizers. We notice that many solutions found by SA-AN and Multistart fail to converge, and therefore the number of solutions is less than MBH. Figure 5.2 plots the comparison of the three algorithms. The results show that local searches started from randomly generated points, as in Multistart, or from the final points returned by an execution of SA-AN, often fail to reach the feasible region. This means that the nonlinear constraints defining the feasible region are rather complicated, and even the detection of feasible solutions is a hard task. According to the experiments, MBH is not only able to provide good local minimizers but has a considerably better capability of generating feasible solutions.

Figure 5.3 plots the x-y projection of a trajectory found by MBH with the highest final mass. On the plot, the solid and dash curves represent thrusting and coasting segments, respectively, while a ΔV is shown as an arrow in the midpoint of a segment. We will call by V_∞ the speed that the spacecraft has with respect to a

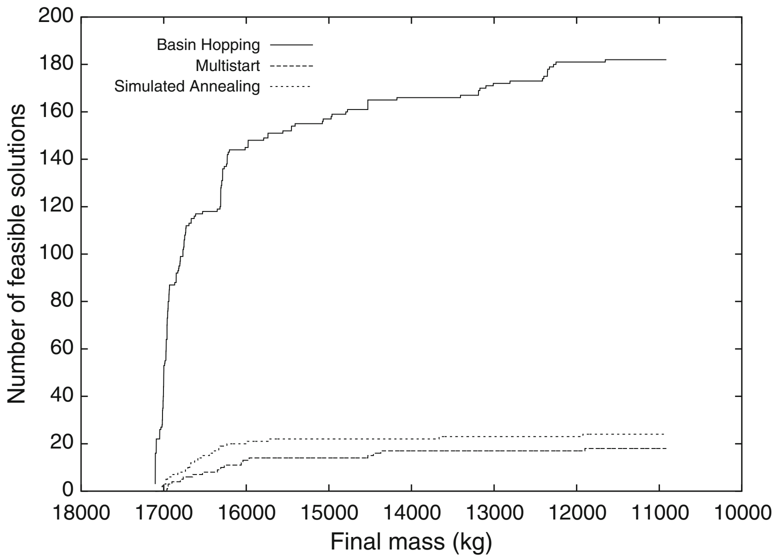


Fig. 5.2 Cumulative number of solutions for the E–E–J mission

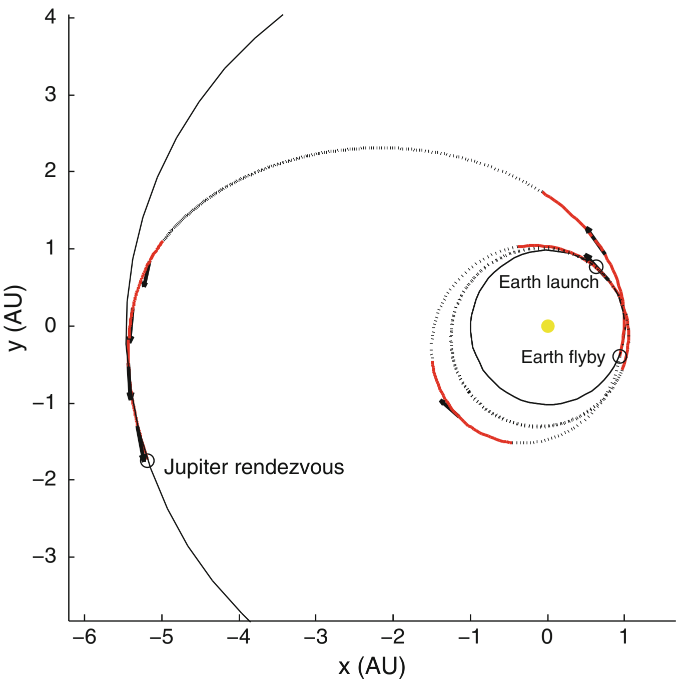


Fig. 5.3 Trajectory plot of an E–E–J rendezvous mission. AU: Astronomical Units

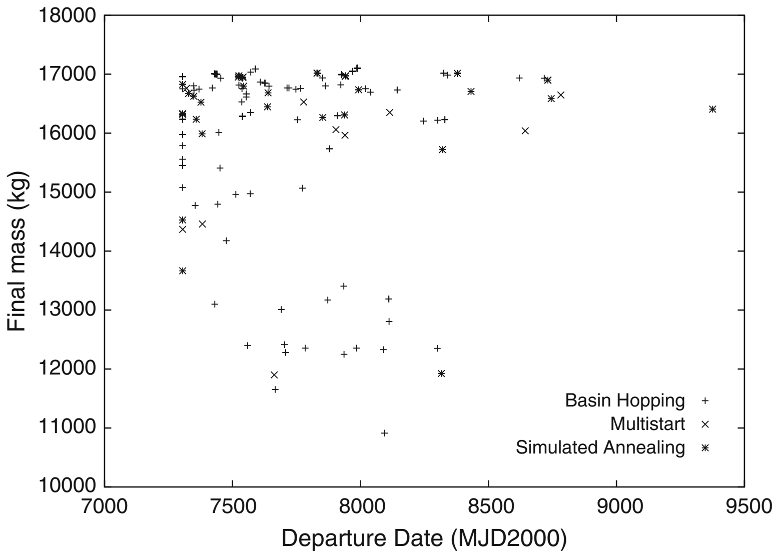


Fig. 5.4 E–E–J solutions with various launch dates

given planet, ideally measured when the body is so far away that the planet itself can no longer affect the speed of the body. In this example, the spacecraft leaves the Earth on November 12, 2021 with a V_∞ of 2 km/s. It enters a 4:3 resonance orbit with a period of $\sim$ 1.3 years and goes around the Sun for 3 revs, before it flybys the Earth after 3.8 years with an increased V_∞ of 8.5 km/s. After the gravity assist at the Earth, its aphelion increases for the transfer to Jupiter. After 7.4 years of interplanetary flight, the spacecraft rendezvous at Jupiter on March 24, 2029 with a final mass of 17,102 kg.

Besides the value of the objective function (final mass), it is also interesting from a mission design point of view that the optimization process is able to find trajectories that launch on different dates. In our example in Fig. 5.4, the difference in the final mass is less than 200 kg (or 1% of the initial mass) for most launch periods. The 1% penalty of the final mass gives flexibility to the mission designer to choose a different date in case there is a change in the mission. The process is also able to locate various locally optimal trajectory families, which is interesting from an astrodynamics point of view. From Fig. 5.5, we note that the "clusters" of solutions belong to different Earth–Earth resonance transfer orbit. For example, 1:1 resonance with flight time $\sim$ 400 days, 2:3 resonance with flight time $\sim$ 800 days, and 3:2 resonance with flight time $\sim$ 1,100 days. Unlike the case in the ballistic transfer, in the low-thrust transfer case, the Earth–Earth flight time does not exactly equal to some integer multiple of Earth's orbital period, and the spacecraft does not encounter the Earth at the same position from launch. However, the mechanism in the low-thrust case is similar to the chemical case [27], where the V_∞ at the second Earth encounter is increased through some small maneuvers.

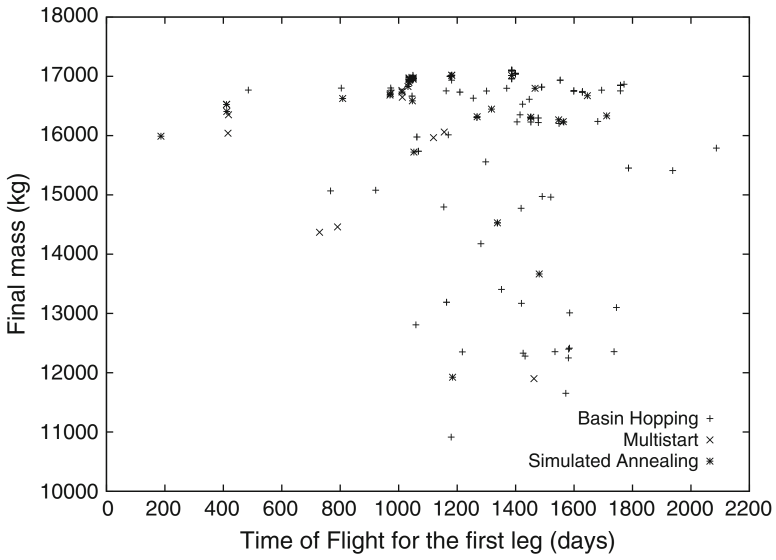


Fig. 5.5 E–E–J solutions with various Earth–Earth transfer times

Table 5.3 Parameters for a mission to Mercury

Parameters	Values
Initial mass of the spacecraft	1, 300 kg
Maximum thrust	0. 34 N
Specific impulse	3, 200 s
Launch date	Aug. 1, 2009–Apr. 27, 2012
Launch V_∞	$\leq$ 1. 925 km/s
Arrival V_∞	$\leq$ 0. 5017 km/s
Arrival date	No later than Nov. 26, 2021
Minimum flyby radius	1.1 R_{planet}

5.4.2 *Mission to Mercury*

Our second test case is a mission to Mercury, inspired from the BepiColombo mission [38]. The planetary encounter sequence is Earth–Venus–Venus–Mercury–Mercury–Mercury (EVVMMM), and the mission parameters are given in Table 5.3. The mission parameters have been subject to many changes since the original concept. Here we consider the mission as was described in a 2002 report from the European Space Operations Centre [18]. Unlike the solar electric propulsion (SEP) used in the BepiColombo mission, we do not model the SEP engine in our test but rather assume the thrust and specific impulse to be constant. In comparison with the first case, the number of flybys increases from 1 to 4, and the dimension of the problem increases to 222 with 99 nonlinear constraints. Also, we do not strictly

Table 5.4 Algorithm statistics for a mission to Mercury

Algorithm	Basin hopping	Simulated annealing	Multistart
Best (kg)	1, 064	988	1, 052
Worst (kg)	101	111	100
Mean (kg)	724	522	530
Mean best 10 (kg)	1, 051	917	1, 032
Std (kg)	277	253	261
No. of sol. above 95% best (1, 011 kg)	35	0	1
No. of sol. above 90% best (958 kg)	88	14	31
Total no. of solutions	345	94	756

require a rendezvous with Mercury with a V_∞ of zero, but the spacecraft is allowed to have a 0.5 km/s speed difference with respect to the planet.

Because of the added complexity, we allowed the global solvers to run for a longer time (∼48 h), and we used a more performant machine (Intel Xeon@2.66 GHz with 8 GB of RAM). The test results are summarized in Table 5.4. MBH found more than three hundred locally optimal solutions, of which more than 80 are above 90% of the best solution. In this case Multistart is able to detect more feasible solutions than MBH found (more than double the number of solutions than MBH), but with fewer high final mass trajectories (e.g., only 30 solutions are above the best 90%). SA-AN only found about one hundred locally optimal solutions and with less good quality solutions than the other two algorithms.

From Figs. 5.6 and 5.7, we note that the first one-third of the solutions have a high final mass (over 900 kg), and it covers many launch opportunities within the search space. If we do not restrict the analysis to the solutions that are also locally optimal, then the range of launch opportunities increases even more. The trajectory of the "best" solution (with the highest final mass) is plotted in Fig. 5.8. Here the spacecraft leaves Earth on April 13, 2010, performs two flybys at Venus to lower its orbit, then encounters Mercury twice to lower its V_∞ to 0.5 km/s before it arrives on Sep 5, 2015. Comparing such trajectory with the one described in the BepiColombo early mission planning stage, we note that our results are similar to those described in [18] (as cited in [8]) with regard to the overall shape of the optimal trajectory, which includes the planetary resonances, the times of flight, and the engine thrust periods, but have been obtained using a completely automated process.

5.5 Computational Experiments on MGA-1DSM Problem Instances

In this section we briefly summarize a large set of numerical experiments on MGA-1DSM problem instances whose details can be found in [2]. Such MGA-1DSM model cannot be used for low-thrust trajectories but fits more nicely

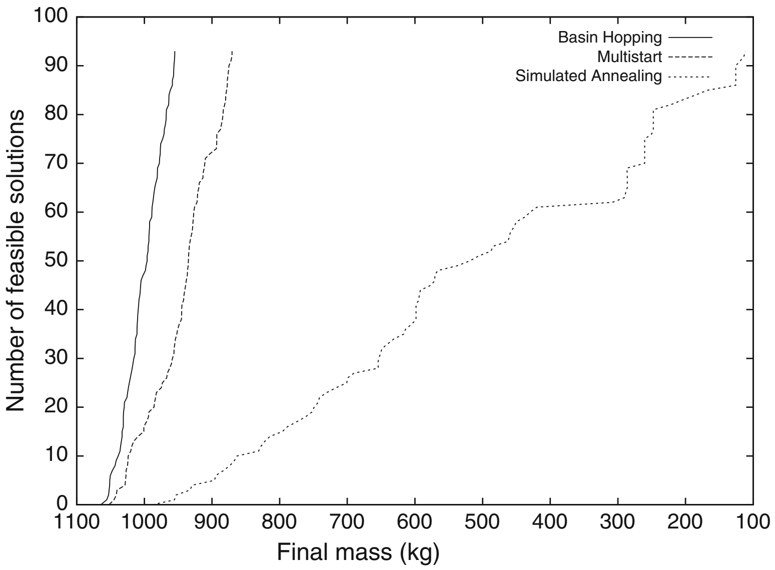


Fig. 5.6 Cumulative number of solutions for the EVVMMM mission

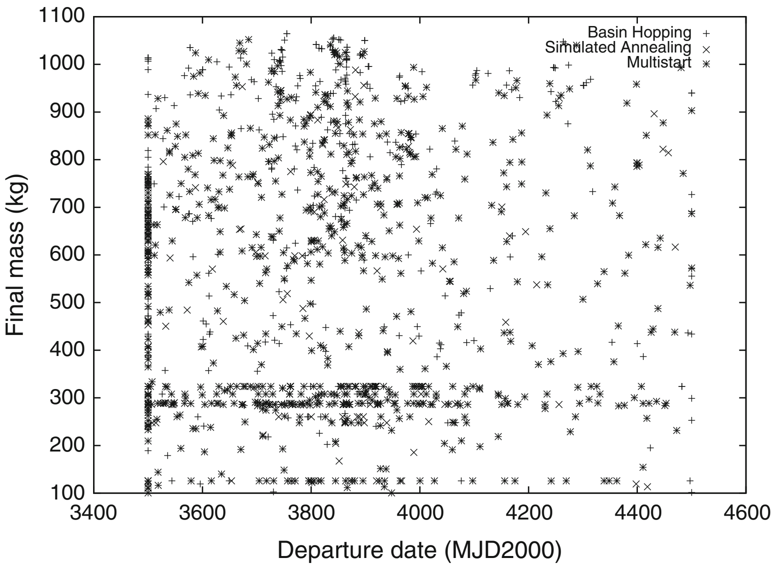


Fig. 5.7 EVVMMM solutions with various launch dates

with chemical ones. In the chemical transfers considered we assume that only one DSM is used in each trajectory leg (hence the name MGA-1DSM). This allows the orbital propagation problem to be performed using solutions to the so-called Lambert problem (see, e.g., [13] for details).

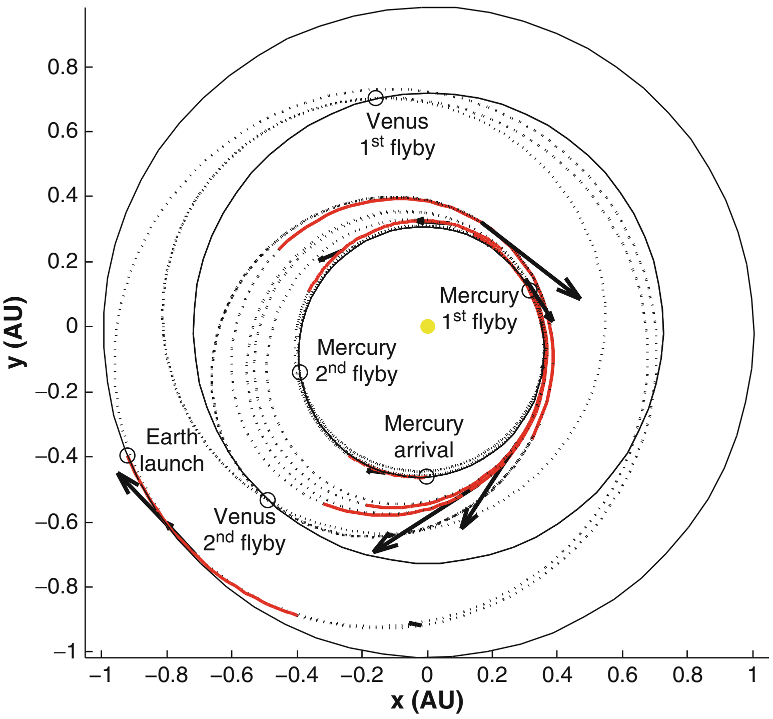


Fig. 5.8 Trajectory plot of a mission to Mercury

In order to illustrate another way the algorithms previously presented can be used, in this section we will comment on the use a so-called black-box model, in which the solvers do not have access to the gradient of the objective function and constraints.

Details about MGA-1DSM models are outside the scope of this section and can be read in [15]. Here we only assume to have a code which computes a suitable objective function, given a set of variables which describe the trajectory. We simply recall that the models are based on suitably choosing a set of dates and times relative to each mission: among the variables available for optimization, there are the starting date and the time duration of each leg. Given these quantities, a Lambert problem is solved in order to compute the trajectory as well as initial and final velocity at the extremes of each leg. Some additional variables are also included for specifying the time instant of each DSM, the pericenter radii, i.e., the minimum distance to all encountered planets, the angles between the planar trajectories at each swingby. Given these quantities, the spacecraft movement within each leg and between consecutive DSMs is completely described by the solution of a specific Lambert's problem.

Extensive numerical tests were performed on the GTOP test set (freely available from the ESA ACT web site, see [17]). The GTOP problems are provided as "black

Table 5.5 Variables for box constrained ESA MGA-1DSM problems

Name	Meaning	#
t_0	Departure time	1
V_∞	Dep. vel. modulus	1
u	Dep. vel. angle1	1
v	Dep. vel. angle2	1
T_i	Time of flight	n
η_i	Time of DSM i	n
rp_i	Pericenter radius at swingby i	$n-1$
b_i	Outc. vel. angle at swingby i	$n-1$

Table 5.6 Box constrained ESA MGA-1DSM problems

Problem name	Variables	Planet sequence
Cassini	22	E V V E J S
Messenger	18	E E V V Me
Rosetta	22	E E V E E 67P
Tandem	18	E P1 P2 P3 S

E Earth, *V* Venus, *J* Jupiter, *S* Saturn, *Me* Mercury, *M* Mars, *67P* Comet67P/Churyumov–Gerasimenko, *Pi* generic planet chosen in the set$\{E, V, M, J\}$

box" ones, so that algorithms can just obtain the objective function value, given a set of parameters. No information, e.g., on gradients, is available, nor differentiability can be assumed. Thus, they can be used for testing purpose by the scientific community in order to create a shared benchmark test suite for space trajectory problems; they are also of interest as hard test bed for derivative-free optimization algorithms.

Only MGA-1DSM problems characterized by the presence of box constraints have been tackled. In Table 5.5 the meaning of each variable is briefly summarized, while problem characteristics (number of variables and sequence of astronomical bodies visited) are listed in Table 5.6. Problem named "Tandem" is not a single one but a collection of 24 problems, each corresponding to a different sequence of planet swingbys from the Earth to Saturn.

Local searches were performed using the SNOPT SQP solver [10] with finite difference approximation for derivatives. We omit all details, except that we can control the accuracy by a step length parameter h.

All tests, if not otherwise stated, have been performed with the following stopping criteria:

- Multistart performed $\mathtt{N} = 1,000$ steps, with uniformly generated starting points.
- In Multistart BH, at each Multistart step, an MBH was executed with a stopping rule calling for stopping as soon as no improvement was observed in the last $\mathtt{MNI} = 500$ iterations.

The huge amount of data generated by our tests is not easy to be summarized, so having observed very similar behavior in all missions we tested, we prefer to present a few pairwise comparisons between different algorithmic options in

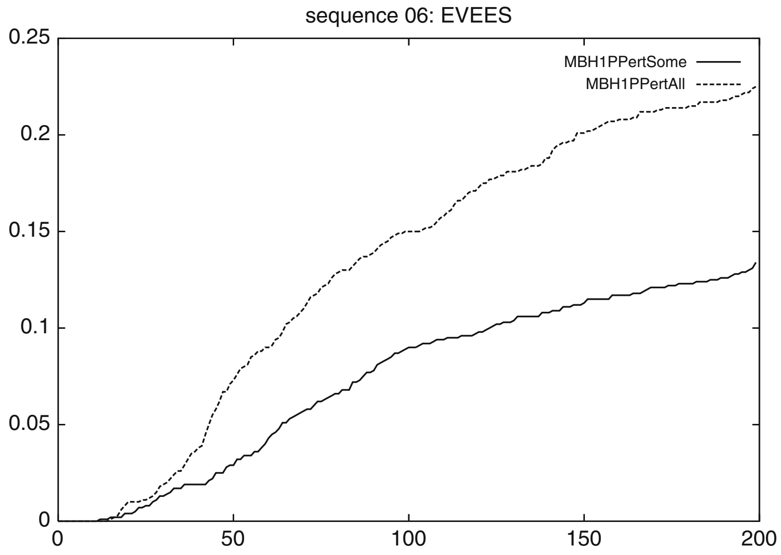


Fig. 5.9 Comparison of two different perturbations: all variables (*dotted line*) vs. just a few ones (*solid line*)

order to ease data analysis. As the principal test bed, we choose the Tandem mission (see Table 5.6). In the following figures we will represent computational profiles in particular for what concerns the mission with highest estimate of the global maximum, i.e., mission 6 (starting with Earth, with three swingbys at Venus, Earth, and Earth again). This problem is formulated as a maximization one, as the objective is a function of the final mass of the spacecraft.

The first experiments were devoted to finding a reasonably good perturbation $\mathcal{P}$ in MBH. As already remarked in Sect. 5.3.2, the choice of the perturbation is crucial to the performances of MBH.

In particular, as there seemed to be some evidence that some variables in the problem like, e.g., starting times and times of flight were in some sense easier to choose than other ones, or, at least, that, once well chosen, they were quite stable, we investigate whether perturbations involving only a few variables at a time were more successful than perturbations in which every variable is randomly displaced. At each step of MBH, the current solution was perturbed in two different ways:

- *Algorithm* `MBH1PPertSome`: between 1 and 4 coordinates were randomly chosen and uniformly perturbed in an interval of radius equal to 5% of the box containing the variable.
- *Algorithm* `MBH1PPertAll`: *every* coordinate was uniformly perturbed in an interval whose radius is 5% of the box.

In Fig. 5.9 we report the results obtained running the two versions of this method, with the graphical representation introduced at page 14. It is quite evident from Fig. 5.9 that perturbation of all coordinates is preferable.

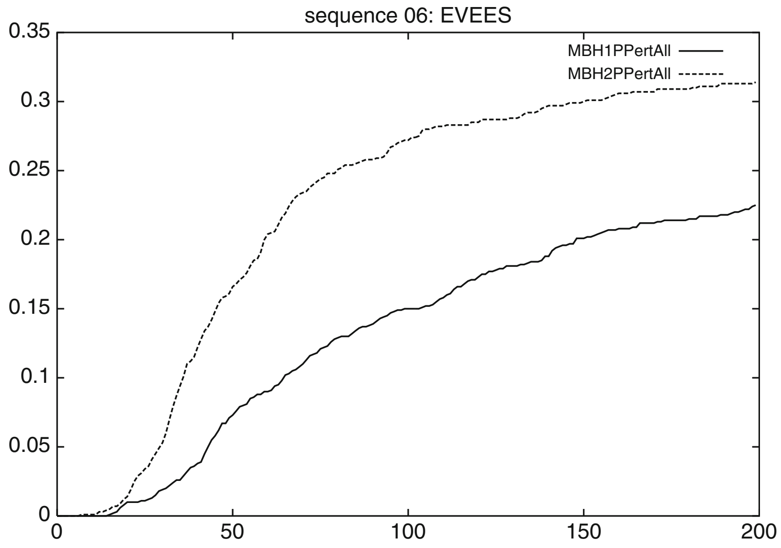


Fig. 5.10 Comparison between 1-(*solid line*) and 2-phase (*dotted line*) algorithms

The second set of experiments was aimed at checking the efficiency of a so-called "two-phase approach," a strategy that has been very successful in many different contexts like, e.g., that of molecular conformation problems studied by some of the authors of this chapter. This approach consists in performing a local search on a "modified" problem (the so-called "first phase"), and then, from the solution of the latter, start a local optimization with the original objective function (the "second phase"). Here the first-phase optimization consisted in using a larger step ($h = 10^{-2}$) when computing numerical derivatives; the second phase used a finer step equal to $h = 10^{-5}$. In Fig. 5.10 we report the comparison between one- and two-phase optimization.

It can be quite clearly seen that using two phases is extremely beneficial both in terms of precision and of robustness.

In order to check whether MBH was indeed useful, we counted, for the 1,000 experiments made, the total number of two-phase local searches performed, which resulted to be 950, 046. We then ran the same number of (two-phase) Multistart iterations, i.e., the classical Multistart method as presented in Sect. 5.3.1 but using a two-phase local search. In Fig. 5.11 we report the comparison between MBH and Multistart: the 1,000 best results found by Multistart have been compared with those of MBH—it is clear that this way the behavior of Multistart is artificially much improved; nonetheless the superiority of MBH is striking. Therefore we conjecture that, similarly to problems in molecular conformation, also space trajectory optimization possesses a "funnel" structure, in which minima are clustered together.

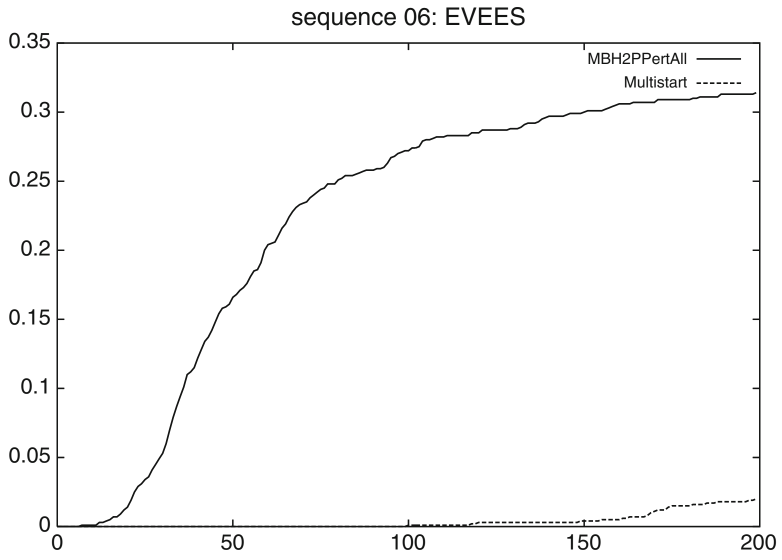


Fig. 5.11 Comparison between Multistart and MBH

For what concerns the experiments performed on the other missions listed in Table 5.6, similar results have been found, confirming the effectiveness of the proposed approach.

In particular, we consider particularly interesting the fact of having been able, for what concerns the Messenger mission, to discover a competitive solution which corresponds to starting the mission years before the starting date of previously known solutions (as well as of the real space mission).

We would like to remark that most of the novel putative global optima we found are truly new solutions, i.e., they cannot be considered as refinement of previously known ones. As an example, we plot in the following figures a trajectory assumed to be optimal for Rosetta mission on April 2008, with objective function (representing total mission variation in velocity) equal to 1.417 km/s and the one we found (and later improved) in May 2008, with objective 1.3678. Although the variation in the objective is not particularly impressive, the trajectories widely differ (see Fig. 5.12–5.13).

As a concluding remark on these experiments, we would like to stress the fact that MBH performed in an excellent way for these problems for which, we recall, no information is available (the problems were coded as black boxes). Nonetheless, thanks to the fact that function evaluation was very cheap, we could use MBH in an efficient way. Moreover, we could use a local optimization method, SNOPT, designed for smooth problems—the excellent performance of our two-phase approach showed that for problems of this kind performing a coarse-grained approximation of derivatives can be very beneficial, as it, in some sense, helps in smoothing out the objective function.

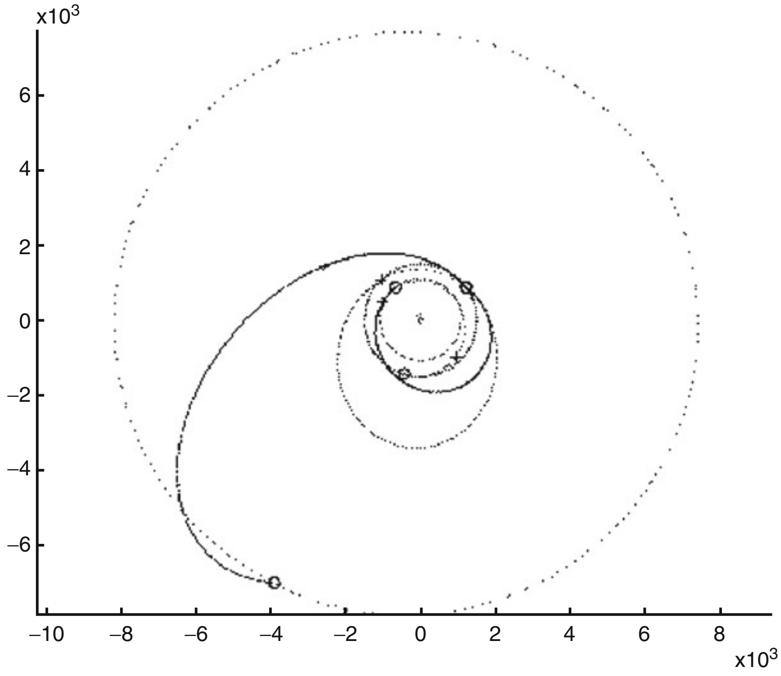


Fig. 5.12 Rosetta mission, $\varDelta_V = 1.\,417$

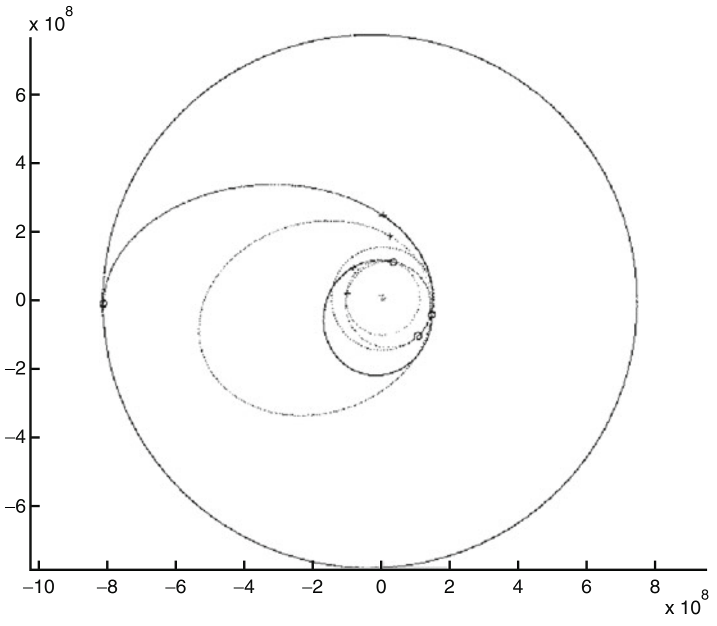


Fig. 5.13 Rosetta mission, $\varDelta_V = 1.\,3678$

5.6 Conclusions and Directions for Further Research

Global optimization techniques offer the possibility of automating the design of interplanetary trajectories. In the past years researchers have shown convincingly such a possibility mostly on high-energy trajectories such as MGA trajectories and using both chemical and low-thrust propulsion. The complexity of the problems solved using an automated search of the solution space included some tough real cases of preliminary mission design such as the BepiColombo mission, the Messenger mission, the Tandem mission, and more. The same global optimization technique, e.g., MBH (as discussed in more detail in this book chapter), can be applied to diverse instances of the interplanetary trajectory problem offering a powerful unified approach to interplanetary trajectory design. Other approaches are possible and can be equally successful, but so far only MBH has been proven to be able to tackle globally low-thrust trajectory design as well as chemical trajectory design. More work is needed to understand the limits of such an approach, when applied to high-energy orbits, as only a few works so far are discussing its performances and on only a few problem instances.

Global optimization techniques need also still to be extended to low-energy transfers, where the search space structure is still largely unknown and the cost of propagating the spacecraft motion can get prohibitive. Future research should tackle this issue and explore possibilities in this area. Eventually the construction of an efficient computational tool able to search with a unified approach for good interplanetary trajectories, given a certain mission profile, will be a challenge for computer scientists. Such a tool, when ready, will allow for good preliminary mission design to be made also without much knowledge on the complex mathematical structure underlying spacecraft dynamics. In Table 5.7 a list of acronyms used in this paper is reported.

Table 5.7 List of acronyms

ACT	Advanced concepts team
DSM	Deep space maneuver
GTOC	Global trajectory optimization competition
GTOP	Global trajectory optimization problems
LT	Low thrust
MBH	Monotonic basin hopping
MGA	Multiple gravity assist
MNI	Max no improve
MS	Multistart
SA	Simulated annealing
SA-AN	Simulated annealing with adaptive neighborhood
SEP	Solar electric propulsion
SQP	Sequential quadratic programming

References

1. Addis, B., Locatelli, M., Schoen, F.: Disk packing in a square: A new global optimization approach. INFORMS J. on Computing **20**(4), 516–524 (2008)
2. Addis, B., Cassioli, A., Locatelli, M., Schoen, F.: A global optimization method for the design of space trajectories. COAP **3**, 635–652 (2011)
3. Armellin, R., Di Lizia, P., Topputo, F., Lavagna, M., Bernelli-Zazzera, F., Berz, M.: Gravity assist space pruning based on differential algebra. Celestial Mech. Dyn. Astron. **106**(1), 1–24 (2010)
4. Corana, A., Marchesi, M., Martini, C., Ridella, S.: Minimizing multimodal functions of continuous variables with the 'simulated annealing' algorithm. ACM Trans. Math. Software (TOMS) **13**(3), 280 (1987)
5. Danby, J.: Fundamentals of Celestial Mechanics. Willman-Bell, Richmond (1988)
6. Di Lizia, P., Radice, G.: Advanced Global Optimisation Tools for Mission Analysis and Design. Tech. Rep. 03-4101b, European Space Agency, the Advanced Concepts Team, 2004
7. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
8. Gil-Fernández, J., Graziano, M., Gomez-Tierno, M., Milic, E.: Autonomous Low-Thrust Guidance: Application to SMART-1 and BepiColombo. Ann. New York Acad. Sci. **1017**, 307–327 (2004), Astrodynamics, Space Missions, and Chaos
9. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM J. Optim. **12**(4), 979–1006 (2002)
10. Gill, P.E., Murray, W., Saunders, M.A.: User's Guide for SNOPT Version 7, Software for Large-Scale Nonlinear Programming. Stanford Business Software Inc., Mountain View, USA (2006)
11. Greeley, R., Johnson, T., E.: Report of the NASA Science Definition Team for the Jupiter Icy Moons Orbiter. Tech. rep., NASA Science Definition Team, Feb. 2004
12. Grosso, A., Jamali, A., Locatelli, M., Schoen, F.: Solving the problem of packing equal and unequal circles in a circular container. J. Global Optim. **47**(1), 63–81 (2010)
13. Izzo, D.: Lambert's problem for exponential sinusoids. Tech. Rep. ACT-RPT-4100-DI-LMSP01, ESA, Apr. 2005
14. Izzo, D.: Global Trajectory Optimization Competition portal. http://www.esa.int/gsp/ACT/mad/pp/GTOC, July 2012
15. Izzo, D.: Global optimization and space pruning for spacecraft trajectory design. In: Spacecraft Trajectory Optimization, pp. 178–200. Cambridge University Press, New York (2010)
16. Izzo, D., Becerra, V., Myatt, D., Nasuto, S., Bishop, J.: Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. J Global Optim. **38**(2), 283–296 (2007)
17. Izzo, D., Vinko, T., Zapatero, M.: Global Trajectory Optimization Competition Database. http://www.esa.int/gsp/ACT/inf/op/globopt.htm, July 2012
18. Katzkowski, M., Corral, C., Jehn, R., Pellon, J.-L., Landgraf, M., Khan, M., Yanez, A., Biesbroek, R.: BepiColombo Mercury Cornerstone Mission Analysis: Input to Definition Study. Tech. rep., ESA European Space Operations Centre, Apr. 2002
19. Kirkpatrick, S., Gelatt, C.D. Jr., Vecchi, M.P.: Optimization by simulated Annealing. Science **220**, 671–680 (1983)
20. Leary, R.H.: Global optimization on funneling landscapes. J. Global Optim. **18**, 367–383 (2000)
21. Locatelli, M.: On the multilevel structure of global optimization problems. Comput. Optim. Appl. **30**(1), 5–22 (2005)
22. Locatelli, M., Vasile, M.: A hybrid multiagent approach for global trajectory optimization. J. Global Optim. **44**(4), 461–479 (2009)

23. Myatt, D., Becerra, V., Nasuto, S., Bishop, J.: Advanced Global Optimisation Tools for Mission Analysis and Design. Tech. Rep. 03-4101a, European Space Agency, the Advanced Concepts Team, 2004
24. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, USA (2006)
25. Parcher, D.W., Sims, J.A.: Gravity-assist trajectories to Jupiter using nuclear electric propulsion. In: AAS/AIAA Astrodynamics Specialist Conference, Aug. 2005
26. Sims, J.A., Flanagan, S.N.: Preliminary design of low-thrust interplanetary missions. In: AAS/AIAA Astrodynamics Specialist Conference, Aug. 1999
27. Strange, N.J., Sims, J.A.: Methods for the design of v-infinity leveraging maneuvers. In: AAS/AIAA Astrodynamics Specialist Conference, July/Aug. 2001
28. Tsiolkovsky, K.E.: Exploration of the universe with reaction machines (in Russian). Sci. Rev. **5** (1903)
29. Vasile, M., De Pascale, P.: Preliminary design of multiple gravity-assist trajectories. J. Spacecraft Rockets **43**(4), 794–805 (2006)
30. Vasile, M., Minisci, E., Locatelli, M.: An inflationary differential evolution algorithm for space trajectory optimization. IEEE Trans. Evol. Comput. **15**(2), 267–281 (2011)
31. Vavrina, M.A., Howell, K.C.: Global low-thrust trajectory optimization through hybridization of a genetic algorithm and a direct method. In: AIAA/AAS Astrodynamics Specialist Conference, Aug. 2008
32. Vinko, T., Izzo, D.: Global Optimisation Heuristics and Test Problems for Preliminary Spacecraft Trajectory Design. Tech. Rep. GOHTPPSTD, European Space Agency, the Advanced Concepts Team, 2008
33. Wales, D.J., Doye, J.P.K.: Global optimization by basin-hopping and the lowest energy structures of Lennard–Jones clusters containing up to 110 atoms. J. Phys. Chem. A **101**(28), 5111–5116 (1997)
34. Whiffen, G.J.: An investigation of a Jupiter Galilean Moon orbiter trajectory. In: AAS/AIAA Astrodynamics Specialist Conference, Aug. 2003
35. Yam, C.H., McConaghy, T.T., Chen, K.J., Longuski, J.M.: Design of low-thrust gravity-assist trajectories to the outer planets. In: 55th International Astronautical Congress, Oct. 2004
36. Yam, C.H., Biscani, F., Izzo, D.: Global optimization of low-thrust trajectories via impulsive Delta-V transcription. In: 27th International Symposium on Space Technology and Science, July 2009
37. Yam, C.H., McConaghy, T.T., Chen, K.J., Longuski, J.M.: Preliminary design of nuclear electric propulsion missions to the outer planets. In: AIAA/AAS Astrodynamics Specialist Conference, Aug. 2004
38. Yarnoz, D.G., Jehn, R., Croon, M.: Interplanetary navigation along the low-thrust trajectory of bepiColombo. Acta Astronautica **59**(1–5), 284–293 (2006)

Chapter 6
Indirect Methods for the Optimization of Spacecraft Trajectories

Guido Colasurdo and Lorenzo Casalino

Abstract In this chapter, a general methodology to apply the theory of optimal control to spacecraft trajectories is outlined. This peculiar procedure allows for an almost mechanical derivation of the boundary conditions which must be satisfied by an optimal trajectory, depending on the specific constraints of the problem under analysis. The general way of posing the optimal control problem makes this indirect approach suitable to manage many specific features of the space missions, such as, impulsive and/or low-thrust engines, planetary flybys, atmospheric flight, and so on. Peculiarities of the problem simply modify the set of differential equations and boundary conditions in the context of the same theoretical frame. Examples will show that the indirect approach can deal efficiently with complex problems of space trajectory optimization. As in the case of direct methods, the indirect approach requires a tentative solution, and convergence to the optimum is typically obtained if the tentative solution is sufficiently close to the optimal one. Suitable procedures to find tentative guesses for the considered problems are described.

Keywords Trajectory optimization • Optimal control theory • Indirect methods • Interplanetary mission

G. Colasurdo
Università di Roma "Sapienza", Via Eudossiana, 18, Roma, Italy
e-mail: guido.colasurdo@uniroma1.it

L. Casalino (✉)
Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, Italy
e-mail: lorenzo.casalino@polito.it

6.1 Introduction

The feasibility and cost of a space mission are strongly related to the spacecraft trajectory. Minimization of propellant requirements may be fundamental to deliver a sufficient payload mass and guarantee the mission feasibility or to allow for a less expensive launcher, thus reducing costs. An efficient method for trajectory optimization is therefore needed during mission planning. Final mass or payload maximization and propellant mass or flight time minimization are the problems that must be typically dealt with.

Most of the methods for the optimization of spacecraft trajectories can be grouped into three main classes. Direct methods transform the problem into a parameter optimization (nonlinear programming) and solve it by means of gradient-based procedures. Indirect methods use the theory of optimal control to transform the optimization problem into a boundary value problem (BVP) solved by means of shooting procedures. Evolutionary algorithms, instead, exploit large populations of solutions which evolve according to specific rules towards the global optimum. Indirect methods are the object of this chapter.

The indirect approach offers many advantages. First, it allows for an exact, even though numerical, optimization (in the limits of the adopted dynamical model and integration accuracy). In addition, as far as low-thrust missions are concerned, the computational cost of indirect methods is typically lower compared to direct methods, which require a much larger number of variables for an accurate trajectory description (severe approximations are necessary to genetic algorithms). Finally, the indirect approach provides useful theoretical information on the problem which is dealt with.

Betts [1], who carries out an accurate comparison of direct and indirect methods, underlines three main drawbacks concerning indirect techniques: first, the necessity of deriving analytic expressions for the necessary conditions, that could become discouraging when dealing with complex problems. Second, the region of convergence for a shooting algorithm may be quite small, as it is necessary to guess values for adjoint variables that may not have an obvious physical meaning. Third, for problems with path inequalities, it is necessary to guess the sequence of constrained and unconstrained subarcs.

In the previous two decades the authors have studied and proposed a peculiar approach that mitigates the drawbacks of the indirect methods. In particular, the position of the problem and the derivation of the optimal conditions have been made general and easy; the application of the indirect approach has been extended to very complex problems of spaceflight mechanics. Some enhancements have been introduced also in the formulation of the BVP, and the convergence of the shooting procedure has improved. The capability of achieving the numerical solution is still dependent on the tentative solution which is assumed to start the procedure, but the simplicity of the theoretical approach permits a fast formulation of a series of optimization problems with increasing difficulty: the solution of the most complex problem (i.e., the actual problem) is obtained via the solution of similar but easier problems. It is important to note that also direct methods rely on a tentative solution and may show convergence difficulties. The easy management of the theoretical problem is useful to treat path inequalities; this topic is not discussed here; an example can be found in [10].

The authors' approach to indirect optimization is described in the present chapter and applied to the optimization of space missions. The application of the theory of optimal control to a generic spacecraft trajectory, according to the two-body problem model, is considered, and the derivation of the optimal control law and conditions for optimality is first exposed. Some interesting applications concerning interplanetary trajectories are then presented and discussed. The numerical procedure to obtain converged solutions is highlighted; examples are presented with the main aim of providing indications on strategies capable of finding the tentative solutions that guarantee the numerical convergence.

6.2 Position of the Optimal Control Problem

The indirect approach to optimization uses the optimal control theory (OCT), which is based on calculus of variations. The position of the optimization problem, which is here described, has the most suitable form to deal with the optimization of space trajectories and to exploit the capabilities of the numerical procedure that has been selected to solve the BVP resulting from the OCT application. A more detailed discussion about OCT can be found in [2, 3, 13].

The system is described by a set of state variables $\mathbf{x}$; differential equations rule the evolution from the initial to the final state

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t). \tag{6.1}$$

They are functions of $\mathbf{x}$, of the control variables $\mathbf{u}$, and the independent variable t (usually, the time).

The trajectory between the initial and final point (the external boundaries) is usefully split into n arcs at the points (internal boundaries) where the state or control variables are discontinuous or constraints are imposed. The jth arc starts at $t_{(j-1)_+}$ and ends at t_{j_-}, where the state variables are $\mathbf{x}_{(j-1)_+}$ and $\mathbf{x}_{j_-}$, respectively ($j-$ and $j+$ denote values just before and after point j).

Nonlinear constraints are imposed at both internal and external boundaries. These boundary conditions are grouped into a vector $\boldsymbol{\psi}$ and written in the form

$$\psi(\mathbf{x}_{(j-1)_+}, \mathbf{x}_{j_-}, t_{(j-1)_+}, t_{j_-}) = 0\mathrm{j} = 1, \ldots, n. \tag{6.2}$$

Additional path constraints may hold along an entire arc; constraints may also concern the control variables $\mathbf{u}$.

Meyer formulation is preferred to define the optimization problem, which searches for extremal values (maxima or minima) of a functional

$$J = \varphi(\mathbf{x}_{(j-1)_+}, \mathbf{x}_{j_-}, t_{(j-1)_+}, t_{j_-}) j = 1, \ldots, n. \tag{6.3}$$

Alternatively, Lagrange formulation could be employed; the formulations are however equivalent, and each one can be easily obtained from the other. A necessary condition for optimality requires that the first variation of J is null for any admissible variation along the path ($\delta\mathbf{x}$ and $\delta\mathbf{u}$) and at boundary points ($\delta\mathbf{x}_{(j-1)_+}$, $\delta\mathbf{x}_{j_-}$, $\delta t_{(j-1)_+}$ and δt_{j_-}).

Lagrange multipliers (constants $\boldsymbol{\mu}$ associated with boundary conditions and adjoint variables $\boldsymbol{\lambda}$ associated with the differential equations) are introduced, and a modified functional is defined:

$$J^* = \varphi + \mu^T \psi + \sum_j \int_{t_{(j-1)_+}}^{t_{j_-}} \lambda^T (\mathbf{f} - \dot{\mathbf{x}}) dt, \tag{6.4}$$

where the dot ($\dot{}$) denotes the time derivative.

The functionals J and J^* coincide if all boundary conditions and differential equations are satisfied. One can differentiate J^* and obtain

$$\begin{aligned}
\delta J^* = & \left(-H_{(j-1)_+} + \frac{\partial \varphi}{\partial t_{(j-1)_+}} + \mu^T \frac{\partial \psi}{\partial t_{(j-1)_+}} \right) \delta t_{(j-1)_+} \\
& + \left(H_{j_-} + \frac{\partial \varphi}{\partial t_{j_-}} + \mu^T \frac{\partial \psi}{\partial t_{j_-}} \right) \delta t_{j_-} \\
& + \left(\lambda^T_{(j-1)_+} + \frac{\partial \varphi}{\partial \mathbf{x}_{(j-1)_+}} + \mu^T \left[\frac{\partial \psi}{\partial \mathbf{x}_{(j-1)_+}} \right] \right) \delta \mathbf{x}_{(j-1)_+} \\
& + \left(-\lambda^T_{j_-} + \frac{\partial \varphi}{\partial \mathbf{x}_{j_-}} + \mu^T \left[\frac{\partial \psi}{\partial \mathbf{x}_{j_-}} \right] \right) \delta \mathbf{x}_{j_-} \\
& + \int_{t_{(j-1)_+}}^{t_j} \left(\left(\frac{\partial H}{\partial \mathbf{x}} + \dot{\lambda}^T \right) \delta \mathbf{x} + \frac{\partial H}{\partial \mathbf{u}} \delta \mathbf{u} \right) \mathrm{d}t \; j = 1, \ldots, n,
\end{aligned} \tag{6.5}$$

where the Hamiltonian has been introduced:

$$H = \lambda^T \mathbf{f}. \tag{6.6}$$

Optimality requires $\delta J^* = 0$ for any admissible variation. By nullifying the coefficients of $\delta\mathbf{x}$ and $\delta\mathbf{u}$, one has the Euler–Lagrange equations for the adjoint variables

$$\frac{\mathrm{d}\lambda}{\mathrm{d}t} = -\left(\frac{\partial H}{\partial \mathbf{x}} \right)^T \tag{6.7}$$

and algebraic equations for the control variables

$$\left(\frac{\partial H}{\partial \mathbf{u}}\right)^T = 0. \tag{6.8}$$

A control variable may be subject to constraints (e.g., the thrust magnitude varies between a minimum value, typically 0, and a maximum value $T_{\max}$). In these cases, Eq. (6.8) may not provide the optimal control. However, in agreement with Pontryagin's maximum principle (PMP) [2], the optimal control must maximize H, given by Eq. (6.6). In particular, when the Hamiltonian is linear with respect to a control variable (e.g., thrust magnitude), a bang-bang control arises, and the control assumes either its maximum or minimum value, except for singular arcs [2, 4].

Finally, the boundary conditions for optimality are obtained by nullifying the coefficients of $\delta\mathbf{x}_{(j-1)_+}$, $\delta\mathbf{x}_{j_-}$, $\delta t_{(j-1)_+}$, δt_{j_-}. One has

$$-\lambda_{j_-}^T + \frac{\partial\varphi}{\partial\mathbf{x}_{j_-}} + \mu^T\left[\frac{\partial\psi}{\partial\mathbf{x}_{j_-}}\right] = 0 \qquad j = 1,\ldots,n, \tag{6.9}$$

$$\lambda_{j_+}^T + \frac{\partial\varphi}{\partial\mathbf{x}_{j_+}} + \mu^T\left[\frac{\partial\psi}{\partial\mathbf{x}_{j_+}}\right] = 0 \qquad j = 0,\ldots,n-1, \tag{6.10}$$

$$H_{j_-} + \frac{\partial\varphi}{\partial t_{j_-}} + \mu^T\frac{\partial\psi}{\partial t_{j_-}} = 0 \qquad j = 1,\ldots,n, \tag{6.11}$$

$$-H_{j_+} + \frac{\partial\varphi}{\partial t_{j_+}} + \mu^T\frac{\partial\psi}{\partial t_{j_+}} = 0 \qquad j = 0,\ldots,n-1. \tag{6.12}$$

The constant Lagrange multipliers are eliminated form Eqs. (6.9)–(6.12); the resulting boundary conditions for optimality and the boundary conditions on the state variables, given by Eq. (6.2), are collected in a single vector in the form

$$\sigma(\mathbf{x}_{(j-1)_+}, \mathbf{x}_{j_-}, \lambda_{(j-1)_+}, \lambda_{j_-}, t_{(j-1)_+}, t_{j_-}) = 0, \tag{6.13}$$

which, together with state and adjoint differential equations, defines a multipoint boundary value problem (MPBVP).

Noticeable difficulties in the solution of a MPBVP arise when the relevant times are unknown and the lengths of the integration intervals are free. A change of independent variable is introduced [11] to deal with the indetermination of the relevant times and fix the extremes of the integration intervals; in the j-th phase, t is replaced by

$$\varepsilon = j - 1 + \frac{t - t_{j-1}}{t_j - t_{j-1}}, \tag{6.14}$$

which assumes consecutive integer values at the boundaries. The differential equation for a generic variable y (either state, x, or adjoint variable, λ) during phase j becomes

$$\frac{\mathrm{d}y}{\mathrm{d}\varepsilon} = (t_j - t_{j-1})\frac{\mathrm{d}y}{\mathrm{d}t}. \tag{6.15}$$

Some initial values of the state and adjoint variables are unknown; several constant parameters, such as the relevant times t_j, are also unknown. An iterative procedure is used to determine the unknowns that permit the fulfillment of the boundary conditions. Tentative values are first assumed; it is very important that these initial values are sufficiently close to the optimal solution to guarantee convergence. The assumption of a suitable solution is a fundamental step in the optimization procedure; details will be given in Sect. 6.4.

Once tentative values have been assigned to the unknowns, the differential equations are integrated and the errors on the boundary conditions are determined. The unknowns are then in turn varied by a small amount to evaluate, according to a forward-finite-difference scheme, the derivatives of the errors with respect to the unknowns. Newton's method is used to bring the errors to zero. A more precise analytical approach [11] may be used when convergence difficulties arise.

6.3 Optimization of Space Trajectories

Preliminary analysis of spacecraft trajectories is typically carried out assuming a point mass spacecraft under the influence of a single body. The two-body model can also be used to deal with interplanetary trajectories, as the patched-conic approximation is usually employed in the initial analyses. An efficient approach only analyzes the heliocentric legs; at the patch points with the planetocentric legs, suitable boundary conditions take the maneuvers inside the planets' spheres of influence into account. The formulation of the trajectory optimization in the two-body problem is given here, making particular reference to the heliocentric legs of an interplanetary trajectory, due to the peculiarity of the relevant boundary conditions.

The state of the spacecraft is described by position $\mathbf{r}$, velocity $\mathbf{v}$, and mass m, and the state equations are

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \mathbf{v}, \tag{6.16}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \mathbf{g} + \frac{\mathbf{T}}{m}, \tag{6.17}$$

$$\frac{\mathrm{d}m}{\mathrm{d}t} = -\frac{T}{c}, \tag{6.18}$$

where $\mathbf{T}$ is the engine thrust and $\mathbf{g}$ is the gravitational acceleration (an inverse-square gravity field $\mathbf{g} = -\mu\mathbf{r}/r^3$ is typically assumed); the propellant mass-flow rate is expressed by the ratio of the thrust magnitude to the constant effective exhaust velocity c. This indirect method is also capable of treating propulsion systems with variable c, i.e., specific impulse [5], as shown in Sect. 6.4.4.

The Hamiltonian, defined by Eq. (6.6), is

$$H = \lambda_r^T \mathbf{v} + \lambda_v^T (\mathbf{g} + \mathbf{T}/m) - \lambda_m T/c. \tag{6.19}$$

The Euler–Lagrange equations for the adjoint variables, Eq. (6.7), provide [8]

$$\left[\frac{\mathrm{d}\lambda_r}{\mathrm{d}t}\right]^T = -\lambda_v^T \left[\frac{\partial \mathbf{g}}{\partial \mathbf{r}}\right], \tag{6.20}$$

$$\left[\frac{\mathrm{d}\lambda_v}{\mathrm{d}t}\right]^T = -\lambda_r^T, \tag{6.21}$$

$$\frac{\mathrm{d}\lambda_m}{\mathrm{d}t} = \frac{\lambda_v T}{m^2}, \tag{6.22}$$

where the gravity-gradient matrix appears in Eq. (6.20). Equations (6.16)–(6.18) and (6.20)–(6.22) constitute the system of differential equations, which is integrated numerically.

The thrust direction and its magnitude are typically the control variables, which must maximize H in agreement with PMP [2]. The optimal thrust direction is therefore parallel to the velocity adjoint vector $\boldsymbol{\lambda}_\mathrm{v}$, which is named primer vector [14]. The *switching function*

$$S_F = \frac{\lambda_v}{m} - \frac{\lambda_m}{c} \tag{6.23}$$

is introduced, and Eq. (6.19) is rewritten as

$$H = \lambda_r^T \mathbf{v} + \lambda_v^T \mathbf{g} + T S_F. \tag{6.24}$$

The thrust magnitude assumes its maximum value when the switching function S_F is positive, whereas it is set to zero when S_F is negative, again to maximize the Hamiltonian. Singular arcs occur when S_F remains zero during a finite time; Eq. (6.24) is not sufficient to decide the optimal thrust magnitude (singular arcs are here excluded; they may be required during atmospheric flight [4]). To improve the numerical accuracy, the trajectory is split into maximum-thrust arcs and coast arcs. The number and order of the arcs, i.e., the trajectory switching structure, are assigned a priori, and the arc time-lengths are additional unknowns. The boundary conditions for optimality state that the switching function S_F must be null at the extremities of each thrust arc. The numerical procedure provides the optimal solution that corresponds to the assigned switching structure. This solution is then checked in the light of PMP, by means of an analysis of the switching function; if PMP is violated, coast or propelled arcs are inserted or removed, in accordance with the behavior of S_F, to obtain an improved solution (e.g., a coast arc is introduced when S_F becomes negative during a propelled arc).

Suitable boundary conditions define the mission. Reference is here made to interplanetary trajectories; a similar (and usually simpler) analysis can be carried out for different cases, for example, orbital maneuvers of an Earth satellite. At the exit from Earth's sphere of influence, spacecraft and Earth positions coincide (the dimension of the sphere of influence can be neglected), and the mass is typically a function of the hyperbolic excess velocity $\mathbf{v}_{\infty 0} = \mathbf{v}_0 - \mathbf{v}_E(t_0)$ where subscript E refers to the Earth. The boundary conditions at the initial point (subscript 0) are

$$\mathbf{r}_0 = \mathbf{r}_E(t_0), \tag{6.25}$$

$$m_0 = f(v_{\infty 0}). \tag{6.26}$$

The arrival at the target body with zero-hyperbolic excess velocity is here considered, as it appears to be the most general end condition, corresponding to the rendezvous with an asteroid or to the minimum-energy approach to a planet. The relevant boundary conditions at the final point (subscript n) are therefore

$$\mathbf{r}_n = \mathbf{r}_T(t_n), \tag{6.27}$$

$$\mathbf{v}_n = \mathbf{v}_T(t_n), \tag{6.28}$$

where subscript T denotes the target body. The final mass m_n is the performance index which is maximized.

In the past, low-thrust and impulsive trajectories constituted two separate classes of missions, optimized by means of quite different methods. In particular, the classical application of the OCT to impulsive maneuvers generates a numerical problem which is typically solved by gradient-based procedures. The present approach can insert an impulsive maneuver in a mission that also uses low-thrust engines, i.e., electric propulsion. A deep-space impulse represents a discontinuity in spacecraft velocity and mass; subscripts $i-$ and $i+$ are used to denote variables just before and after the impulse, and the velocity change $\Delta\mathbf{v} = \mathbf{v}_{i+} - \mathbf{v}_{i-}$ is introduced; one has the boundary conditions

$$\mathbf{r}_{i+} = \mathbf{r}_{i-}, \tag{6.29}$$

$$m_{i+} = m_{i-} \exp(-\Delta v/c), \tag{6.30}$$

$$t_{i+} = t_{i-}\,. \tag{6.31}$$

Gravity assists are dealt with by means of a similar approach. At planetary flybys, spacecraft position is the same as the planet both before (subscript $g-$) and after (subscript $g+$) the flyby; the hyperbolic excess velocity $\mathbf{v}_\infty = \mathbf{v}_g - \mathbf{v}_p(t_g)$ changes its direction but maintains the same magnitude. One has

$$\mathbf{r}_{g+} = \mathbf{r}_P, \tag{6.32}$$

$$\mathbf{r}_{g-} = \mathbf{r}_P, \tag{6.33}$$

$$\mathbf{v}^2_{\infty g+} = \mathbf{v}^2_{\infty g-}, \tag{6.34}$$

$$t_{g+} = t_{g-}\,. \tag{6.35}$$

In some cases the spacecraft would fly too close to the planet surface. A constraint on the periapsis height might be required; in this occurrence, an additional condition on the velocity turn arises

$$\mathbf{v}^T_{\infty g+}\mathbf{v}_{\infty g-} = -\cos 2\phi \mathbf{v}^2_{\infty g-}, \tag{6.36}$$

where $\cos\phi = V_p^2/(\mathbf{v}^2_{\infty g-} + V_p^2)$, with $V_P = \sqrt{\mu_p/R_p}$ being the circular velocity at the minimum allowable distance from the planet surface.

Equations (6.9)–(6.12) are used to determine the boundary conditions for optimality. At departure, after eliminating the constant Lagrange multipliers μ, one has

$$\lambda_{v0} = \lambda_{m0}\frac{\mathrm{d}f}{\mathrm{d}v_{\infty 0}}\frac{\mathbf{v}_{\infty 0}}{v_{\infty 0}}, \tag{6.37}$$

which states that the primer vector must be parallel to the hyperbolic excess velocity (and fixes its magnitude) and the transversality condition

$$\lambda^T_{r0}\mathbf{v}_{\infty 0} + TS_{F0} = 0. \tag{6.38}$$

In a similar way, at the final point, the transversality condition would be analogous to Eq. (6.38)

$$\lambda^T_{rn}\mathbf{v}_{\infty n} + TS_{Fn} = 0 \tag{6.39}$$

but becomes $S_{Fn} = 0$ if zero-hyperbolic excess velocity is imposed. In addition, one also has $\lambda_{mn} = 1$ when the final mass is maximized; $\lambda_{mn} = 0$ and $H_n = 1$ are instead obtained if the final time is minimized and m_n is left free. Additional time constraints (e.g., on the overall trip time $t_n - t_0$) may be introduced; the approach described here would provide the specific transversality conditions corresponding to the imposed constraints.

At each deep-space impulse one has

$$\lambda_{vi-} = \lambda_{vi+} = \frac{\lambda_{mi-}m_{i-}}{c}\frac{\Delta\mathbf{v}_i}{\Delta v_i} = \frac{\lambda_{mi+}m_{i+}}{c}\frac{\Delta\mathbf{v}_i}{\Delta v_i}, \tag{6.40}$$

which states that both the primer vector and the product $m\lambda_m$ are constant across the maneuver; the primer vector is parallel to the velocity impulse. The transversality condition is

$$\lambda_{ri+}^T \Delta\mathbf{v}_i = \lambda_{ri-}^T \Delta\mathbf{v}_i. \tag{6.41}$$

By using Eqs. (6.21) and (6.40), one can show [6] that Eq. (6.41) is equivalent to nullify the time derivative of the primer magnitude. The impulse can be seen as a limiting case of the finite-thrust arc, with length approaching zero; the primer vector, which is parallel to thrust in the finite-thrust case, becomes parallel to the velocity impulse; the two conditions $S_\mathrm{F} = 0$ enforced at the extremities of a finite-thrust arc become $S_F = \dot{S}_F = 0$: for an impulsive maneuver, both switching function and its time derivative must be zero at the impulse (otherwise, S_F would be positive on either side of the impulse). Note that similar conditions also hold for departure from Earth and, in general, for impulses inside a planetary sphere of influence, with a different value of the primer magnitude (and S_F, consequently) to reflect the convenience of using the thrust in the closest proximity of the planet surface.

For a free-height planetary flyby one has

$$\lambda_{vg-} = \lambda_{vg-} \frac{\mathbf{v}_{\infty g-}}{v_{\infty g-}}, \tag{6.42}$$

$$\lambda_{vg+} = \lambda_{vg+} \frac{\mathbf{v}_{\infty g+}}{v_{\infty g+}}, \tag{6.43}$$

which enforce primer vector parallel to the hyperbolic excess velocity both before and after the flyby; the primer magnitude is continuous across the flyby. The parallelism does not hold if the flyby height is constrained; the primer component normal to the flyby plane is zero; the in-plane component $\lambda_v^{\mathrm{perp}}$ perpendicular to the hyperbolic excess velocity is the same before and after the flyby, but the parallel component λ_v^{par} is discontinuous according to

$$\lambda_{vg+}^{par} = \lambda_{vg-}^{par} + 2A\lambda_{vg-}^{perp} \tag{6.44}$$

with

$$A = \frac{\mathrm{d}\phi}{\mathrm{d}\mathbf{v}_{\infty g-}} \mathbf{v}_{\infty g-} = \frac{2}{\tan\phi} \frac{v_{\infty g-}^2}{v_{\infty g-}^2 + V_p^2} \tag{6.45}$$

The transversality condition both for free-height and minimum-height flyby becomes

$$\lambda_{rg-}^T \mathbf{v}_{\infty g-} + TS_{Fg-} = \lambda_{rg+}^T \mathbf{v}_{\infty g+} + TS_{Fg+} \tag{6.46}$$

To improve the convergence, the trajectories are first optimized by introducing additional degrees of freedom and letting the relevant bodies assume the best phasing with the Earth. The corresponding trajectories, which define the most favorable positions of the target planets at encounter, can be flown every year departing on the same day. The positions of the planets on the encounter day of each year in the launch window are then compared to the optimal-phasing positions; favorable opportunities occur when differences are small. The corresponding actual missions are easily found by using the optimal-phasing solution as tentative guess.

6.4 Examples

The optimization of interplanetary trajectories is quite complex and is considered here to highlight the features of the indirect optimization method which is described in this chapter. The most original and significant contributions of this approach are, in fact, in the field of interplanetary missions.

6.4.1 Earth–Mars Round Trip Impulsive Trajectories

Different classes of Earth–Mars round-trip trajectories exist, according to time of flight and stay time on Mars surface [17]. Conjunction-class missions exhibit two Hohmann-like transfers joined by a long stay on Mars, waiting for a favorable return opportunity. Opposition-class missions reduce the stay time by means of an impulse during either the outbound or the inbound leg. Venus, when conveniently placed, can provide a gravity assist which replaces the deep-space impulse and reduces the mission ΔV requirements.

Impulsive trajectories are typically dealt with by direct optimization methods or evolutionary algorithms; however, the use of an indirect approach [6] may offer several advantages. Solutions for single legs are easily obtained and can be combined to build a tentative solution for the complete round-trip mission; usually, convergence does not present difficulties. An additional benefit comes from the problem periodicities that can be exploited to obtain all the mission opportunities in a large launch window with minimum effort [7]. In fact, Earth–Mars relative angular position repeats after a synodic period of 2.13 years; conjunction-class and opposition-class (without flyby) missions recur with the same periodicity; small differences are mainly due to Mars eccentricity and inclination. After a full 32-year cycle (15 synodic periods) the planet positions and the solutions repeat almost exactly. For missions which exploit Venus flyby, the same phase angles between the relevant three planets are found after a syzigistic period of 6.4 years (comprising 3 Earth–Mars and 4 Venus–Mars synodic periods), which defines the problem periodicity; in a syzigistic period, one good opportunity and two secondary opportunities are found with flyby in the outbound leg; symmetrical missions

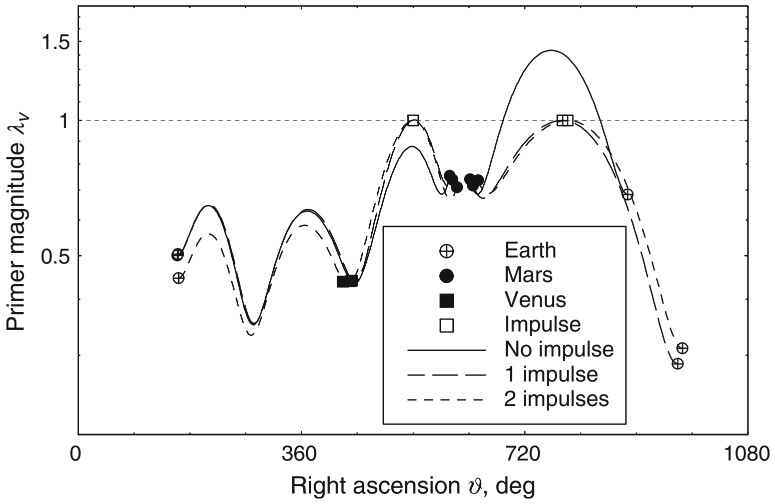


Fig. 6.1 Primer magnitude for round-trip Mars missions with Venus flyby in the outbound leg

with flyby in the inbound leg exist. It is important to note that a single solution for each type of mission can be used as a tentative guess to find other missions of the same type in different launch windows, with almost immediate convergence. In contrast, other optimization methods cannot exploit the similarity of the solutions with comparable efficiency.

The analysis of the switching function of the baseline solution may suggest the introduction of additional impulses during the flight. As an example, Fig. 6.1 shows the primer magnitude behavior during missions with Venus flyby in the outbound leg, departure in March 2015 and Mars stay of 60 days [7]. No mass disposal (e.g., during the Mars stay) is considered, and $m\lambda_m$ results to be constant (one should note that the presence of mass dumping [16] would reduce $m\lambda_m$ and the required primer value at impulses after the discontinuities, to signal the convenience of using thrust after the mass has diminished). A suitable normalization is here adopted to have positive S_F where $\lambda_v > 1$. Impulses at departure from and arrival to a planet occur with $\lambda_v < 1$, as impulses inside a planet sphere of influence are more convenient than deep-space impulses. The baseline solution with no deep-space impulse (solid line in Fig. 6.1) requires $\Delta v = 17.24$ km/s. The switching function becomes positive during the return leg, suggesting the addition of a deep-space impulse. When this impulse is added, an improved solution with a single impulse in the inbound leg is obtained ($\Delta v = 15.42$ km/s); however, this new solution requires the addition of another impulse during the outbound leg, as the switching function becomes slightly positive during the Venus–Mars leg. The global-optimum solution (2 impulses) is then obtained with a small additional improvement ($\Delta v = 15.39$ km/s). Note that no constraint on the height of Venus flybys is required; therefore the primer magnitude is continuous at the flybys. The analysis of the switching function proves to be very useful; it has also been employed in [15] to determine which legs of a multiple gravity assist mission require the presence of a deep-space impulse.

6.4.2 Multiple Gravity Assist Missions with Electric Propulsion

Indirect methods are better suited to deal with electric propulsion (EP) missions, which imply long thrust arcs. A large number of variables are required to accurately describe the trajectory when direct methods or evolutionary approaches are employed; this fact makes the optimization of EP trajectories by means of direct methods computationally demanding, with long CPU times to optimize a single trajectory. The use of evolutionary algorithms is almost prohibitive, unless substantial approximations in the trajectory description are adopted (e.g., shape-based methods). On the contrary, the use of the indirect approach is straightforward, with a limited number of variables, high accuracy, and relatively small computation time. The main difficulty which is related to the use of this indirect approach is the definition of a suitable tentative solution. It is convenient to split the trajectory in several branches, which can be separately optimized with limited effort; the solutions are then patched together to obtain the tentative guess for the whole mission. The use of multiple shooting methods may be useful to improve convergence; the trajectory is split into two or more parts; each part is integrated separately starting from its own tentative values (instead of continuing the previous part), and continuity is enforced by additional boundary conditions at the patch points.

A multiple gravity assist trajectory to Mercury is here presented. The mission performs flybys of Earth, Venus (twice), and Mercury (three times) to progressively reduce the spacecraft energy, change the inclination and obtain the rendezvous with the target planet. First, the Earth–Venus trajectory with Earth flyby (EEV) has been optimized. Further legs have been added to complete an Earth–Mercury trajectory with Earth and Venus (two) flybys (EEVVM). In parallel, three Mercury-Mercury resonant trajectories, which progressively reduce the hyperbolic excess velocity at each Mercury encounter, have been optimized and patched together. Finally, the EEVVM trajectory and the MMM trajectory were joined, and continuity was enforced to obtain the complete, rigorously optimal, EEVVMMMM mission. The presence of thrust arcs was stated by means of PMP. Several mission opportunities with this flyby structure have been found [9].

Figures 6.2 and 6.3 refer to the trajectory with departure in 2014. It is worth noting that the mission that will be flown by the BepiColombo mission [12] (originally planned for departure in 2012 and then delayed to 2014) presents similar features. Gravity assists provide almost entirely the required orbit changes with minimal adjustments obtained by means of propulsion. Thrust is initially used to have a favorable Earth gravity assist to reduce the perihelion and intercept Venus. Two Venus flybys are required; first, a minimum-height flyby inserts the spacecraft on a 1:1 Venus resonant orbit with a reduced aphelion (its large inclination is dictated by the resonance requirement); the second (free-height) flyby adjusts inclination, closely matching that of Mercury; both aphelion and perihelion are also reduced to intercept Mercury. Three gravity assists of the target planet and propulsion are used to eventually achieve rendezvous.

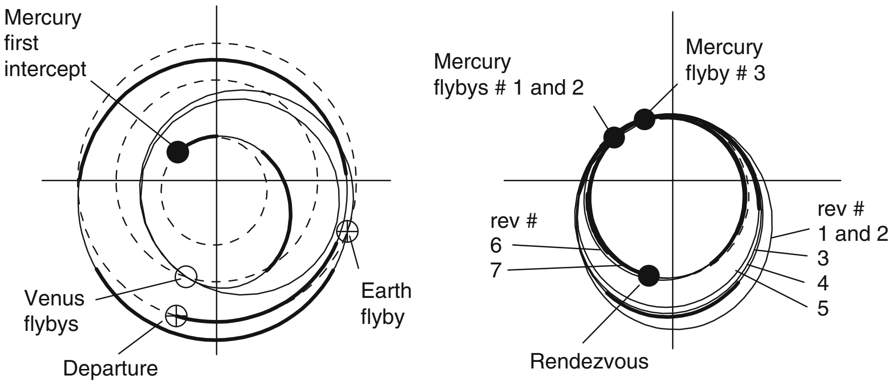


Fig. 6.2 Mission to Mercury with departure in 2014. *Bold line* = thrust arcs, *light line* = coast arcs

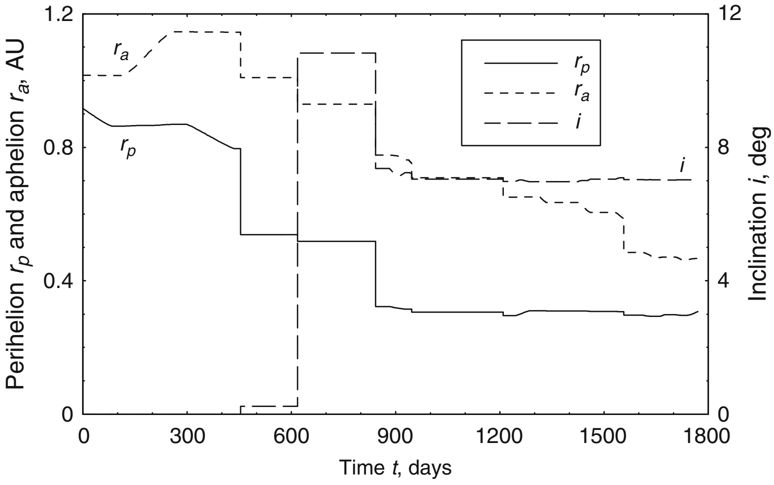


Fig. 6.3 Perihelion, aphelion, and inclination histories during Mercury mission departing in 2014

6.4.3 Propulsion System Optimization

The useful mass of an EP spacecraft is reduced not only by the required propellant but also by the mass of the power system necessary to provide the exhaust velocity c. It is well known that, when EP is employed, an optimal value of the specific impulse exists and maximizes the payload; moreover, the trip time is strictly related to the thrust level. An interesting problem consists in the determination of thrust magnitude, specific impulse, and corresponding optimal trajectory to optimize the mission payload for assigned time of flight. The problem mixes the search for

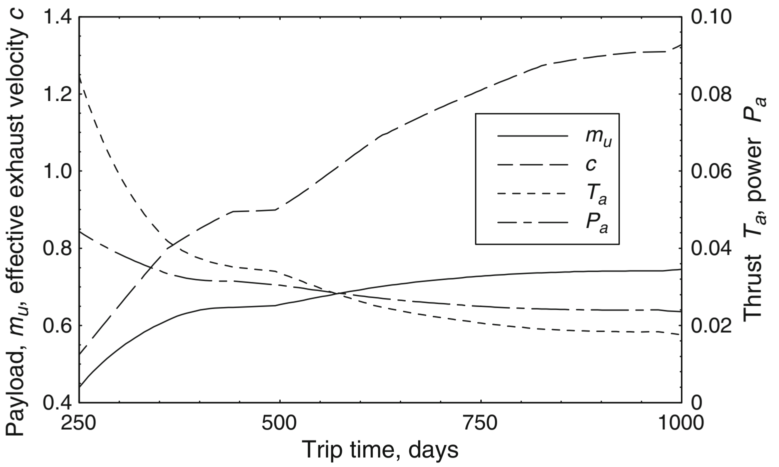


Fig. 6.4 Optimal characteristics of the propulsion system for an Earth–Mars SEP transfer

continuous controls (thrust direction) and optimal parameters (thrust magnitude and specific impulse).

The available thrust T_a and effective exhaust velocity c are additional unknowns; the payload $m_u = m_n - \beta P_a$ (final mass minus propulsion system mass) is the performance index to be maximized, where β is the specific mass of the propulsion system, related to the available thrust power $P_a = T_a c/2$. The optimization procedure considers T_a and c as additional state variables with trivial differential equations $\dot{T}_a = 0$ and $\dot{c} = 0$; the adjoint equations are $\dot{\lambda}_T = -\partial H/\partial T_a$ and $\dot{\lambda}_c = -\partial H/\partial c$ (note that $T = T_a$ when S_F is positive and $T = 0$ when negative). Boundary conditions for optimality state that $\lambda_{T0} = 0$, $\lambda_{c0} = 0$ at the initial point and $\lambda_{Tn} = -\beta c/2$, $\lambda_{cn} = -\beta T_a/2$ at the final point, which implicitly determine T_a and c. The solution of the same problem with assigned thrust and specific impulse can be used as tentative guess. The same approach can be employed when solar electric propulsion (SEP) is employed, with P_a being now the thrust power at 1 astronomical unit (AU); during the propelled arcs, the actual available thrust is considered to be dependent on the distance from the Sun (e.g., $T = T_a/r^2$, with r in astronomical units).

Figure 6.4 refers to an Earth–Mars rendezvous mission with SEP; optimal phasing is assumed to eliminate the influence of the actual positions of the planets. Nondimensional values of payload, effective exhaust velocity, and thrust and power at 1 AU, are shown as functions of the trip time (Earth–Sun mean distance, the corresponding circular velocity, and the maximum mass that can escape from Earth, are the reference values); β = 10 is assumed. Solutions can be obtained with a continuation technique, progressively increasing the trip time. Switching structures with either a single thrust arc or two thrust arcs alternate; PMP is used to assess the optimal switching structure. As the trip time increases, the effective exhaust velocity grows to reduce the propellant consumption and increase the payload mass; thrust instead decreases as less acceleration is required, allowing the further benefit of a lighter propulsion system (the decrease of T_a offsets the increase of c).

6.4.4 Variable Specific Impulse

In principle, many EP systems could present the capability of varying specific impulse (i.e., c) and thrust magnitude at constant thrust power. High thrust and low specific impulse are used where the thrust can be favorably exploited to reduce the trip time; where the thrust is less useful, a large specific impulse is adopted to reduce the propellant consumption.

The engine could operate using a fraction of the available power, and one has $T = 2P/c$ with $0 \leq P \leq P_a$; the Hamiltonian is rewritten as

$$H = \lambda_r^T \mathbf{v} + \lambda_v^T \mathbf{g} + \left(\frac{\lambda_v}{m} - \frac{\lambda_m}{c}\right) 2\frac{P}{c}. \tag{6.47}$$

The optimal specific impulse is first determined. By nullifying $\partial H / \partial c$ one has

$$c_{opt} = 2\frac{m\lambda_m}{\lambda_v}. \tag{6.48}$$

The Hamiltonian linearly depends on P, and a bang-bang control arises; if the specific impulse is unbounded, the engine can operate with $c = c_{\text{opt}}$; one easily finds that the power coefficient in the Hamiltonian is always positive and the thruster is always on at maximum power ($P = P_a$). On the contrary, in the presence of bounds on the attainable values of c, variable between $c_{\min}$ and $c_{\max}$, coast arcs may be required [5] depending on the sign of the power switching function

$$S_P = \frac{\lambda_v}{m} - \frac{\lambda_m}{c_{max}}. \tag{6.49}$$

The same considerations hold for SEP systems, with available power dependent on the distance from the Sun.

Figure 6.5 shows c and T histories during a 1,000-day Earth–Mars SEP mission with nondimensional available power $P_a = 0.01$ at 1 AU. The unbounded-c engine requires large values of c, which exceed the capabilities of existing EP systems. The unbounded solution is used as tentative guess to obtain a rapid convergence of the same problem with bounded c; $c_{\min} = 1$ and $c_{\max} = 1.5$ (corresponding to specific impulses of about 3,000 s and 4,500 s, respectively) are here adopted. The thrust magnitude during the trajectory is constrained between $T_{\min} = 2P/c_{\max}$ and $T_{\max} = 2P/c_{\min}$, with $P = P_a/r^2$ to reflect its dependence on the distance from the Sun. However, P and T are set to zero when S_P is negative. The penalty, due to the bounds on attainable c, is only 0.005 in comparison to the nondimensional payload (0.751 vs. 0.756).

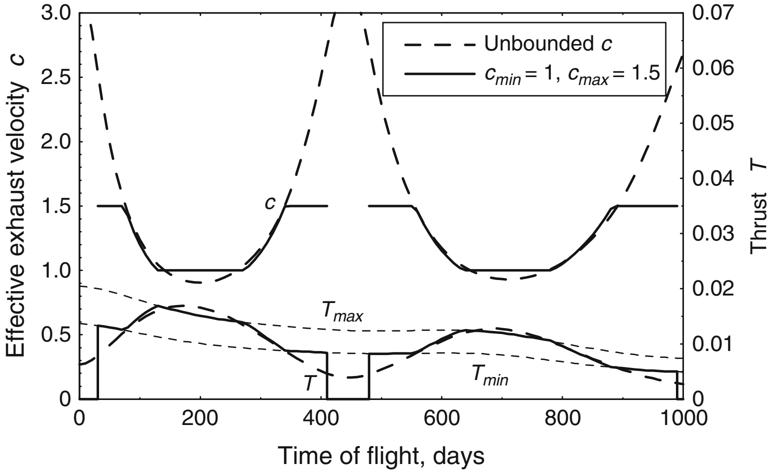


Fig. 6.5 Control histories for 1,000-day Earth–Mars SEP transfer with variable specific impulse

6.5 Final Remarks

The indirect procedure presented here is a powerful means to deal with the optimization of spacecraft trajectories. The necessary conditions for optimality are easily obtained even for complex problems. This has permitted the extension of the indirect approach to problems that were usually left to direct methods. When the problem is formulated in a convenient way, for instance by following the guidelines provided here, extremely accurate solutions can be obtained with very short computation times. Robustness is typically an issue when indirect methods are adopted; however, tentative solutions which allow for convergence can be build by properly splitting the trajectory in elementary legs (optimized separately), or by employing continuation techniques, or, finally, by exploiting periodicities in the motion of the relevant bodies and building on existing solutions.

References

1. Betts, T.: Survey of numerical methods for trajectory optimization. J. Guid. Contr. Dynam. **21**, 193–207 (1998). doi: 10.2514/2.4231
2. Bryson, E.A., Ho, Y.-C.: Applied Optimal Control. Hemisphere, New York, NY (1975)
3. Burghes, D.N., Graham, A.: Introduction to Control Theory, Including Optimal Control. Wiley, New York, NY (1980)
4. Casalino, L.: Singular arc during aerocruise. J. Guid. Contr. Dynam. **23**, 118–123 (2000)
5. Casalino, L., Colasurdo, G.: Optimization of variable-specific-impulse interplanetary trajectories. J. Guid. Contr. Dynam. **27**, 678–684 (2004)

6. Casalino, L., Colasurdo, G., Pastrone, D.: Optimization procedure for preliminary design of opposition-class Mars missions. J. Guid. Contr. Dynam. **21**, 134–140 (1998)
7. Casalino, L., Colasurdo, G., Pastrone, D.: Mission opportunities for human exploration of Mars. Planet. Space Sci. **46**, 1613–1622 (1998)
8. Casalino, L., Colasurdo, G., Pastrone, D.: Optimal low-thrust escape trajectories using gravity assist. J. Guid. Contr. Dynam. **22**, 637–642 (1999)
9. Casalino, L., Colasurdo, G., Rosa Sentinella, M.: Low-thrust trajectories to Mercury with multiple gravity assists. Paper AIAA 2007-5233, AIAA, Reston, VA (2007)
10. Casalino, L., Pastrone, D., Colasurdo, G.: Integrated design of hybrid rocket upper stage and launcher trajectory. Paper 2009-4843, AIAA, Reston, VA (2009)
11. Colasurdo, G., Pastrone, D.: Indirect optimization method for impulsive transfer. Paper AIAA 94–3762, AIAA, Reston, VA (1994)
12. Jehn, R.: BepiColombo a mission to Mercury. 21st International Symposium on Space Flight Dynamics (ISSFD), Toulouse, France, Sept 2009
13. Kirk, D.E.: Optimal Control Theory: An Introduction. Prentice-Hall, Englewood Cliffs (1970)
14. Lawden, D.F.: Optimal Trajectories for Space Navigation. Butterworths, London (1963)
15. Olympio, J.T.: Designing optimal multi-gravity-assist trajectories with free number of impulses. International Symposium on Space Flights Dynamics, Toulouse, France. ESA ESTEC (2009). Available at http://www.esa.int/gsp/ACT/doc/MAD/pub/ACT-RPR-MAD-2009-DesignMGADSM.pdf
16. Ranieri, C., Ocampo, C.: Optimizing finite-burn, round-trip trajectories with Isp constraints and mass discontinuities. J. Guid. Contr. Dynam. **28**, 775–781 (2005). doi: 10.2514/1.9188
17. Walberg, G.: How shall we go to Mars? A review of mission scenarios. J. Spacecraft Rockets **30**, 129–139 (1993)

Chapter 7
Trajectory Optimization for Launchers and Re-entry Vehicles

Francesco Cremaschi

Abstract Launchers and reentry vehicles have in common the necessity to perform part of their trajectory along the planet atmosphere. While this has only negative effects for the launch vehicle, for the re-entry vehicle, the interaction with the atmosphere can be suitably exploited for the fulfillment of the mission. The launch and reentry trajectory is a complex scenario that should be modeled using simplified physics equations describing with sufficient accuracy the subsystems of the vehicle and the environment. In a second step, this simplified physical model, described by sets of differential equations, should be transcribed in a mathematical set of algebraic equations that can be solved by non-linear programming methods (NLP solvers). Upon further analysis of these direct transcription methods, two subclasses can be identified: shooting methods and collocation methods. The NLP solver can be a global optimizer, e.g., genetic algorithm, particle swarm, some other metaheuristics or sequential quadratic programming (SQP), in the case of differentiable functions. Several examples of launch and reentry vehicle problems are presented, with a strong emphasis on the advantages and disadvantages of the various transcription methods and solvers when applied to "real" world problems.

Keywords Launcher • Re-entry vehicle • Trajectory • Gradient method • Multiple shooting • Collocation • Modeling

7.1 Introduction

The trajectory optimization for space applications comprises a wide spectrum of different scenarios: from powerful ascent of a multistage rocket to the gentle descent of a lunar lander, from the slow rendezvous and docking of an automatic

F. Cremaschi (✉)
Astos Solutions GmbH, Meitnerstrasse 8, Stuttgart, Germany
e-mail: Francesco.cremaschi@astos.de

cargo to the International Space Station (ISS,[1]) to the fast planet fly-by of an interplanetary probe. This chapter analyzes only the launchers and reentry vehicles, which have in common the necessity to perform part of their trajectory inside the planet atmosphere. In the case of a conventional launch vehicle, this thin layer counteracts the vehicle propulsion system reducing its performance; for the reentry vehicle, however, the interaction with the atmosphere can even be necessary for the fulfillment of the mission. The complexity of these scenarios requires an automatic optimization performed by a mathematical solver.

Section 7.2 presents the modeling of satellite launchers, whereas Sect. 7.3 details the modeling of the re-entry vehicles. The most common simplified physical equations are listed in Sect. 7.4 whereas the transcription methods and the solvers are detailed in Sect. 7.5. The chapter is enriched with various examples taken from past and future missions (Sect. 7.6); Sect. 7.7 provides the conclusion.

7.2 Launcher Modeling

This section presents the modeling of a typical launcher mission and of a typical launcher vehicle itself; the concepts introduced here may be extended, due to their similarities with those vehicles, also to sub-orbital applications: space-planes and sounding rockets.

7.2.1 Vehicle Modeling

A launch vehicle is a complex aerospace object, with the primary function of placing a certain mass (payload) in a defined orbit. Relaxing this definition, it is also possible to classify the sub-orbital vehicles as launch vehicles i.e., amateur rockets and sounding rockets.

The vehicle design has not changed much since the first attempts: a cylindrical body with the various stages sequentially placed. Optionally, pairs of strap-on boosters are attached to the first stage to increase the thrust at liftoff. Other more exotic configurations are present in launchers still in the design phase or in aborted projects e.g., winged vehicle, that takeoff horizontally and use the lift to reduce the propellant fraction required to reach a stable orbit, cf. Reaction Engines Skylon [2] or EADS Hopper [3].

An interesting application that parallels the trajectory optimization is vehicle design optimization: the optimizer can modify the vehicle model in order to fulfill the requested mission. In this scenario, the modeling of the vehicle increases in complexity with the definition of several constraints between the various modules, e.g., the structural mass of a stage can be linearly linked to the propellant mass of the same stage, or the engine mass can be a function of the maximum thrust provided by it. Increasing in complexity, it is possible to include a first structural

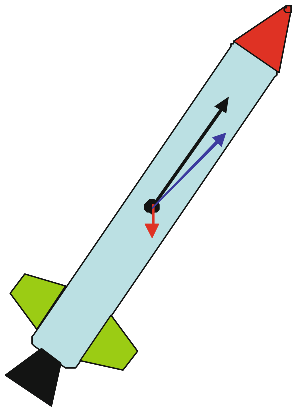


Fig. 7.1 Rocket accelerations: thrust aligned with the main axis, gravity pointing down and total

analysis of the vehicle with the shell thickness function of the loads computed during the mission. In the case of criticality, the solver has the dual possibilities of changing the trajectory to reduce the loads or of increasing the thickness with a consequent increase of the vehicle structural mass.

7.2.1.1 Fairing

Located at the tip of the rocket, a fairing (frontal cone in Fig. 7.1) has two main functions: to reduce the atmospheric drag force and to protect the payload from the external loads present in the first phases of the mission. The shape of the fairing is therefore a compromise between a good aerodynamic effect and a high internal volume required to accommodate the payload. The fairing diameter is a clear example of this: the payload would request at least 4 or 5 m diameter, whereas a diameter of 3 m would reduce considerably the reference area, which is linearly correlated with the drag force (see Eq. (7.6)). The fairing protection function is required for the acoustic loads at liftoff and for the thermal loads caused by the atmosphere molecular friction with the vehicle itself; more details are presented in the Sect. 7.4.2. Vehicles with a wide fairing (e.g., Soyuz—[4]) possess the upper stage covered by the fairing envelope.

In the case of a vehicle design optimization which follows trajectory optimization, the fairing mass can be linked to the payload mass and to the vehicle diameter.

7.2.1.2 Lower Stages

The lower stages have the basic function of both storing the propellant required to fulfill the mission as well as providing the structural stability of the entire vehicle. They generally have a cylindrical section of which the majority of the volume is filled with propellant: on average 90% of the total mass is propellant. The liquid propellants present in these vehicle stages consist of fuel and oxidizer, which require therefore two separate tanks for each stage. For solid propellant, the rocket stage itself is filled with the propellant that presents a typical grain section e.g., cylindrical or star. The grain geometry defines the propellant mass flow function of burn time. One example is provided in Fig. 7.3.

In the typical configuration, there are several lower stages; the reason for this is the mathematical impossibility of reaching a stable orbit with a single stage, which was proven as early as 1903 by the Russian scientist Tsiolkovsky [5]. Equation (7.1) is the ideal rocket equation that describes the achievable increase of velocity (ΔV) as a function of the engine specific impulse (I_{sp}) and the natural logarithm of the ratio between the initial (M_i) and final mass (M_f). I_{sp} is an indication of the performance of the engine and can be computed as the ratio between the exhaust velocity and g_0, the gravity acceleration at sea level.

$$\Delta V = I_{\text{sp}} \cdot g_0 \cdot \ln \frac{M_i}{M_f}. \tag{7.1}$$

The ΔV required to achieve a stable orbit is 9.5 km/s, which produces a final mass equal to 4% of the initial mass with a specific impulse of 300 s. This percentage is not feasible at the current state of technology. In the case of three stages with the same efficiency and effectiveness (which is not fully practicable), each stage has a final mass of 33% of the initial mass: 10% structural and 23% for the upper stages and payload. The final payload in orbit is around 1% of the initial mass.

There exists the possibility to have cross-feeding: the propellant present in a stage is used by engines present in another stage. This is the case for the Space Shuttle [6] and for some vehicles still in the design phase (Falcon Heavy—[7]).

7.2.1.3 Upper Stage

The upper stage is active at higher altitudes where the atmospheric effects are negligible and the attitude of the vehicle can be optimized. This presents the advantage of a precise placement of the payload in the target orbit. This procedure however requires a dedicated avionics system. Together with the reduced dimension of the stage, this causes a structural ratio (i.e., structural mass divided by total stage mass) higher than in the lower stages. Size then becomes interesting as the propellant mass is a function of the tank volume whereas the structural mass is a function of the tank surface: with smaller tanks the ratio between surface and volume increases.

In a typical launcher trajectory optimization, the controllability of the vehicle, in particular of the upper stage, is reduced to ensure that the attitude angle rates are lower than the upper limit achievable by the control system of that stage. At this level of complexity, it is generally not required to design the position of the actuators or the performance of the sensors on board.

7.2.1.4 Other Components

A launcher is composed of several additional components, most of these may however be annexed to the ones presented above: the mass of the interstage between stages 1 and 2 can be added to the structural mass of stage 1. This approximation will be accurate since the interstage is jettisoned together with the exhausted stage 1. Similarly the payload adapter mass should be incorporated in the upper stage structural mass. The payload, on the other hand, is treated differently. In general, this component is modeled as a variable mass that the optimizer can modify to reach the target orbit.

7.2.2 Mission Modeling

The starting point of the trajectory is normally located on the surface of a planet, which need not exclusively be Earth, but also possibly the Moon or Mars. The trajectory can be separated into a propelled ascent phase where the various stage engines are activated, an optional coast phase to reach the desired altitude, and finally the orbit insertion phase. Each phase is characterized by various accelerations acting on the vehicle. During the ascent phase, the propulsion, aerodynamics, and gravity are predominant, whereas in the coast and orbit insertion, the gravity is perturbed by third body and solar radiation effects. The particularity of reusable or partially reusable launch vehicles should also be mentioned: a part of the vehicle will actively return to the planet surface to be refurbished and reused for future missions.

7.2.2.1 Phases Definition

The launcher trajectory should be divided into several phases to consider the events that occur during the mission: separation of a component, coast arc or change in the attitude control law. These discontinuities in the models cannot be introduced within a single phase otherwise, the optimization process might not work properly; the differentiability is a necessary condition inside each integration range.

A typical launcher starts its mission locked to the launchpad: the engines are not yet ignited, or the thrust provided is still lower than the vehicle weight. For solid propulsion the time between the ignition and the liftoff is extremely short (lower than 1 s); in the case of liquid propulsion this time may be longer, which allows for the verification of correct engine operation before releasing the vehicle; such is the case for Falcon 9 [7].

The second phase is a vertical liftoff required to gain velocity before the gravity turn phase and to avoid any contact with the launchpad tower.

The vertical flight ends with the pitch-over maneuver: the launcher turns in the direction of the azimuth required to achieve the target orbit inclination, thus defining the initial plane of the trajectory. The azimuth is defined as the angle between the vehicle's horizontal velocity and the north direction. This maneuver has this name because the pitch, the angle between the main axis of the rocket and the horizon, changes from 90° to lower values. On most launch-pads a restricted azimuth range is allowed, e.g., at Kourou, it is from $-10°$ to $91.5°$. The reason is usually related to safety issues: the ground track of the rocket should not cross any populated area because, in case of aborted missions, fragments may create a risk for people. Should the target orbit require a disallowed launch azimuth, a dog-leg maneuver is performed during the flight: the yaw, i.e., angle between the projection of the main vehicle axis on the horizon and the north direction, changing the ground track considerably. This maneuver drastically reduces the performance of the rocket.

The next phase is called "gravity turn": the rocket pitch angle changes automatically under the action of the gravity acceleration. This is true when the rocket is aerodynamically stable otherwise, the gimbals (i.e., the pivoted supports that allow the rotation of an object about a single axis) of the engine nozzle are required to constantly align the thrust with the velocity vector. Figure 7.1 presents the rocket flying a gravityturn: the thrust acceleration is aligned with the main axis of the vehicle and with the velocity vector; the gravity acceleration pointing towards the center of the planet produces a total acceleration that is not aligned with the vehicle main axis. The direction of the total acceleration causes the velocity vector at the next time step to rotate creating an angle between the main axis of the vehicle and the velocity vector itself (also known as the angle of attack). In case the aerodynamics of the vehicle is stable, the generated moments reduce this angle to zero; unstable aerodynamics requires the thrust control system to perform the same maneuver. The presence of an angle of attack different from zero when the dynamic pressure is higher than 1 kPa generates forces and moments that could lead to the breakup of the rocket. The drag acceleration, not shown in Fig. 7.1, is aligned with the velocity and the thrust acceleration, and thus does not influence the gravity turn.

The gravity turn phase is typical for the first stage and in some cases also for the second stage; the end of the gravity turn phase is normally performed when the dynamic pressure reaches low values (lower than 1 kPa). In case the attitude of the vehicle is not controllable, the full-powered ascent is accomplished following the gravity turn.

As soon as the heat-flux density drops under the typical value of 1,135 W/m^2, at an altitude higher than 100 km, the thermal protection provided by the fairing is not required anymore and is therefore jettisoned.

The third stage and the upper stage in general present a fully optimizable attitude control (pitch and yaw): the solver can modify the attitude profile during the upper stage burn phase, in order to minimize (respectively maximize) the objective function.

In case where the target orbit altitude is higher than the "natural" altitude of the launcher, it is more efficient to insert a coast arc between two burns of the upper stage: the first burn is required to increase the orbit apoapsis up to the target orbit altitude with a periapsis higher than 120 km for safety reasons. The apoapsis and periapsis are, respectively the two points at the largest and smallest distance from the central body in an elliptic orbit. During the coast phase, the rocket reaches the apoapsis, and a second burn is required to obtain a circular orbit at the desired altitude. This approach requires a restartable upper stage, and it is more efficient than a single burn of the upper stage: e.g., the VEGA launcher [8] in a polar circular orbit at an altitude of 500 km has a payload mass increase of 33%. The polar orbit has an inclination of exactly 90°: the orbit thus passes over the poles. The periapsis at the end of the first burn should be at least 120 km so that in case the upper stage engine would not restart at the apoapsis, there will be at least a second full orbit for a second attempt.

7.2.2.2 Objective Function

The objective function is the most important "player" in the optimization game. Apart from the maximization of the payload mass, additional typical objective functions for the optimization of the launcher trajectory are the minimization of the gross lift-off weight (GLOW), the maximization of the orbit altitude, and more recently the minimization of the launcher cost. The latter can be evaluated as a function of the propellant mass and the technological level of the various components of the launcher, engines, and stages.

For a parallel trajectory and vehicle design optimization, the objective function can be dependent on some model parameter, e.g., fairing diameter.

The so-called Chebyshev objective function is quite interesting: it minimizes the highest peak of a specified function (e.g., dynamic pressure) evaluated in the full mission or along a specific phase. This type of objective function is extremely useful in cases when the peak is not located at the end of a phase; this is the case, for instance, for the heat-flux density and the angle of view of a ground station.

$$\min\{\max|f(x,u,t)|\} \approx \begin{array}{c}\min(c)\\ f(x,u,t)\leq c\end{array}. \tag{7.2}$$

The minimization of the peak of a function (f) dependent on the states (x), the controls (u) and the time (t) is rewritten as the minimization of the Chebyshev parameter (c) with an additional path constraint that ensures that the function f is always lower than the parameter c.

7.2.2.3 Constraints

The constraints present in a typical launcher mission can be grouped in four categories:

- Initial boundary constraints, evaluated at the beginning of a phase, typically at the beginning of the mission

- Final boundary constraints, evaluated at the end of a phase, i.e. end of the mission
- Path constraints, evaluated along a phase or all along the full mission
- Parameter constraints, evaluation not linked to a precise time point

An example of initial boundary constraint is the initial position, coincident with the launch-pad coordinates, as well as the initial azimuth range. For an airborne launcher (Pegasus type), the separation velocity and altitude, are initial boundary constraints.

A nontrivial initial boundary constraint is related to the ratio between thrust and vehicle weight: the launchpad clamp should not release the vehicle until this value is higher than 1, otherwise, the rocket falls to the ground.

$$r_{\min} \leq \frac{T}{m_{\text{tot}} \cdot (g - V_h^2/R)}, \tag{7.3}$$

where $r_{\min}$ is the lower bound of the ratio, a typical value is 1.2. T is the thrust, m_{tot} the total mass, g the gravity acceleration, V_h the inertial horizontal velocity, and R the distance between the vehicle and the center of the planet. The last term is the contribution of the centrifugal acceleration.

The launcher mission presents several final boundary constraints: the launch-pad clearance, the fairing separation, the splashdown of jettisoned components, the load factor, and, of course, the target orbit altitude and inclination. It is not enough to reach the required altitude: the nonradial component of the inertial velocity should provide a centrifugal acceleration equal to the gravity acceleration at that altitude in order to achieve a stable orbit.

$$\begin{gathered} V_i = \sqrt{(R_E + h) \cdot g} \\ V_R = 0, \end{gathered} \tag{7.4}$$

where V_i is the inertial velocity, R_E is the Earth radius, h the target orbit altitude, and V_R the radial component of the inertial velocity.

The constraints that should be evaluated during a phase or during the entire mission are related to the dynamic pressure, the heat-flux density (see Sect. 7.4.2), the load factor for a nonconstant thrust profile (solid propellant), and the ground station visibility. In particular, the latter should be enforced along the full trajectory, but it could be relaxed during a long coast arc.

7.3 Re-entry Modeling

This section presents the modeling of a typical reentry mission and of a typical re-entry vehicle itself; the concepts introduced here may be extended, due to their similarities to such vehicles, also to the flying back boosters and the descent phase of a space-plane or sounding rockets.

7.3.1 Vehicle Modeling

In many ways the opposite of the launch vehicle, a reentry vehicle, is an aerospace object whose primary function is bringing a certain mass from a starting orbit to the planet surface. Included in this general definition are capsules, winged vehicles, lifting bodies as well as the descent phase of sub-orbital vehicles and reusable launchers.

The typical reentry vehicle may be modeled as a single component with structural and, optionally, propellant mass. In this type of vehicles the propellant mass is the lower fraction in most of the cases, there is no propellant at all. The vehicle is then composed of the structural skeleton, the electronics, and the payload, plus a heat shield in such cases when the vehicle is designed to survive the reentry.

The pure trajectory optimization of a reentry mission may consider the vehicle as a point mass with the fixed thermal and aerodynamic characteristics used only for forces/moments computation and constraint definition. The situation becomes more interesting in case of a design optimization parallel to the trajectory optimization: here the shape of the vehicle and the thickness of the thermal protection system (TPS) can be modified by the solver with impact on the aerodynamics and on the vehicle mass. As for the launchers, in case of a constraint violation, the solver has the possibility to modify the trajectory or the vehicle itself.

7.3.1.1 Capsule

The first missions with re-entry vehicles were based on capsules. The reason for this resides in the simplified design which is completely focused on the fulfillment of the basic function, namely taking a certain mass from orbit to the planet surface. The shape can vary from spherical bodies (Vostok in 1961—[9]), biconics, truncated cones with a spherical nose to pure cones with a spherical nose (Fig. 7.2). The spherical part is normally in the direction of the flight.

This vehicle has no aerodynamically active surfaces, but it is able to produce a lift force in the same order of magnitude of the drag force in case the velocity is not aligned with the main vehicle axis. The only exception is the spherical capsule: it gives rise to drag force, but no lift.

7.3.1.2 Lifting Body and Winged Vehicle

The shape of the lifting body (e.g., HL20,[10]) and winged vehicle is completely different from the capsules presented in Sect. 7.3.1.1. Interestingly, the modeling of the two is extremely similar: the aerodynamics presents important differences, but the model is almost identical for all three classes of vehicles.

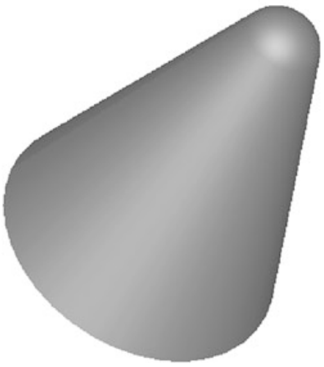


Fig. 7.2 Re-entry vehicle example: cone with spherical nose

These vehicles can have a propulsion system used for the de-orbit phase, a TPS not uniformly distributed on the external surface of the vehicle and a wing plane producing a lift perpendicular to this plane.

7.3.2 Mission Modeling

The starting point of the trajectory is normally located in an orbit around an object. This orbit can be elliptic, parabolic, or hyperbolic. The object is not limited to the Earth, but it could be a planet within the solar system, a natural satellite, or an asteroid. In cases where the initial orbit of the vehicle is stable, e.g., the periapsis altitude is greater than 120 km around the Earth, the vehicle requires a change in velocity in order to start the re-entering trajectory. This delta velocity can be provided by a propulsion system integrated in the vehicle, by a physical characteristic of the vehicle (aero breaking) or by separation from the main vehicle, e.g., lander and orbiter.

The next phase is normally a long coast until the vehicle reaches the atmosphere's upper layers (120 km on the Earth). The following atmosphere phase terminates with the impact on the planet, satellite, or asteroid. The most interesting phase is the atmospheric one: there the most commonly used control method is based on the lift force that the reentry vehicle can provide with a nonzero angle of attack the attitude change of the vehicle can be used to reduce loads (mechanical and thermal) and to modify the trajectory and the impact position. For example the control of the bank angle, the angle between the vehicle plane of symmetry and the local vertical plane, changes the direction of the lift force towards the side of the vehicle. This increases the cross-range of the trajectory.

A multiple parachute deployment could precede the impact when an atmosphere is present. If an atmosphere is not present, the integrated propulsion system could be exploited to reduce the impact velocity and loads to within acceptable limits. The rationale behind the implementation of at least two parachute systems is the wide range of velocities the vehicle exhibits. The first parachute is also called drogue-chute: it is small and robust its function is to reduce the velocities from transonic or high subsonic to low sub-sonic and to stabilize the vehicle attitude. The second parachute system is normally the main one. Being larger, it decelerates the vehicle down to the accepted impact velocity. The modeling of the parachute opening could be critical due to the almost instantaneous change of the aerodynamic drag force; the suggestion is to provide the reference area as a smooth function of altitude and velocity. This model is representative of the reality where an altimeter (or more recently, a GPS) is used to trigger the parachute opening.

7.3.2.1 Objective Function

The reentry vehicle encounters critical conditions during its trajectory down to the planet surface. These criticalities are normally not present at the end of the mission; for this reason it is not possible to use an objective function based on the final value of a function. The Chebyshev objective function is well suited (see Sect. 7.2.2.2). This approach is typically applied to the load factor, to the dynamic pressure, or to the heat-flux density (see Sect. 7.4.2).

A special objective function dedicated to the reentry trajectory optimization is related to the “safety” factor. The plots of the trajectory at the maximum load factor, respectively, maximum dynamic pressure, and maximum heat-flux density create a sort of re-entry corridor. The maximization of the distance between the effective trajectory and the maximum sustainable loads, mechanical and aero-thermal, produces an optimized trajectory of the vehicle in the middle of this corridor. In this case, an unexpected event (uncertainty on the atmosphere characteristics) will not produce a trajectory that violates any of the vehicle constraints.

Another typical objective function is related to the down- and cross-range: the distance between the impact position and the initial position computed respectively along the orbital plane and perpendicular to it. These are good indicators of the efficiency of the vehicle aerodynamics and they provide a potential area that can be reached during the re-entry trajectory. Requesting a final position outside this area produces infeasible optimization runs.

7.3.2.2 Constraints

Most of the constraints have already been cited in this section: load factor, dynamic pressure, as well as heat-flux density evaluated during the entire mission. The final point of the trajectory is also constrained in term, of position, velocity, and attitude: the reason is that in most scenarios, the reentry trajectory stops at the terminal area energy management (TAEM). The last phase to the ground is simulated with other tools and is not part of the optimization task. A useful constraint is related to the distance from the target final position:

$$d_{\max} \geq R_E \cdot a\cos\left[\bar{R}_f \cdot \bar{R}_T\right]. \tag{7.5}$$

where $d_{\max}$ is the maximum acceptable distance, R_E is Earth's radius; R_f and R_T are the normalized vector between the center of the planet and, respectively, the final position of the vehicle and the target final position.

An important class of constraints is related to the attitude control of the vehicle: most of the vehicles can fly only with a reduced range of angle of attack outside of which the vehicle cannot be controlled. This range is normally a function of the Mach number (i.e., the ratio between the vehicle velocity and the speed of sound), then a path constraint is required in order to check the angle of attack versus the Mach number. As an extreme example, some vehicles can fly only at the so-called trimmed condition: the angle of attack is a fixed function of the Mach number without any possibility for optimization.

7.4 Common Physical Equations

The launch and reentry trajectory is a complex scenario that should be modeled using simplified physics equations that describe with sufficient accuracy the subsystems of the vehicle and the environment. During the various phases of the mission, different sets of equation of motion could be implemented in order to consider the sensitivity to some parameters (e.g., the flight-path inclination angle during atmospheric reentry phase) and the particularity of the physical model (e.g., attitude control based on the Euler angles during burn phases).

7.4.1 Aerodynamics

The aerodynamic forces and moments are modeled using a set of coefficients function of the Mach number and the vehicle attitude (e.g., the angle of attack). Each coefficient (C_D) linearly links the square of the vehicle air-path speed (V_a), the atmospheric density (ρ), and the reference area (A_{ref}) with the corresponding force; in the example provided, the drag force (D) is

$$D = \frac{1}{2} \cdot \rho \cdot {V_a}^2 \cdot A_{\text{ref}} \cdot C_D. \tag{7.6}$$

In the case of aerodynamic moments the reference length (l_{ref}) is also present in the equations:

$$M_Q = \frac{1}{2} \cdot \rho \cdot {V_a}^2 \cdot A_{\text{ref}} \cdot l_{\text{ref}} \cdot C_{Mq}. \tag{7.7}$$

In a second level of complexity, two main regimes that these vehicles encounter may be modeled separately, and a bridging function is responsible for a smooth transition between the free molecular flow and the continuous regime. These regimes are identified by the Knudsen number [11] computed at each time step of the vehicle mission.

7.4.2 Aero-Thermodynamics

The interaction between atmosphere and vehicle not only produces aerodynamic effects but also heat flux due to convection and radiation. Depending on the flying regime, which can be supersonic or hypersonic, different models should be applied to compute the heat transmission at the stagnation point at the thermal equilibrium.

For launchers, the fairing protects the payload for the first part of the mission; when the heat flux becomes lower than a critical value, the fairing can be jettisoned, while monitoring the heat-flux to ensure it remains low for the rest of the mission. In this scenario a free-stream enthalpy convective model is fairly accurate:

$$\dot{Q} = \frac{1}{2} \cdot \rho \cdot {V_a}^3, \tag{7.8}$$

with the heat-flux density ($\dot{Q}$) being a function of the atmospheric density (ρ) and the air-path velocity (V_a).

In the case of atmospheric flight in the supersonic regime, more complex models based on Detra–Kamp–Riddel formulation should be used. These consider also the nose radius and different exponents for atmospheric density and air-path velocity. The interested reader can find a comprehensive treatment in [12].

For hypersonic regimes (which are also referred to atmospheric phases) the heat-flux density should be computed with the modified Fay–Riddell formula.

$$\dot{Q} = C\sqrt{\frac{\rho \cdot {V_a}^2}{r_N}} \cdot \left(\frac{{V_a}^2}{2} + 1004(T - T_W)\right). \tag{7.9}$$

Among the already specified elements, this formula also contains a model constant (C), the nose radius (r_N), the free-stream temperature (T) and the wall temperature of the vehicle (T_W). As for the supersonic model, the interested reader is referred to [13].

All the models presented so far refer to the convective heat flux in the case of radiative flux, the specific formulation is also a function of the atmospheric composition. Additional details are outside the scope of the present chapter but may be

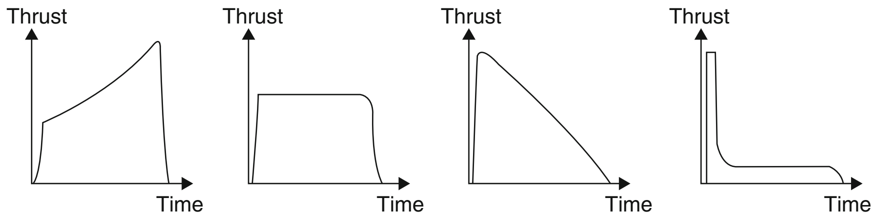


Fig. 7.3 Solid propulsion thrust profiles: progressive, neutral, regressive and exotic

found in [14]. Assuming the vehicle to be in thermal equilibrium, it is possible to compute analytically the wall temperature combining Eqs. (7.9) and (7.10).

$$T_W \approx 4\sqrt{\frac{\dot{Q}}{\sigma \cdot \varepsilon}}, \tag{7.10}$$

where σ is the Boltzmann constant and ε the emissivity.

7.4.3 *Propulsion*

There are two main types of propulsion systems, the first one is based on liquid propellant and the second one on solid propellant. A third option is the so-called hybrid engine: the propellant is present in both states, liquid the oxidizer and solid the fuel.

Liquid propulsion has several advantages; among others it is restartable and throttleable (i.e., capable of having the thrust varied), and it has a higher specific impulse. Solid propulsion on the other hand has higher thrust, higher propellant density, and is both cheaper and more reliable.

In general, liquid propulsion is more flexible, but it requires a more complex engine, and therefore the cost increases whereas reliability is reduced. The propellant mass flow $(\dot{m}_{pb})$ can be kept almost constant along the burn duration, with an effective thrust (T) changing only due to the backward pressure (p_a) of the atmosphere acting on the exit area of the nozzle (A_e).

$$T = I_{\mathrm{sp}} \cdot g_0 \cdot \dot{m}_{\mathrm{pb}} - A_e \cdot p_a \tag{7.11}$$

for solid propulsion, Eq. (7.11) is still valid, and the mass flow, however, is not constant along the burn duration; the respective thrust profile is dependent on the grain geometry. Some examples of thrust profiles are presented in Fig. 7.3.

For an exhaustive description of the solid propulsion, please refer to [15].

7.4.4 Equations of Motion

The motion of the launcher or reentry vehicle can be described with a set of differential equations linearized at each time step. In the literature, there are several sets of equations of motion (EoM), most of these are dedicated to a specific phase of the flight. In general, there are sets based on position and velocity and set based on the orbital elements.

This list is not intended to be exhaustive, but it presents a good overview of the possibilities in this field; the complete mathematical equations may be found in [14].

- Inertial Cartesian: position and velocity are Cartesian components of an inertial reference system centered in the center of the planet; this set is extremely robust, but the sensitivity is poor due to bad scaling; it is required for polar trajectories.
- Inertial velocities: position is defined as longitude, declination, and radius; the velocity components are in the same frame, radial, longitudinal, and along declination; it is the most efficient set for normal launcher trajectories, but it has a singularity over the poles.
- Flight-path velocities: the position components are identical to the inertial velocities; the velocity components are the flight-path speed modulus, flight-path inclination angle, and flight-path heading angle; it has a singularity over the poles where the flight-path speed is zero; due to a dedicated state it is the most efficient in cases where the velocity direction (flight-path angle) is highly sensitive, i.e., re-entry trajectories and atmospheric cruise phase.
- Launch-pad: the position and velocity derivative are zero, only the mass is integrated to account for burned propellant mass.
- Accelerate flight-path: the movement is allowed only along one direction; the accelerations perpendicular to this direction are not considered; this set is intended for a movement along a fixed rail.
- Perturbed Keplerian elements: the states are the six Keplerian elements; in the case of a trajectory outside the influence of the atmosphere, these are more efficient than the set based on the inertial velocity; they have several singularities.
- Equinoctial elements: orbital elements best suited for trajectories outside the influence of the atmosphere; in contrast to the Keplerian elements, this set has no singularities and it is to be preferred to the Keplerian set.

The above list refers to EoM that integrate the position and velocity of the vehicle treated as a point mass they can also be defined as 3 degrees of freedom (DoF). In some special situations, it could be required to integrate the attitude of the vehicle together with the position; these are known as 6 DoF EoM. These sets include position, velocity, attitude angles and attitude angle rates.

The attitude can be defined via Euler angles, i.e., the angles between the horizon and the body axis frame of the vehicle or with aerodynamic angles, the angles between the velocity vector and the body axis frame of the vehicle. Due to affinity, either the flight-path velocities with aerodynamics angles (flight-path 6 DoF) or inertial velocities with Euler angles (inertial velocities 6 DoF) are employed.

Additionally, the inertial velocities can be coupled with the quaternion angle for the definition and integration of the attitude.

The mathematical equations that define these set, are not reported in this chapter, but the interested reader may find them in [14].

7.4.5 Environment

The vehicle ordinarily moves inside an atmosphere subject to a gravity field, together with the wind these models define the environment of the mission. The wind is not of interest for satellite launchers, but it is a key factor for sounding rockets and the final descent phase with a parachute.

There are several models available in the literature for the atmosphere, from the simple analytic but accurate United States (US) Standard 76 [16] to the experimental and complex Earth-GRAM 2010 [17]. The reader should pay attention to the computational speed of each model since the optimization task requires the integration of the full trajectory several times for each step: too complex and too slow a model would hinder the full optimization without achieving interesting results. A good compromise is presented by tabular data for the required functions (pressure, density, and speed of sound), data that reproduce the real conditions at the place and time of the mission.

An analog evaluation should be performed for the gravity field of the central body: a model based on a point mass enriched by series expansion of spherical harmonics is normally used, with the inclusion of the higher terms only. Referring to the Earth, the first coefficient of the Legendre polynomials J2 is quite strong, but for a typical launcher vehicle, all the other coefficients may be neglected.

7.5 Transcription Methods and Mathematical Solvers

Once the real problem is rewritten with simplified physical equations, this model should be transcribed in a mathematical set of algebraic or trascendent equations, in order to be solved by non-linear programming methods (NLP solvers). The class of direct transcription methods discretizes the entire problem on a grid, obtaining a non-linear program. The values of the states and control variables are limited to the grid points. An exhaustive mathematical dissertation can be found in [18].

7.5.1 Historical Overview

Methods for solving trajectory optimization (TO) problems have been developed since the late 1960s. Different methods can be classified using key features.

For instance, such algorithms which require, not only the state, but also the adjoint differential equations, are called indirect methods. Those which do not require the adjoints are termed direct methods. Algorithms which integrate the state or the adjoint differential equations from the initial time to the final time in one run are defined as single shooting methods; those that stop in between at user-specified intermediate grid points and continue from these grid points with user-specified intermediate values for the states are called multiple shooting methods. Furthermore, it is possible to classify the methods according to whether the control, the state time histories, or both of the optimal solution are approximated by some model function, e.g., piecewise linear functions or polynomials. These algorithms solve the parametrized optimal control problem, where the parameters define the control polynomial functions. If both controls and states are approximated, this is known as the collocation method (see Sect. 7.5.3.2).

According to these classifications, the sequential gradient restoration algorithm (SGRA,[19]) is an indirect method, as are the codes AeroSpace trajectory optimization (ASTROP,[20]) and boundary value problem with switching conditions (BNDSCO,[21]). ASTROP has been used at the European Space Operations Center (ESOC) extensively for exo-atmospheric trajectory optimization. The program to optimize simulated trajectories (POST,[22]) uses a single shooting direct method with control parametrization; it has been developed by the US industry. Optimal trajectories by implicit simulation (OTIS,[23]) uses a direct collocation method and has been developed by the US industry as well. Both these codes are in widespread use at the National Aeronautics and Space Administration (NASA) and at various US government laboratories as well as US universities, but they are not in general available to European organizations. Sparse optimal control software (SOCS,[24]) is an extremely fast direct collocation code, developed by J. Betts and commercialized by Boeing. Optimisation de performances Ariane (OPERA) is a direct collocation code developed by the Office National d'Etudes et de Recherches Aérospatiales (ONERA) which also has seen widespread use in rocket trajectory optimization. TOMP [25] is a direct method, using single shooting, similar to the method adopted by POST. Multiple shooting code for direct optimal control (MUSCOD,[26]) is the first direct method that combines control parametrization with multiple shooting. Trajectory optimization using direct collocation (TROPIC) and parameterized trajectory optimization by direct multiple shooting (PROMIS) are codes initially developed by DLR [27, 28]. TROPIC is a direct collocation code similar to the one used within OTIS but has additional features such as automatic function-and-parameter scaling. PROMIS is a similar method to MUSCOD but also has parameter-and-function scaling and a very flexible problem interface. Collocation and multiple shooting trajectory optimization software (CAMTOS,[29]), developed by the University of Stuttgart and maintained by Astos Solutions, is a hybrid optimizer which allows the choice of collocation and multiple shooting in each phase. In a recent evolution (2011), the transcription method CAMTOS can utilize the European non linear programming (eNLP,[30]) solvers in parallel with the Stanford University sparse nonlinear optimizer (SNOPT, [31]).

7.5.2 Transcription and Solver Selection

The selection of the "best" code for a particular problem depends on the complexity of the problem, on the level of user know-how in optimization theory, on the user expertise and intent, and on features of the candidate code such as efficiency, achievable accuracy, robustness, ease of use, and convergence behavior. For problems of moderate size and complexity, indirect methods such as multiple shooting with its implementation BNDSCO, have been applied successfully, especially since recently a direct collocation method has been introduced as a "preprocessor" which has made the cumbersome task of estimating the values for the Lagrange multipliers much easier, see, for instance, [32]. The level of know-how required to use this technique is nevertheless very high, since the user needs to implement the necessary optimality conditions for each individual problem and thus needs a priori knowledge of the sequence of sub-arcs of the optimal solution. In addition, the right-hand sides of the system of differential equations must be sufficiently differentiable which requires some effort in interpolation and approximation. Once these initial difficulties (usually 95% of total time spent in solving a particular problem) are overcome, indirect methods such as BNDSCO or ASTROP are extremely accurate and efficient—better than any other technique. Accordingly, this approach is to be favored if, for instance, the user intent is to generate a large number of solutions for varying boundary conditions.

On the other side of the spectrum is the class of direct methods that are based on parameterization of the control functions or on collocation. These techniques require less knowledge of optimization theory, no a priori knowledge of the type and sequence of sub-arcs of the optimal solution. They are robust, that is to say, relatively insensitive to nonsmoothness of the mathematical models (right-hand side of the differential equations). In addition, they converge to an approximation of the solution even from rather inexact initial estimates. Numerical experiments have shown that in this respect TROPIC performs better than TOMP or PROMIS. Further numerical experiments with MUSCOD and PROMIS have shown that multiple shooting enlarges the convergence region in comparison to single shooting (TOMP) considerably. These classes of methods are, however, less efficient and accurate than the indirect approaches. Despite these deficiencies, these techniques are usually preferred by non-expert users (for instance, engineers) for solving complex and even large-sized problems. The limit on problem size is determined by the maximum number of variables and constraints that the available NLPcodes can handle.

Once the transcription method has converted the simplified physical model into a set of equality constraints, the NLP solver can be applied to identify the optimal control that minimizes the cost function. The solver can be a global optimizer, e.g., genetic algorithm, particle swarm, some alternative heuristic or a sequential quadratic programming (SQP) when dealing with differentiable functions. In the specific case of launchers trajectory optimization, the typical problem presents around one thousand optimizable parameters and constraints with a sparse Hessian matrix. These characteristics fit the capabilities of the SQP solvers, leaving the application of global optimizer to rare cases.

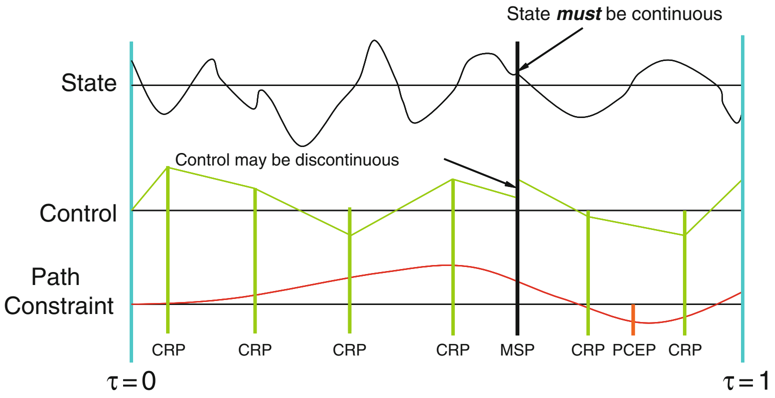


Fig. 7.4 Transcription within a phase for multiple shooting approach

7.5.3 Direct Transcription Method Technical Details

The transcription method is intended as the interface between the simplified physical equations presented in Sect. 7.4 and the mathematical SQP solver. Two examples of transcription methods are presented, the first one implementing a multiple shooting approach and the second a collocation approach.

7.5.3.1 Multiple Shooting

As learned in Sect. 7.2.2.1, the trajectory is separated into several phases; for each phase a major grid is defined that contains the multiple shooting nodes (MSP in Fig. 7.4): the states are integrated between two nodes with the possibility to have a discontinuity between the final integration and the value stored in the next node. These discontinuities will be removed by the solver during the optimization process, but their presence allows flexibility that increases the robustness of the integration and the convergence radius as opposed to a single shooting approach. It is important to accurately select the number of nodes for each phase, a rule of thumb valuable for launchers suggests the insertion of a major grid node only in phases longer than 50 s and around one node every 100 s for longer phases. A high number of nodes increases the number of optimizable parameters and the time required by the solver to obtain a solution.

Each phase contains also a control refinement grid for each control function; the control is discretized, interpolating it along these nodes (CRP in Fig. 7.4). Several nodes are therefore required to compute a complex control profile.

Each phase has a third grid containing the constraint evaluation nodes: the path constraints are not evaluated at each time step as this would be too time consuming, but only at the nodes present in the constraint grid (PCEP in Fig. 7.4). It is good practice to insert the constraint nodes in correspondence with the major grid and the control refinement grid nodes (i.e., at the same normalized time).

The mathematical problem transferred to the solver is made by the states at each major grid node and the parameters; the constraint vector is incremented with the continuity condition at each MSP.

7.5.3.2 Collocation

Similar to the multiple shooting, for each phase a major grid is defined that contains the collocation nodes. In contrast to multiple shooting, the states are not integrated but approximated with polynomials; these polynomials are created on the basis of the state values at the major grid nodes. The discontinuity at each collocation node is automatically checked by the equality constraints between the polynomials and the real state. Additionally, the equality between the first derivative of the real state and the first derivative of the polynomials is enforced not only at each collocation node, but also at the midpoint of each collocation interval.

Violation of these constraints will be removed by the solver during the optimization process. It is important to note that a large number of collocation nodes are required to provide a realistic approximation of each state; a rule of thumb valuable for launchers is to insert five major grid nodes in short phases (e.g., vertical liftoff) and between 30 and 50 nodes in longer phases. For reentry vehicles, the atmospheric phase is typically too long for just 50 nodes; one solution could be to insert 20 nodes every 100 s with the possibility to refine the major grid during the optimization. A larger number of nodes require a larger number of optimizable parameters, and thus the time required by the solver to obtain a solution increases accordingly. The task of the user is therefore to identify the minimum number of nodes that provide a realistic solution.

The collocation transcription method uses the major grid nodes for the control approximation as well, and no additional control refinement grid is required.

For each phase there is a path constraint evaluation grid: the path constraints are evaluated only at the nodes present in this grid. It is good practice to insert one constraint node for each major grid node.

The mathematical problem passed to the solver is given by the states at each major grid node and by the parameters; the constraint vector is incremented with the collocation constraints active on the state and on their first derivatives.

An important point that should be noted is that the solution "seen" by the solver is formed only by polynomials: each state is correct with respect to the major grid nodes, but it could be completely different from the reality inside each collocation interval. It is therefore required to compare the polynomial solution with the real state history obtained by integrating the full trajectory in one shot. Should the two trajectories differ too greatly, a denser major grid is required.

A detailed mathematical explanation may be found in [33].

Table 7.1 Comparison between direct multiple shooting and collocation

Direct multiple shooting	Collocation
Slow, due to state integration with variable step size	Faster (no "integration")
Accuracy of the state integration is sufficient in most cases	Not as accurate, accuracy should be verified with a single shooting simulation
Smaller convergence radius	Larger convergence radius
Less robust	More robust

7.5.3.3 Comparison Between Collocation and Multiple Shooting

Table 7.1 summarizes the main characteristics of the direct multiple shooting and collocation transcription method.

7.6 Trajectory Optimization Examples

Several examples of launch and reentry vehicle problems are presented hereafter, with a strong emphasis on the advantages and disadvantages of the various transcription methods and solvers when applied to "real" problems.

The examples exploit the wide application of this field: from a simple launch to escape orbit and the performance map of a polar launcher, over the design of a new family of launchers with a multimission approach to the SpaceLiner [34] ascent, cruise, and approach to Amsterdam. The re-entry of the atmospheric reentry demonstrator (ARD,[35]) completes the chapter.

7.6.1 Conventional Launcher Examples

Two conventional launcher examples are proposed: Ariane 5 to an escape orbit and the performance map of VEGA in polar orbits.

7.6.1.1 Ariane 5 to Escape

The typical launcher trajectory optimization exercise is the maximization of the payload mass for a given vehicle and target orbit: e.g., compute the payload mass of Ariane 5 on an escape orbit. This example is typically solved with inertial velocity EoM for all the phases. There are no coast arcs, so an attitude control based on Euler angles (pitch and yaw) provides the best performance. These angles link directly the main force, i.e., the thrust aligned with the main vehicle axis, with the states, i.e., the velocity components aligned with horizon and radius. The first phases until the jettison of the solid booster use the gravity turn control law; the last part of the main stage burn and all the upper stage phase present instead a fully optimized attitude control. While both transcription methods may be applied, the relatively short integration time allows for the use of the more accurate multiple shooting method.

Apart from the typical launcher constraints, two challenges are particular for this mission: the full coverage from ground stations and the impact point of the main stage of Ariane 5. The ground track starting from Kourou crosses the Atlantic Ocean; a single ground station cannot cover the entire mission therefore, a network of stations with overlapping visibility is required to ensure safety margins. These stations are situated on several islands, depending on the target inclination however, the coverage may not be complete; an interesting exercise for the solver would be the modification of the trajectory to ensure full coverage. Alternatively, the optimization could identify the most convenient location for a mobile station on a ship. The second challenge is the impact position of the main stage: a trajectory optimization which does not take this into account could produce an impact location on land, namely Africa or Europe depending on the target inclination. This situation is not acceptable therefore, the solver modifies the trajectory to move the impact position in the Atlantic Ocean. Additionally, the walking impact line of the upper stage can be analyzed: the line formed by the instantaneous impact position of the upper stage in case of an abort mission. The best design could allow this line to lay only in the Atlantic Ocean; this is possible since the impact line is defined only with a periapsis altitude less than zero.

7.6.1.2 VEGA Performance Map

Adding a level of complexity it is possible to compute the performance map of a different launcher, VEGA, in polar circular orbit at different altitudes. The problem is still related to the maximization of the payload mass, but there are some important differences in the model that require modifications also in the transcription method. The polar orbit has an inclination of 90° and the vehicle could pass over the north or south pole, where the inertial velocities EoM, in particular the longitudinal component, are not defined (see also Sect. 7.4.4). This configuration cannot be analyzed by the solver; therefore a different set of equation of motion is required, namely inertial Cartesian. The application of this set to the entire trajectory is not efficient due to the poor scaling of the velocity and position components. A better approach would be to apply the Cartesian EoM only in the phases that cross the pole, whereas the other phases are modeled with inertial velocities EoM.

Additionally, VEGA presents a restartable upper stage, and for orbit altitude higher than 300 km, the implementation of a two burn strategy is beneficial. This leads to the presence of a coast arc with a duration dependent on the final orbit altitude, e.g., 1,800 s for an orbit altitude of 700 km. This long ballistic phase presents no attitude control, it is a simple transfer to the apoapsis, but the numeric integration required by the multiple shooting method is quite time consuming due to the long duration. A performance improvement can be achieved with a collocation method applied only to this long phase: the advantages of both methods can be appreciated, the accuracy of multiple shooting and the fast computation of the collocation.

7.6.2 Multi-mission Optimization

An interesting example is provided in the design of a full family of launchers, where a simple two stage to orbit (TSTO) system is enriched by a variable number of strap-on boosters to target different orbits or payload masses. The frame is the future launcher preparatory program (FLPP,[36]) of the European Space Agency (ESA); this program aims at providing a sound substitute to the current European launchers. The new vehicle family should have the flexibility to cover the entire range of payload in the institutional and commercial markets. An intelligent solution is the optimization of all the required missions in a parallel run with the various configurations linked by parameter constraints: constraints that act on the model of the vehicle. In this scenario, the stage masses and the attitude controls are optimized in order to achieve the most efficient solution in all the considered missions. From a computational point of view, the dimension of the optimization problem depends linearly on the number of missions that are optimized in parallel. As discussed above, it is best to keep this number as low as possible.

Some effort is required in order to define a sound objective function formed by various terms which attempt to evaluate the quality of the launcher family. Unfortunately, a clear and comprehensive metric to define the launcher performance does not exist, so the reader should analyze each specific case and apply his knowledge and intelligence to identify the key elements.

7.6.3 Sub-orbital Space-Plane

In recent years, there has been a particular interest in fast transport systems which allow a long track (e.g., AustraliaEurope) in a fraction of the time required by a normal airplane. Such a mission comprises an ascent and a reentry trajectory making this scenario a challenging optimization problem. In particular, SpaceLiner consists of two main stages: a booster and a winged orbiter that hosts the crew and passengers. The first phases are identical to a normal launcher mission with a vertical liftoff, pitch-over, and gravity turn control law. At the separation between booster and orbiter, the latter activates its engine with a low pitch angle to maximize the downrange of the mission. After the orbiter main engine cutoff

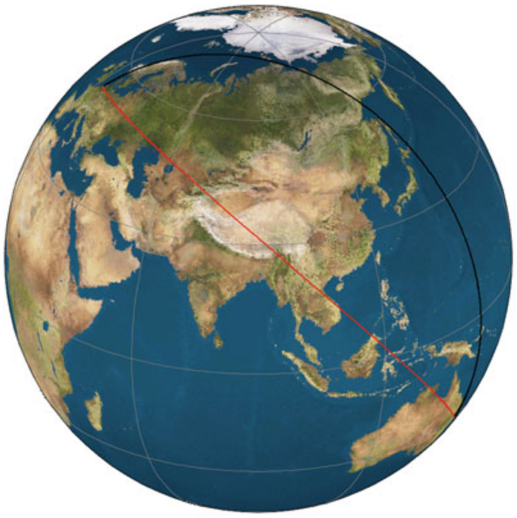


Fig. 7.5 SpaceLiner Sydney–Amsterdam trajectories, the initial guess along the great circle and the fully optimized versus north

(MECO), a long cruise phase (approximately 5,000 s) enables the vehicle to reach its final destination. This phase is defined as "cruise" instead of "coast" because it is below 60 km of altitude and the aerodynamic forces allow for an optimizable attitude control of the vehicle.

Similarly to the example presented in Sect. 7.6.1.2 it can be noted that a collocation method applied to this long phase will improve the optimization performance, but a novel aspect should be mentioned: there is no thrust available. The previous examples with optimizable attitude control are based on Euler angles defined between the main axis of the vehicle and the horizon. This set is the most indicated in the case of phases with active propulsion systems, normally aligned with the main vehicle axis; but in this case, the implementation of a new set of attitude control angles is more convenient. The available control forces during the cruise phase are only the aerodynamic ones, then the angle of attack and bank angle (angle between the plane of the wings and the horizon) should be implemented. Additionally, the flight-path velocity EoM are more efficient during the optimization process, due to the sensitivity of the flight-path inclination angle in this type of trajectories.

The optimized trajectory is able to avoid any fluctuations during the cruise phase (skips) associated with a remarkable propellant mass reduction provided by the capability to fly out of the plane that contains the starting and final position (initial guess in Fig. 7.5). The reason for this is that in flying from Australia to Europe the vehicle has to additionally counteract the Earth rotation, then a reduced initial azimuth (almost to the North Pole in Fig. 7.5) minimizes the Earth rotation effect increasing the required range.

7.6.4 Atmospheric Re-entry Demonstrator

This vehicle is part of a controlled reentry mission performed in Europe with recovery of the capsule. The vehicle is a conical spheroid, similar to the Apollo capsule with sets of thrusters to control the attitude angles.

The mission performed in 1998 was suborbital, but an interesting exercise could be the optimization of the reentry trajectory from the ISS orbit: a low earth orbit with 51° inclination. Due to the high orbit inclination, the landing position could be set on a wide geographical belt. After a short deorbit phase with the optimization active on the duration and direction of the thrust force, the vehicle starts its descent until the encounter with the atmosphere at an altitude of 120 km. During the atmospheric phase, the optimizer becomes active on the attitude control of the vehicle. The deorbit phase can be solved with a multiple shooting, method, whereas the atmospheric phase is more efficient with collocation. The objective function is the safety of the trajectory in the reentry corridor (see Sect. 7.3.2.1).

A parametric study can be performed linking the periapsis altitude at the end of the deorbit impulse with the landing dispersion area extension. The latter can be computed by perturbing the atmospheric characteristics, the aerodynamic coefficients and the initial state.

A typical maneuver during a controlled reentry is the bank reversal (Fig. 7.6). The vehicle flying at a zero bank angle presents the lift in the vertical plane, and this produces the highest down-range capability. It is clear that it is not possible to use this setting: a non-nominal parameter could not be compensated. The solution is to fly with a bank angle always higher than 10° or 20°; this will create a sort of performance reserve that would allow the compensation for non-nominal parameters (e.g., higher atmospheric density). The side effect is that the trajectory will deviate from the orbital plane creating a form of spiral; it is then required to change the sign of the bank angle from time to time. The final ground track then resembles a large “S.”.

7.7 Conclusions

The optimization of launchers and reentry missions is undoubtedly an interesting task that requires a solid understanding of aerospace, physics, and mathematics. Both classes of vehicles have in common their interaction with the planet atmosphere during the completion of their mission. The required simplifications introduced during the modeling exercise are instrumental in improving the performance of the optimization task. With the help of examples taken from real world applications, the author analyzes and solves several scenarios from a numerical point of view. These missions could be used as a template in solving future challenges in this aerospace field.

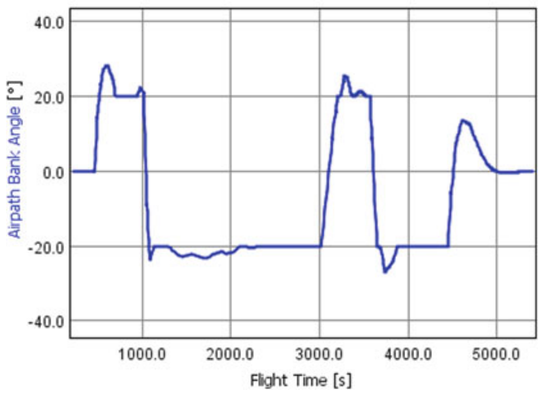


Fig. 7.6 Example of bank reversals during a controlled reentry

References

1. Whitmore, S.: Interim access to the international space station using evolved expendable launch vehicles. J. Spacecraft Rockets, 0022–4650 **47**(3), 503–520 (2010)
2. Varvill, R., Bond, A.: The SKYLON Spaceplane—progress to realisation. J. Br. Interplanet. Soc. (JBIS) **61**, 412–418 (2008)
3. Spies, J., Kuczera, H.: The sub-orbital hopper—one of FESTIP's preferred concepts. In: AIAA-1999-4945, AIAA International Space Planes and Hypersonic Systems and Technologies Conference, 9th, Norfolk, VA, 1–5 Nov (1999)
4. Hall, R.D., Shayler, D.J.: Soyuz: A Universal Spacecraft. Springer Praxis Books. Praxis/Springer, New York (2003)
5. Tsiolkovsky, KE.: The Exploration of Cosmic Space by Means of Reaction Devices. Sci. Rev. (5) (1903)
6. Lockney, D.: NASA's Space shuttle: perspectives on technology transfer. In: AIAA-2010-8885, AIAA SPACE Conference and Exposition, Anaheim, California, 30 Aug–2 Sept (2010)
7. Koenigsmann, H., Musk, E., Gurevich, G.: The Falcon launch vehicle. IAC-03-V.1.08. In: 54th International Astronautical Congress of the International Astronautical Federation, Bremen, Germany, 29 Sept–3 Oct (2003)
8. Lafranconi, R., Lopez, M.: VEGA The Small Launcher for Europe. ESA Publications Division, Noordwijk (2005)
9. Hall, R., Shayler, D.: The Rocket Men: Vostok & Voskhod, the First Soviet Manned Spaceflights. Springer, Berlin (2001). 350pp
10. Ehrlich, C.F.: HL-20 concept—design rationale and approach. J. Spacecraft Rockets, 0022–4650 **30**(5), 573–581 (1993)
11. Bird, G.A.: Molecular Gas Dynamics and the Direct Simulation of Gas Flows. Oxford University Press, Oxford (1994)
12. Detra, R.W., Kemp, N.H., Riddell, F.R.: Addendum to heat transfer to satellite vehicles reentering the atmosphere. Jet Propulsion **27**(12), 1256–1257 (1957)
13. Fay, J.A., Riddell, F.R.: Theory of stagnation point heat transfer in dissociated air. J. Aeronaut. Sci. **25**, 73–85 (1958)
14. Wiegand, A., Cremaschi, F., et al.: ASTOS Model Library, Version 7.0.2. Astos Solutions GmbH, Stuttgart (DE) (2011)

15. Sutton, G.P.: Rocket Propulsion Elements. Wiley, New York (1992)
16. COESA: U.S. Standard Atmosphere 1976. NOAA, NASA, US Air Force, Washington (1976)
17. Justus, C.G., Duvall, A., Johnson, D.L.: Adv. Space Res. **34**, 1731 (2004)
18. Büskens, C., Wassel, D.: The ESA NLP solver WORHP. In: Fasano, G., Pinter, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
19. Miele, A.: Recent advances in gradient algorithms for optimal control problems. J. Optim. Theor. Appli. **17**(5, 6), 361–430 (1975)
20. Dixon, L.C.W., Bartholomew-Biggs, M.C.: Adjoint-control transformations for solving practical optimal control problems. Optim. Contr. Appl. Meth. **2**, 365–381 (1981)
21. Bulirsch, R.: Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung. Carl-Cranz-Gesellschaft, Heidelberg (1971)
22. Brauer, G.L., Cormick, D.E., Stevenson, R.: Capabilities and Applications of the Program to Optimize Simulated Trajectories (POST). NASA CR-2770, Washington (1987)
23. Hargraves, C.R., Paris, S.W.: Direct trajectory optimization using nonlinear programming and collocation. J. Guidance Contr. Dynam. **10**(4), 338–342 (1987)
24. Betts, J.: SOCS User Guide Release 7.1. The Boeing Company, Chicago (2009)
25. Kraft, D.: FORTRAN-Programme zur numerischen Loesung optimaler Steuerungsprobleme. In: Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (DFVLR), IB 515-80/3 (1980)
26. Bock, HG., Plitt, KJ.: A multiple shooting algorithm for direct solution of optimal control problems. In: Preprints of IFAC International Federation of Automatic Control, 9th World Congress, 2–6 July, pp. 243–247 (1984)
27. Jänsch, C., Paus, M.: Aircraft trajectory optimization with direct collocation using movable gridpoints. In: Proceedings of the 1990 American Control Conference, San Diego, 23–25 May, pp. 262–267 (1990)
28. Schnepper, CA: Large grained parallelism in equation-based flowsheeting using interval newton / generalized bisection techniques. Dissertation, University of Illinois (1992)
29. Gath, PF., Well, KH.: Trajectory optimization using a combination of direct multiple shooting and collocation. In: AIAA 2001–4047, AIAA Guidance, Navigation, and Control Conference, Montréal, Canada, 6–9 Aug 2001 (2001)
30. Erb, S.: eNLP: Application-centric NLP-based optimization in the Aerospace market. ITN Sadco First Industrial Workshop, Paris, 2 Mar (2011)
31. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM J. Optim. **12**, 979–1006 (2002)
32. Stryck, O.V., Bulirsch, R.: Direct and indirect methods for trajectory optimization. J.C. Baltzer AG, Scientific Publishing Company. Annal. Oper. Res. **37**, 357–373 (1992)
33. Becerra, V.M.: Practical direct collocation methods for computational optimal control. In: Fasano, G., Pinter, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
34. Sippel, M.: Promising roadmap alternatives for the SpaceLiner. Acta Astronaut. **66**, 11–12 (2010)
35. Macret, JL., Leveugle, T.: The ARD (atmospheric re-entry demonstrator) program—an overview. In: AIAA-1999-4934, AIAA International Space Planes and Hypersonic Systems and Technologies Conference, 9th, Norfolk, VA, 1–5 Nov (1999)
36. Pilchen, J et al.: Future launchers preparatory programme (FLPP)—preparing for the future through technology maturation and integrated demonstrators status and perspectives. In: IAC-08-D 2.5.2, 59th International Astronautical Congress Glasgow, UK (2008)

Chapter 8
Global Optimization of Interplanetary Transfers with Deep Space Maneuvers Using Differential Algebra

Pierluigi Di Lizia, Roberto Armellin, Francesco Topputo, Franco Bernelli-Zazzera, and Martin Berz

Abstract In this chapter, differential algebra is used to globally optimize multi-gravity assist interplanetary trajectories with deep space maneuvers. A search space pruning procedure is adopted, and the trajectory design is decomposed into a sequence of sub-problems. As far as differential algebra is used, the objective function and the constraints are represented by Taylor series of the design variables over boxes in which the search space is divided. Thanks to the polynomial representation of the function and the constraints, a coarse grid can be used, and an efficient design space pruning is performed. The manipulation of the polynomials eases the subsequent local optimization process, so avoiding the use of stochastic optimizers. These aspects, along with the efficient management of the list of boxes, make differential algebra a powerful tool to design multi-gravity assist transfers including deep-space maneuvers.

Keywords Global optimization • Multi-gravity assist transfer • Deep-space maneuver • Search space pruning • Differential algebra

P. Di Lizia (✉) • R. Armellin • F. Topputo • F. Bernelli-Zazzera
Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Via La Masa 34, 20156 Milano, Italy
e-mail: dilizia@aero.polimi.it; armellin@aero.polimi.it; topputo@aero.polimi.it; bernelli@aero.polimi.it

M. Berz
Department of Physics and Astronomy, Michigan State University, East Lansing MI 48824, USA
e-mail: berz@msu.edu

List of Acronyms

DA	Differential algebra
DSM	Deep space maneuver
FP	Floating point
GASP	Gravity assist space pruning
MGA	Multi-gravity assist

8.1 Introduction

The preliminary design of impulsive interplanetary transfers is usually carried out in the frame of the patched-conics approximation. Within this context, different conic arcs are linked together to define the whole transfer trajectory. The patched-conics method allows the designer to define multiple gravity assist (MGA) transfers. MGA trajectories are usually made up of a sequence of planet-to-planet transfers in which the spacecraft exploits each planet encounter to achieve a velocity change. This method is well established in astrodynamics, and several past missions have used MGA trajectories to reach both inner and outer planets.

In the last two decades, mission designers have exploited the benefits of approaching complex MGA problems from a global optimization standpoint. Nowadays, the aim of the trajectory design is not only to find a solution, but also to find the best solution in terms of propellant consumption, while still achieving the mission goals. In the formalism of global optimization, this means that the problem consists in looking for the optimal solution in those regions of the search space that satisfy the problem constraints. Unfortunately, the MGA problems are characterized by an objective function with a large number of clustered minima, which are prevalently associated to the complex relative motion of the planets and to the nonlinearities governing the simple Kepler problem. This causes local optimization methods to converge to local minima. Hence, despite their efficiency, they should be avoided when looking for the global minimum of a MGA problem, at least in the first stage of the search process.

Extensive work has been devoted to address the global optimization of MGA transfers with impulsive maneuvers. This is mainly done by applying stochastic [1, 2, 3], branch and bound [4], meta-model-based [5], and combined [6] methods. Although some of them showed good performances, they tend to be computationally inefficient if not tailored on the MGA problem and on the structure of its search space.

The gravity assist space pruning (GASP) is a global optimization method that addresses this issue. GASP relies on a systematic evaluation of the objective and constraint functions on a grid of points distributed over the search space. The constraints are used to efficiently prune the search space [7]. Thanks to the particular class of interplanetary transfers solved by GASP, the planet-to-planet arcs making up the whole transfer are treated independently, and forward and backward constraining is applied. This way, the search space is preprocessed, and global optimization algorithms are employed in the reduced domain [8].

The class of MGA transfers formulated above does not cover all possible trajectories for a chemical-propelled spacecraft. An important option to take into account is the introduction of deep-space maneuvers (DSM). DSM are impulsive maneuvers, usually carried out between two planet encounters to improve the performances of a trajectory. When DSM are included into a MGA transfer, the resulting trajectory is usually referred to as MGA-DSM.

Unfortunately, pruning the solution space of MGA-DSM transfers is not trivial. First of all, the increased number of variables and the larger search space inhibits the use of a systematic approach to the pruning process. Moreover, local minima tend to proliferate, which makes difficult the detection of big prunable regions. Thus, it is necessary to rethink the whole pruning process implemented in GASP, and to reformulate it when DSM are included.

Differential algebra (DA) is proposed in this chapter as a valuable tool to address this task. Differential algebra serves the purpose of automatic differentiation, i.e., the accurate computation of the derivatives of functions in a computer environment. This goal is actually achieved by replacing the classical implementation of the real algebra with the proper implementation of a new algebra based on Taylor polynomials. Given a generic function f of v variables, the Taylor expansion of f up to any desired order k can be easily obtained from a computer algorithm that implements its evaluation.

The main idea behind the introduction of DA techniques into the pruning process is the substitution of the pointwise evaluation of the constraints, typical of GASP, with the computation of their Taylor expansions with respect to the design variables. The Taylor expansions are used to approximate the functions over boxes of the search space, and polynomial bounders are then exploited to estimate their ranges within each box. Consequently, the pointwise approach proposed in GASP can be substituted by a sampling process relying on box samples. This results in the possibility of enlarging the grid for the domain discretization, and reducing considerably the computational burden.

The chapter is organized as follows. A short description of the method underlying GASP is given in Sect. 8.2. Then, the implementation of a DA-based GASP algorithm is presented in Sect. 8.3. The introduction of DSM is addressed in Sect. 8.4. The performances of the resulting algorithm are assessed in Sect. 8.5, and some final remarks conclude the chapter.

8.2 Gravity Assist Space Pruning

An MGA transfer is modeled in GASP as a sequence of conic arcs, each patched to the subsequent one by a powered gravity assist maneuver. Consequently, a transfer involving n planets is a n-dimensional problem, as n epochs are needed to identify the position of the planets at each encounter. The main idea behind GASP is to split

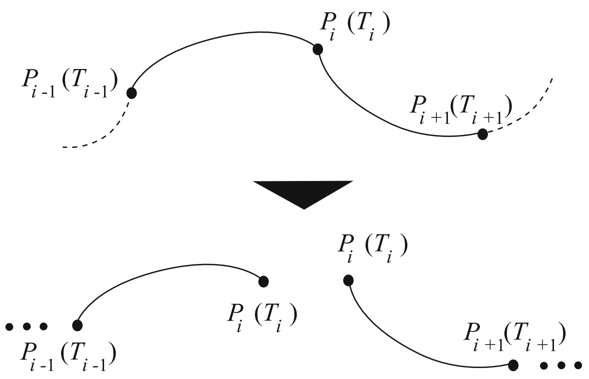


Fig. 8.1 Reduction of the MGA transfer to a cascade of two-dimensional subproblems

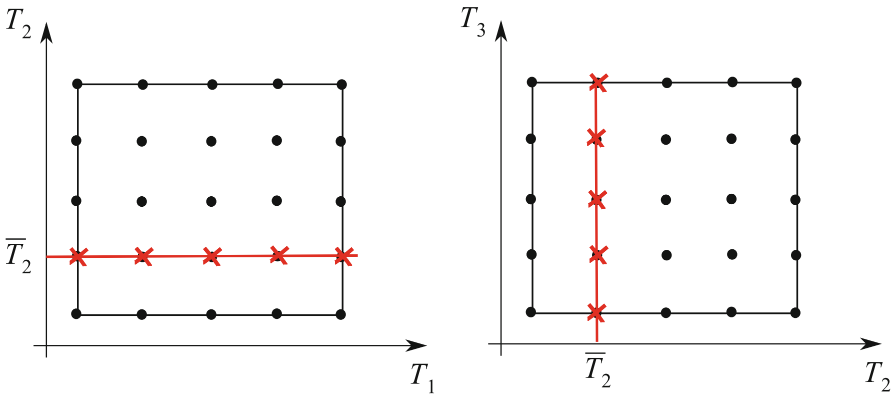


Fig. 8.2 Constraint propagation mechanism in GASP

the whole trajectory in its elementary arcs. With reference to Fig. 8.1, if P_i and T_i, $i = 1, \ldots, n$, are used to denote the planets and the epochs of the corresponding encounters, respectively, the arc connecting P_i to P_{i+1} can be treated as a two-dimensional subproblem with variables T_i and T_{i+1}.

For each subproblem, three constraints are imposed:

- Maximum ΔV at departure (first arc only) and arrival (last arc only).
- Maximum ΔV at gravity assist.
- Minimum pericenter radius at gravity assist.

These constraints can be profitably used to prune the search space. Consider, as an example, the first two arcs of an MGA transfer. These are characterized in the (T_1, T_2) and (T_2, T_3) spaces, respectively (see Fig. 8.2). A uniform grid of points is built to sample each of the search spaces. For each point in (T_1, T_2), the constraints of the P_1–P_2 transfer are evaluated. If any constraint is violated, the point is pruned away, together with all its subsequent combinations with the remaining epochs. In particular, if an entire row corresponding to $T_2 = \overline{T}_2$ yields unfeasible constraints in (T_1, T_2), the entire column corresponding to $T_2 = \overline{T}_2$ in (T_2, T_3) is pruned away. Similar statements hold for the subsequent arcs. This process is called forward constraint propagation. Analogously, a backward constraint propagation can be implemented. The final result is a reduced search space including only feasible regions, where optimization tools are run. The reduced dimension of the search space improves the performances of the optimization algorithms (the reader may refer to [7, 8] for details).

8.3 Gravity Assist Space Pruning with Differential Algebra

The use of differential algebra has been proposed in [9] to improve the performances of GASP. In the DA-based implementation of GASP, the search space is split into boxes, which are processed in place of grid points. More specifically, the point-wise evaluation of the constraint functions is substituted by the computation of their Taylor expansion over the sampling boxes. A polynomial bounder is then used to estimate the ranges of the functions within each box and to prune away unfeasible boxes. The formulation of the algorithm GASP into the DA framework is briefly described in this section. The reader may refer to [9] for additional details.

8.3.1 Notes on Differential Algebra

Differential algebra finds its origin in the attempt to solve analytical problems by an algebraic approach [10]. Historically, the treatment of functions in numerics has been based on the treatment of numbers, and the classical numerical algorithms are based on the mere evaluation of functions at specific points. DA techniques rely on the observation that it is possible to extract more information on a function than its mere values. The basic idea is to bring the treatment of functions and the operations on them to the computer environment in a similar way as the treatment of real numbers. Referring to Fig. 8.3, consider two real numbers a and b. Their transformation into the floating-point representation, $\overline{a}$ and $\overline{b}$, respectively, is performed to operate on them in a computer environment. Then, given any operation $*$ in the set of real numbers, an adjoint operation $\circledast$ is defined in the set of floating-point (FP) numbers so that the diagram in Fig. 8.3 commutes (The diagram commutes approximately in practice due to truncation errors.). Consequently, transforming the real numbers a and b into their FP representation and operating on them in the set of FP numbers returns the same result as carrying out the operation in the set of real numbers and then transforming the achieved result in its FP representation.

$$\begin{tikzcd}
a, b \in R \arrow[r, "\mathcal{T}"] \arrow[d, "*"'] & \overline{a}, \overline{b} \in FP \arrow[d, "\circledast"] \\
a * b \arrow[r, "\mathcal{T}"'] & \overline{a} \circledast \overline{b}
\end{tikzcd}
\qquad
\begin{tikzcd}
f, g \arrow[r, "\mathcal{P}"] \arrow[d, "*"'] & F, G \arrow[d, "\circledast"] \\
f * g \arrow[r, "\mathcal{P}"'] & F \circledast G
\end{tikzcd}$$

Fig. 8.3 Analogy between the floating-point representation of real numbers in a computer environment (*left figure*) and the introduction of the algebra of Taylor polynomials in the differential algebraic framework (*right figure*)

In a similar way, let us suppose two k-differentiable functions f and g in v variables are given. In the framework of differential algebra, the computer operates on them using their k-th order Taylor expansions, F and G, respectively. Therefore, the transformation of real numbers in their FP representation is now substituted by the extraction of the kth order Taylor expansions of f and g. For each operation in the space of k-differentiable functions, an adjoint operation in the space of Taylor polynomials is defined so that the corresponding diagram commutes, i.e., extracting the Taylor expansions of f and g and operating on them in the space of Taylor polynomials returns the same result as operating on f and g in the original space and then extracting the Taylor expansion of the resulting function.

The straightforward implementation of differential algebra in a computer allows to compute the Taylor coefficients of a function up to a specified order k, along with the function evaluation, with a fixed amount of effort. The Taylor coefficients of order k for sums and product of functions, as well as scalar products with reals, can be computed from those of summands and factors; therefore, the set of equivalence classes of functions can be endowed with well-defined operations, leading to the so-called truncated power series algebra [11, 12]. Similarly to the algorithms for floating-point arithmetic, the algorithms for functions follow, including methods to perform composition of functions, to invert them, to solve nonlinear systems explicitly, and to treat common elementary functions [10, 13]. In addition to these algebraic operations, the DA framework is endowed with differentiation and integration operators, therefore finalizing the definition of the DA structure. The differential algebra sketched in this section is implemented in the software COSY-Infinity [14].

For the sake of a more comprehensive illustration of the DA basics, the next section introduces the simplest nontrivial differential algebra for the first-order expansion of univariate functions. The reader can refer to [10] for its extension to the arbitrary order expansion of multivariate functions.

8.3.1.1 The Minimal Differential Algebra

Consider all ordered pairs (q_0, q_1), with q_0 and q_1 real numbers. Define addition, scalar multiplication, and vector multiplication as follows:

$$\begin{aligned}(q_0, q_1) + (r_0, r_1) &= (q_0 + r_0, q_1 + r_1),\\ t \cdot (q_0, q_1) &= (t \cdot q_0, t \cdot q_1),\\ (q_0, q_1) \cdot (r_0, r_1) &= (q_0 \cdot r_0, q_0 \cdot r_1 + q_1 \cdot r_0).\end{aligned} \tag{8.1}$$

The ordered pairs with the above arithmetic are called ${}_1D_1$. The multiplication of vectors is seen to have (1, 0) as the unity element. The multiplication is commutative, associative, and distributive with respect to addition. Together, the three operations defined in Eq. (8.1) form an algebra. Furthermore, they form an extension of real numbers, as $(r, 0) + (s, 0) = (r + s, 0)$ and (r, 0) ·(s, 0) = (r ·s, 0), so that the reals are included.

The multiplicative inverse of the pair (q_0, q_1) in ${}_1D_1$ is

$$(q_0, q_1)^{-1} = \left(\frac{1}{-q_0}, -\frac{q_1}{q_0^2}\right), \tag{8.2}$$

which is defined for any $q_0 \neq 0$.

One important property of this algebra is that it has an order compatible with its algebraic operations. Given two elements (q_0, q_1) and (r_0, r_1) in ${}_1D_1$, it is defined

$$\begin{aligned}(q_0, q_1) < (r_0, r_1) \quad &\textit{if} \quad q_0 < r_0 \quad \textit{or} \quad (q_0 = r_0 \text{ and } q_1 < r_1),\\ (q_0, q_1) > (r_0, r_1) \quad &\textit{if} \quad (r_0, r_1) \ < \ (q_0, q_1),\\ (q_0, q_1) = (r_0, r_1) \quad &\textit{if} \quad q_0 = r_0 \ \text{ and } q_1 = r_1.\end{aligned} \tag{8.3}$$

As for any two elements (q_0, q_1) and (r_0, r_1) only one of the three relation holds, ${}_1D_1$ is said totally ordered. The order is compatible with the addition and multiplication; for all (q_0, q_1), (r_0, r_1), (s_0, s_1) $\in$ ${}_1D_1$, it follows (q_0, q_1) < (r_0, r_1) $\Rightarrow$ $(q_0, q_1) + (s_0, s_1) < (r_0, r_1) + (s_0, s_1)$, and ($s_0$, s_1) > (0, 0) = 0 $\Rightarrow$ (q_0, q_1) ·(s_0, s_1) < (r_0, r_1) ·(s_0, s_1).

The number d = (0, 1) has the interesting property of being positive but smaller than any positive real number; indeed (0, 0) < (0, 1) < (r, 0) = r. For this reason d is called an infinitesimal or a differential. In fact, d is so small that its square vanishes. Since for any (q_0, q_1) $\in {}_1D_1$

$$(q_0, q_1) = (q_0, 0) + (0, q_1) = q_0 + d \cdot q_1, \tag{8.4}$$

the first component is called the real part and the second component the differential part.

The algebra in ${}_1D_1$ becomes a differential algebra by introducing a map ∂ from ${}_1D_1$ to itself, and proving that the map is a derivation. Define $\partial : {}_1D_1 \rightarrow {}_1D_1$ by

$$\partial(q_0, q_1) = (0, q_1). \tag{8.5}$$

Note that

$$\begin{aligned}\partial\{(q_0, q_1) + (r_0, r_1)\} &= \partial(q_0 + r_0, q_1 + r_1) = (0, q_1 + r_1)\\ &= (0, q_1) + (0, r_1) = \partial(q_0, q_1) + \partial(r_0, r_1)\end{aligned} \tag{8.6}$$

and

$$\begin{aligned}\partial\{(q_0, q_1) \cdot (r_0, r_1)\} &= \partial(q_0 \cdot r_0, q_0 \cdot r_1 + r_0 \cdot q_1) = (0, q_0 \cdot r_1 + r_0 \cdot q_1)\\ &= (0, q_1) \cdot (r_0, r_1) + (0, r_1) \cdot (q_0, q_1)\\ &= \partial\{(q_0, q_1)\} \cdot (r_0, r_1) + (q_0, q_1) \cdot \partial\{(r_0, r_1).\}\end{aligned} \tag{8.7}$$

This holds for all (q_0, q_1), $(r_0, r_1) \in {}_1D_1$. Therefore, ∂ is a derivation and $({}_1D_1, \partial)$ is a differential algebra.

The most important aspect of ${}_1D_1$ is that it allows the automatic computation of derivatives. Assume to have two functions f and g and to put their values and their derivatives at the origin in the form $(f(0), f'(0))$ and $(g(0), g'(0))$ as two vectors in ${}_1D_1$. If the derivative of the product $f \cdot g$ is of interest, it has just to be looked at the second component of the product $(f(0), f'(0)) \cdot (g(0), g'(0))$, whereas the first component gives the value of the product of the functions. Therefore, if two vectors contain the values and the derivatives of two functions, their product contains the values and the derivatives of the product function. Defining the operator [ ] from the space of differential functions to ${}_1D_1$ via

$$[f] = (f(0), f\prime(0)), \tag{8.8}$$

it holds

$$\begin{aligned}[f + g] &= [f] + [g],\\ [f \cdot g] &= [f] \cdot [g]\end{aligned} \tag{8.9}$$

and

$$[1/g] = [1]/[g] = 1/[g] \tag{8.10}$$

by using (8.2). This observation can be used to compute derivatives of many kinds of functions algebraically by merely applying arithmetic rules on ${}_1D_1$, beginning from the value and the derivative of the identity function $[x] = (x, 1) = x + \delta x$. Consider the example

$$f(x) = \frac{1}{x + (1/x)} \tag{8.11}$$

and its derivative

$$f'(x) = \frac{(1/x^2) - 1}{(x + (1/x))^2}. \tag{8.12}$$

The function value and its derivative at the point $x = 3$ are

$$f(3) = \frac{3}{10}, \quad f'(3) = -\frac{2}{25}. \tag{8.13}$$

Evaluating the function (8.11) in the DA framework at $(3, 1) = 3 + \delta x$ yields

$$\begin{aligned} f((3,1)) &= \frac{1}{(3,1) + 1/(3,1)} = \frac{1}{(3,1) + (1/3, -1/9)} \\ &= \frac{1}{(10/3, 8/9)} = \left(\frac{3}{10}, -\frac{8}{9} \bigg/ \frac{100}{9}\right) = \left(\frac{3}{10}, -\frac{2}{25}\right). \end{aligned} \tag{8.14}$$

Thus, the real part of the result is the value of the function at $x = 3$, whereas the differential part is the value of the derivative of the function at $x = 3$. This is simply justified by applying the relations (8.9) and (8.10).

$$\begin{aligned} [f(x)] &= \left[\frac{1}{x + 1/x}\right] = \frac{1}{[x + 1/x]} \\ &= \frac{1}{[x] + [1/x]} = \frac{1}{[x] + 1/[x]} \\ &= f([x]). \end{aligned} \tag{8.15}$$

The method can be generalized to treat common intrinsic functions.

8.3.2 Representation of Objective and Constraint Functions

The evaluation of the objective and constraint functions in MGA transfers involves solving implicit equations, which become parametric when their Taylor expansion in the design variables is of interest. Three implicit equations have to be solved in the model adopted. Two of them already appear in simple planet-to-planet transfers. These are illustrated with a practical example in the following. Let us consider the transfer from planet P_1 to planet P_2 sketched in Fig. 8.4.

The objective function for this problem is the overall ΔV; this can be evaluated by using two design variables. A common choice is selecting the departure epoch

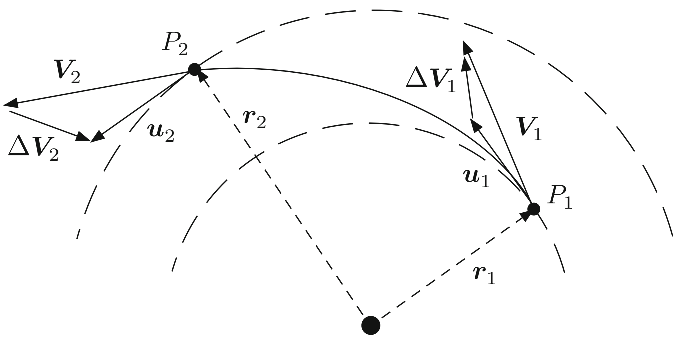


Fig. 8.4 A two-impulse planet-to-planet transfer

from P_1, T_1, and the time of flight, t_{12}. The arrival epoch at P_2 is $T_2 = T_1 + t_{12}$, and the position and velocity of P_1 and P_2 at both ends of the transfer ($\mathbf{r}_1$, $\mathbf{v}_1$ and $\mathbf{r}_2$, $\mathbf{v}_2$, respectively) are obtained through their planetary ephemerides. Given $\mathbf{r}_1$, $\mathbf{r}_2$, and t_{12}, the corresponding Lambert's problem is solved to compute the heliocentric initial and final velocities, $\mathbf{V}_1$ and $\mathbf{V}_2$, respectively. The two velocity impulses required to accomplish the transfer are $\Delta V_i = \mathbf{V}_i - \mathbf{v}_i$, $i = 1, 2$.

Problem Statement. Let $\mathbf{x} = \{T_1, t_{12}\}$, optimal two-impulse transfers from P_1 to P_2 are found by solving

$$\begin{aligned} \min_{\mathbf{x}} \Delta V(\mathbf{x}) \quad \textit{subject to} \quad & \Delta V_1(\mathbf{x}) \le \Delta V_1^{max} \\ & \Delta V_2(\mathbf{x}) \le \Delta V_2^{max}, \end{aligned} \tag{8.16}$$

where $\Delta V = \Delta V_1 + \Delta V_2 = \|\Delta V_1\| + \|\Delta V_2\|$ and $\mathit{\Delta} V_1{}^{\max}$ and $\mathit{\Delta} V_2{}^{\max}$ are maximum allowed values for $\mathit{\Delta} V_1$ and $\mathit{\Delta} V_2$, respectively.

The evaluation of planetary ephemerides is required to compute $\mathbf{r}_i$ and $\mathbf{v}_i$, for $i = 1, 2$. An analytical ephemeris model is used, which is based on interpolating the planetary orbital elements delivered by JPL's Horizons system [15] with cubic splines. The analytical model supplies the eccentricity of the planet orbit, e, and the mean anomaly of the planet, M, at the evaluation epoch. Then, the Kepler equation

$$f(E) = E - e \sin E - M = 0 \, , \tag{8.17}$$

must be solved for the eccentric anomaly, E, which is necessary to evaluate the planet position and velocity.

The second implicit equation appears in the solution of the Lambert problem for $\mathbf{V}_1$ and $\mathbf{V}_2$. In Lambert's problem, the initial position, final position, and the time of flight between the two positions are given. Solving Lambert's problem defines the Keplerian orbit that connects the two position vectors in the given time, allowing the calculation of the velocities at the initial and final positions. Lambert's theorem states that the time of flight $\Delta t = t_2 - t_1$ depends only on the semi-major axis a, the sum of the two radii $r_1 + r_2$, and the distance between the initial and final positions, i.e., the chord length $c = ||r_2 - r_1||$ [16]. The time required for the transfer can be written as

$$\Delta t = \sqrt{\frac{a^3}{\mu}}(2k\pi + (E_2 - e\sin E_2) - (E_1 - e\sin E_1)), \tag{8.18}$$

where E_1 and E_2 are the eccentric anomalies of the initial and final positions respectively, measured on the connecting arc. The problem, now is to find the correct values of a, E_1, E_2, and e that give the desired time of flight. As Lambert stated, however, the transfer time depends only on the three quantities mentioned earlier. The two radii and the chord length are already known from the problem definition. The semi-major axis is the only unknown parameter. Thus, it is possible to write the transfer time as a function of the semi-major axis only, or some other parameters such as p or ΔE. In our approach, based on Battin's algorithm [16], the nonlinear equation to be solved is

$$A(x) - \Delta t = 0, \tag{8.19}$$

in which

$$A(x) = g(x)^{3/2}(\alpha(x) - \sin\alpha(x) - \beta(x) + \sin\beta(x)). \tag{8.20}$$

The functions $\alpha(x)$ and $\beta(x)$ are related to x via the relations

$$\sin^2\frac{1}{2}\alpha(x) = \frac{s}{2g(x)} \qquad \sin^2\frac{1}{2}\beta(x) = \frac{s-c}{2g(x)}, \tag{8.21}$$

with

$$g(x) = \frac{s}{2(1-x^2)}, \tag{8.22}$$

and the semi-perimeter

$$s = (r_1 + r_2 + c)/2. \tag{8.23}$$

Note that the relation between a and x is simply given by

$$a = \frac{s}{2(1-x^2)}. \tag{8.24}$$

Once Eq. (8.19) is solved, the initial and final heliocentric velocities of the spacecraft are computed via algebraic and transcendental functions.

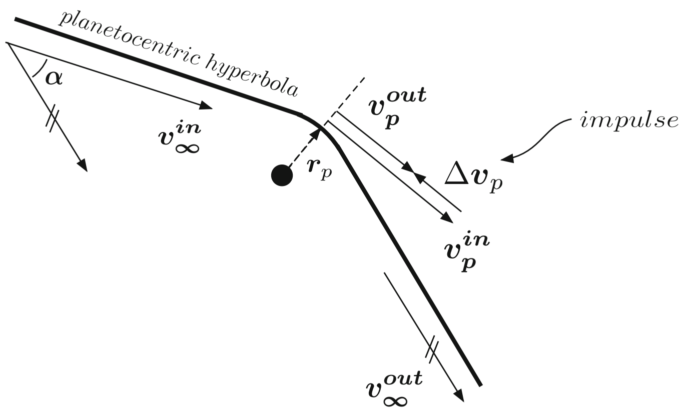


Fig. 8.5 Powered gravity assist

The third implicit equation occurs when transfers with powered gravity assists are considered. In a powered gravity assist, the spacecraft provides a tangential impulse at the pericenter of the incoming hyperbola. Therefore, the planeto-centric trajectory is made up of two arcs of hyperbola patched together (see Fig. 8.5). The angle α, usually referred to as bending angle, between the incoming and the outgoing asymptotic velocities, $v_\infty{}^{\text{in}}$ and $v_\infty{}^{\text{out}}$, respectively, is related to the pericenter radius via [8]

$$f(r_p) = \arcsin\frac{a^-}{a^- + r_p} + \arcsin\frac{a^+}{a^+ + r_p} - \alpha = 0\ , \tag{8.25}$$

where $a^- = 1/(\mathbf{v}_\infty^{in} \cdot \mathbf{v}_\infty^{in})$ and $a^+ = 1/(\mathbf{v}_\infty^{out} \cdot \mathbf{v}_\infty^{out})$. The angle α can be easily computed from the two heliocentric arcs connected at the gravity assist. The solution of the implicit equation (8.25) delivers the pericenter radius of the planetocentric trajectory. The planetocentric velocities $\boldsymbol{v}_p{}^{\text{in}}$ and $\boldsymbol{v}_p{}^{\text{out}}$ at the pericenter, corresponding to the incoming and outgoing hyperbolic arcs, respectively, are computed using $\boldsymbol{r}_p$, $\boldsymbol{v}_\infty{}^{\text{in}}$, and $\boldsymbol{v}_\infty{}^{\text{out}}$. Thus, the magnitude of the impulsive maneuver at the pericenter is simply $\Delta v_p = \|\mathbf{v}_p^{out} - \mathbf{v}_p^{in}\|$.

A classical numerical method for the solution of implicit equations can be used to solve Eqs. (8.17)–(8.25) for a point-wise evaluation of the objective and constraint functions. This is not true when the Taylor expansion of the objective and constraint functions is of interest, as the implicit equations become parametric in the design variables. This is briefly illustrated for Eq. (8.17) in the following. Similar arguments hold for Eqs. (8.19) and (8.25).

Let us consider the evaluation of planetary ephemerides. In the DA framework, we are interested in the Taylor expansion of planet's position and velocity with respect to the evaluation epoch. Thus, Kepler's equation (8.17) is not solved for real values of the eccentric anomaly, but rather for its Taylor expansion with respect to the epoch. More specifically, the epoch is initialized as a DA variable, $[T] = T^0$

$+\delta T$, where δT is the displacement of the epoch from the reference value T^0. Then, the simple evaluation of the analytical ephemeris model in the DA framework delivers the Taylor expansion of the eccentricity e and the mean anomaly M with respect to the epoch,

$$\begin{aligned} [e] &= \mathscr{M}_e(\delta T), \\ [M] &= \mathscr{M}_M(\delta T), \end{aligned} \tag{8.26}$$

where $\mathscr{M}_e$ and $\mathscr{M}_M$ denote the resulting Taylor polynomials for e and M. Thus, the explicit dependence of e and M on δT appears in Kepler's equation, which now reads

$$f(E, \delta T) = E - [e] \sin E - [M] = E - \mathscr{M}_e(\delta T) \sin E - \mathscr{M}_M(\delta T) = 0. \tag{8.27}$$

The parametric implicit equation (8.27) must be solved for the Taylor expansion of E with respect to the parameter δT, $[E] = \mathscr{M}_E(\delta T)$. Dedicated techniques have been developed in past works to address the previous task [9]. Once $\mathscr{M}_E(\delta T)$ is available, the Taylor expansions of the planet position and velocity are readily obtained by carrying out the remaining algebra in the DA framework.

8.3.3 Implementation of GASP-DA

The use of differential algebra is now introduced in GASP, with the primary goal of expanding the objective function with respect to the optimization variables in subdomains of the original search space. The resulting algorithm, referred to as GASP–DA, is summarized in the following for the P_1–P_2 transfer problem (8.16):

1. Subdivide the search space $\mathbf{x} = \{ T_1, t_{12}\}$ into boxes and put them in a list $\mathscr{L}$.
2. While $\mathscr{L} \neq \phi$,
   - i. Take out a box $\mathbf{X}$ from $\mathscr{L}$.
   - ii. Initialize T_1 and t_{12} as DA variables and compute the Taylor expansion of ΔV_1 on $\mathbf{X}$.
   - iii. Bound the polynomial expansion of ΔV_1 on $\mathbf{X}$, i.e., estimate its minimum $\underline{\Delta V_1}$ and maximum $\overline{\Delta V_1}$ on $\mathbf{X}$.
   - iv. If $\underline{\Delta V_1} > \Delta V_1^{max} \Rightarrow$ discard the current box $\mathbf{X}$ and go to step *i*.
   - v. Compute the Taylor expansion of ΔV_2 on $\mathbf{X}$.
   - vi. Bound the polynomial expansion of ΔV_2 on $\mathbf{X}$, i.e., estimate its minimum $\underline{\Delta V_2}$ and maximum $\overline{\Delta V_2}$ on $\mathbf{X}$.
   - vii. If $\underline{\Delta V_2} > \Delta V_2^{max} \Rightarrow$ discard the current box $\mathbf{X}$ and go to step *i*.
   - viii. Put $\mathbf{X}$ in a list of feasible boxes $\mathscr{X}$.

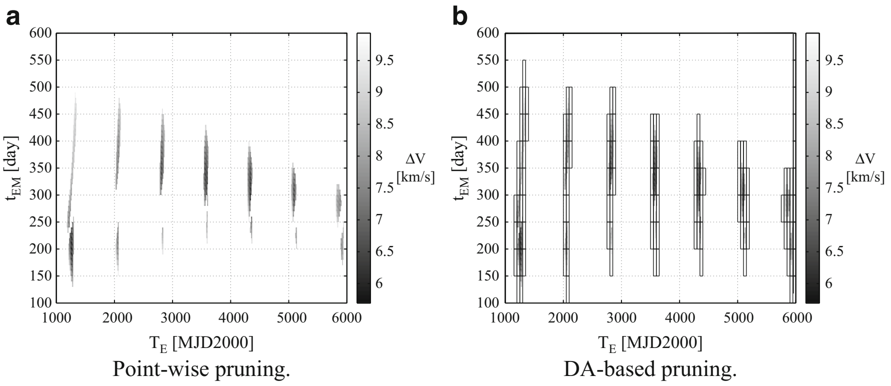


Fig. 8.6 Search space pruning for Earth–Mars transfers

It is worth mentioning that bounding the Taylor expansions, as required in steps 2.iii and 2.vi of the previous algorithm, is not a trivial task. This is done with a non-validated quadratic bounder [17]. The bounder makes use of the quadratic part of the Taylor expansion to get estimates of the minimum of a function over each box.

To assess the performances of GASP-DA, its application to an Earth–Mars transfer is analyzed. A search space of 5,000 days on the departure epoch ($T_1 \in$ [1000, 6000] MJD2000) and 500 days on the transfer time ($t_{12} \in$ [100, 600]) is selected. Figure 8.6a is obtained with classical pointwise techniques, and reports the search space remaining after imposing the two constraints:

$$\begin{aligned} \Delta V_1 &\leq 5\ km/s, \\ \Delta V_2 &\leq 5\ km/s. \end{aligned} \tag{8.28}$$

In the DA implementation of problem (8.16), the search space is uniformly subdivided in boxes of size 50 days on each variable, and the pruning is then performed using the constraints (8.28). The boxes remaining after pruning (Fig. 8.6b) sharply enclose the feasible space in Fig. 8.6a. A comprehensive assessment of the performances of GASP-DA can be found in [9].

8.4 Introduction of Deep Space Maneuvers in GASP-DA

The GASP-DA algorithm is extended in this section to manage DSM. These maneuvers are usually carried out to improve the performances of the transfer trajectories in terms of total cost. From the trajectory optimization standpoint, the introduction of DSM increases the chances of reducing the overall transfer cost associated to pure MGA transfers. On the other hand, each DSM involves additional degrees of freedom that widen the search space and affect convergence to the global minimum.

The mathematical formulation of this new problem is not unique, and the performances of the optimization process strongly depend upon problem transcription, especially in the DA frame. Different formulations have been investigated by the authors in [17]. After some preliminary considerations on the introduction of DSM in MGA transfers, this section describes the strategy that better fits the DA implementation of GASP. The performances of the resulting GASP–DSM–DA algorithm are then assessed on practical cases.

8.4.1 Preliminary Considerations

The solution of several Lambert's problems is yet at the basis of the objective function evaluation in an MGA-DSM problem. However, unlike MGA problems, Lambert's arcs connect either two consecutive planets, or a planet to a maneuver point (and vice versa). The location of the maneuver points has to be specified by adding new variables to the decision vector. It can be easily shown that, for each DSM introduced, a minimum set of four variables must be added for a three-dimensional transfer problem (three variables in the planar case). Based on rationales in [9, 17] the search space pruning of MGA-DSM transfers is carried out in a planar model. The optimal, spatial trajectory is then caught by the subsequent optimization in the three-dimensional environment. This implies that, letting n_P and n_D be the number of planets and maneuvers, respectively, the decision vector for search space pruning includes $n_\mathrm{P} + 3n_\mathrm{D}$ variables.

Similarly to the MGA case, the pruning process of MGA-DSM problems consists in (1) expanding the objective function and constraints in Taylor series of the decision variables over subsets of the search space, (2) bounding the resulting polynomials and (3) pruning away unfeasible boxes from the search space. Unlike a point-wise approach, the performance of the whole procedure depends on the availability of accurate range bounds of the constraint functions over each box. Thus, working with smooth functions of the least number of variables is desirable to efficiently prune the search space. Different strategies for the introduction of DSM show different dependencies on the decision variables, which is the key aspect in a DA framework.

An additional consideration concerns the increased computational burden when moving from the MGA to the MGA-DSM problem. This pertains not only the increased dimension of the search space (from n_P to $n_\mathrm{P} + 3n_\mathrm{D}$), but rather it is an intrinsic consequence of representing a function with its Taylor expansion. The number of monomials needed to represent a function of v variables up to the order n is $NM = (k + v)!/(k!v!)$. Thus, at fixed k, the number of monomials for a MGA-DSM problem increases with factorial law with respect to a simple MGA problem, together with the number of required operations.

Based on the previous observations, the complexity of MGA problems increases when DSM are introduced. Nevertheless, the associated issues can be prevented and limited by carefully selecting the strategy for DSM introduction. It is anyway

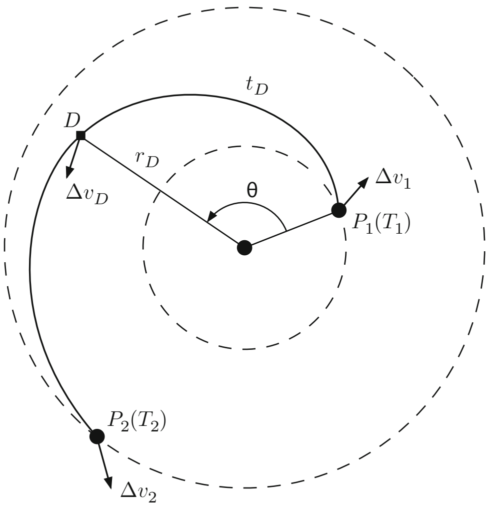


Fig. 8.7 Planet-to-planet transfer with one DSM

important to preserve the idea of the GASP algorithm: subdivide the problem into a cascade of subproblems and exploit the cut-off values to prune away unfeasible zones.

8.4.2 Formulation of GASP–DSM–DA

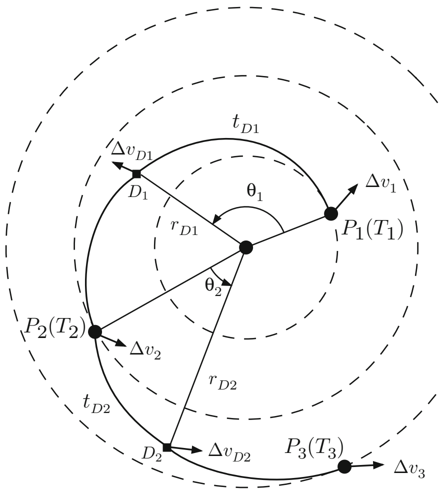
The strategy to introduce DSM into GASP-DA is addressed in the following. The main idea is to identify the problem formulation that most verges the objective function evaluation to the solution of multiple Lambert's problems by breaking the whole transfer trajectory into subsequent Lambert's arcs. The strategy is first illustrated on a simple planet-to-planet transfer. Then, the extension to general MGA transfers is addressed.

8.4.2.1 Planet-to-Planet Case

Let us consider a planar planet-to-planet transfer as defined in Sect. 8.3.2, and let us introduce one intermediate maneuver (D). With reference to Fig. 8.7, three variables are added to the design vector:

- r_D—*maneuver radius* is the distance between D and the Sun
- t_D—*partial tof* is the time of flight associated to the P_1–D arc
- θ—*incremental anomaly* is the anomaly of D relative to P_1

Fig. 8.8 MGA transfer with two DSM



Clearly, $t_D \leq T_2 - T_1$. The ephemeris model gives the position of P_1 at T_1, $\mathbf{r}_1$, and P_2 at T_2, $\mathbf{r}_2$. Thus, the position of D, $\mathbf{r}_D$, is uniquely determined by the angle θ and r_D. Within this strategy, the following dependency holds

$$\mathbf{r}_D = \mathbf{r}_D(T_1, r_D, \theta).$$

The overall transfer can be characterized by solving two Lambert's problems: one from $\mathbf{r}_1$ to $\mathbf{r}_D$ with time of flight t_D and one from $\mathbf{r}_D$ to $\mathbf{r}_2$ with time of flight $T_2 - T_1 - t_D$. Thus, the decision vector is $\mathbf{x} = [T_1, T_2, r_D, \theta, t_D]$.

The search space pruning problem consists now in finding $\mathscr{X}$ such that

$$\exists\ \mathbf{x}^* \in \mathscr{X} \mid \ \Delta v_1(\mathbf{x}^*) \leq \Delta v_1^{max},\ \ \Delta v_D(\mathbf{x}^*) \leq \Delta v_D^{max},\ \ \Delta v_2(\mathbf{x}^*) \leq \Delta v_2^{max}, \tag{8.29}$$

where Δv_D is the cost of the DSM and $\Delta v_D{}^{\max}$ is its maximum allowed value. The dependencies of the three functions in Eq. (8.29) are

$$\begin{aligned}
\Delta v_1 &= \Delta v_1(T_1, r_D, \theta, t_D), \\
\Delta v_D &= \Delta v_D(T_1, T_2, r_D, \theta, t_D), \\
\Delta v_2 &= \Delta v_2(T_1, T_2, r_D, \theta, t_D).
\end{aligned} \tag{8.30}$$

Thus, all constraint functions depend on five variables at most.

8.4.2.2 MGA Case

The MGA transfer case is now addressed. Referring to Fig. 8.8, we first consider a MGA case with three planets (P_1, P_2, P_3) and two DSM (D_1 and D_2). The search space is defined by the decision vector

$$\mathbf{x} = [T_1, T_2, T_3, r_{D_1}, \theta_1, t_{D_1}, r_{D_2}, \theta_2, t_{D_2}],$$

where the last three variables are introduced to identify the position of the second DSM, $\mathbf{r}_{D_2}$, and the transfer time between planet P_2 and D_2, t_{D_2}. The inequality $t_{D_1} \leq T_2 - T_1$ still holds, whereas t_{D_2} is subject to $t_{D_2} \leq T_3 - T_2$. Thus, the overall transfer can be characterized by solving four Lambert's problems. The pruning problem consists in finding $\mathscr{X}$ such that $\exists\ \mathbf{x}^* \in \mathscr{X}$ that yields

$$\begin{aligned} \Delta v_1(\mathbf{x}^*) \leq \Delta v_1^{max}, \Delta v_{D_1}(\mathbf{x}^*) \leq \Delta v_{D_1}^{max}, \Delta v_2(\mathbf{x}^*) \leq \Delta v_2^{max}, \\ r_p(\mathbf{x}^*) \geq r_p^{\min}, \Delta v_{D_2}(\mathbf{x}^*) \leq \Delta v_{D_2}^{max}, \Delta v_3(\mathbf{x}^*) \leq \Delta v_3^{max}, \end{aligned} \tag{8.31}$$

where ${r_p}^{\min}$ is the minimum allowed pericenter radius for the gravity assist at P_2.

Analogously to the previous case, we are interested in assessing the dependence of the position of D_1 and D_2, $\mathbf{r}_{D_1}$ and $\mathbf{r}_{D_2}$, on the problem variables. The main advantage of the approach proposed is that $\mathbf{r}_{D_2}$ is identified on the basis of the position of P_2, and therefore

$$\begin{aligned} \mathbf{r}_{D_1} &= \mathbf{r}_{D_1}(T_1, r_{D_1}, \theta_1), \\ \mathbf{r}_{D_2} &= \mathbf{r}_{D_2}(T_2, r_{D_2}, \theta_2). \end{aligned}$$

Consequently, after the solution of the Lambert problems, the constraint functions show the dependencies

$$\begin{aligned} \Delta v_1 &= \Delta v_1(T_1, r_{D_1}, \theta_1, t_{D_1}), \\ \Delta v_{D_1} &= \Delta v_{D_1}(T_1, T_2, r_{D_1}, \theta_1, t_{D_1}), \\ \Delta v_2 &= \Delta v_2(T_1, T_2, r_{D_1}, \theta_1, t_{D_1}, r_{D_2}, \theta_2, t_{D_2}), \\ r_p &= r_p(T_1, T_2, r_{D_1}, \theta_1, t_{D_1}, r_{D_2}, \theta_2, t_{D_2}), \\ \Delta v_{D_2} &= \Delta v_{D_2}(T_2, T_3, r_{D_2}, \theta_2, t_{D_2}), \\ \Delta v_3 &= \Delta v_3(T_2, T_3, r_{D_2}, \theta_2, t_{D_2}). \end{aligned} \tag{8.32}$$

As can be seen, the critical functions in terms of dependencies are Δv_2 and r_p, which depend on eight variables. These are the constraint functions associated to the gravity assist at P_2, which is located between D_1 and D_2.

From simple reasoning, this result can be extended to a general MGA transfer problem with at most one DSM within each planet-to-planet arc. More specifically, let us consider a MGA transfer with n planets $P_1, \ldots, P_i, \ldots, P_n$ and $n - 1$ maneuvers $D_1, \ldots, D_i, \ldots, D_{n-1}$, where D_i is performed between P_i and P_{i+1}. The dependency of constraint functions is

$$\begin{aligned} \Delta v_1 &= \Delta v_1(T_1, r_{D_1}, \theta_1, t_{D_1}), \\ \Delta v_{D_{i-1}} &= \Delta v_{D_{i-1}}(T_{i-1}, T_i, r_{D_{i-1}}, \theta_{i-1}, t_{D_{i-1}}), \\ \Delta v_i &= \Delta v_i(T_{i-1}, T_i, r_{D_{i-1}}, \theta_{i-1}, t_{D_{i-1}}, r_{D_i}, \theta_i, t_{D_i}), \\ r_{p_i} &= r_{p_i}(T_{i-1}, T_i, r_{D_{i-1}}, \theta_{i-1}, t_{D_{i-1}}, r_{D_i}, \theta_i, t_{D_i}), \\ \Delta v_n &= \Delta v_n(T_{n-1}, T_n, r_{D_{n-1}}, \theta_{n-1}, t_{D_{n-1}}). \end{aligned} \tag{8.33}$$

for $i = 2, \ldots, n-1$, where r_{p_i} is the pericenter radius of the gravity assist at P_i. As can be seen, the proposed strategy limits the maximum dependency to eight variables, regardless of the number of planets and maneuvers.

8.5 Test Cases

A number of application cases are dealt with in this section to assess the performances of the algorithm. More specifically, the classic Earth–Mars transfer, with an intermediate DSM, is first discussed (Sect. 8.5.2). After this simple case, four transfer options for a mission to Jupiter are taken into account (Sects. 8.5.3–8.5.6). These cases differ in the transfer strategy, although they all include only one DSM. The last two cases are devoted to Cassini-like transfers with one and two DSM (Sects. 8.5.7 and 8.5.8, respectively). For each case, the global optimum achieved is compared with the results of GASP-DA, where pure MGA transfers are treated [9].

The outcome of the pruning process is a list of boxes that enclose feasible regions of the search space. A local optimization is then carried out within the remaining boxes to locate the minimum of the objective function, which is the purpose of the original optimization problem. This choice speeds-up the local optimization as the optimizer runs over small domains. Thus, the whole pruning and optimization sequence is implemented in a deterministic way, and the repeatability of the results is preserved. The computational time is relative to a PC with 2.01 Ghz CPU and 512 Mb RAM.

8.5.1 Search Space Definition

The search space bounds and the size of the boxes in which it is subdivided are chosen depending on the problem to solve. This is valid for both the n_{P} epochs and the $3n_{\mathrm{D}}$ auxiliary variables that identify the DSM. The bounds and box-size for the n_{P} epochs are given in dedicated tables. Specific arguments must be provided for the selection of bounds and box-sizes for the $3n_{\mathrm{D}}$ auxiliary variables. These values, reported in Table 8.1, have been selected heuristically, based on the results of an extensive test campaign. They represent a good trade-off between the accuracy of the Taylor representation on the resulting boxes and a limited computational time. The table shows the lower and upper bounds for these variables. These bounds are relative to a maneuver located in the transfer arc between P_i and P_{i+1} (Fig. 8.7). The bounds for θ are trivial. The terms r_i and r_{i+1}, $r_{i+1} > r_i$, stand for the mean radii of P_i and P_{i+1} orbits, respectively.

Table 8.1 Bounds and box-sizes for the $3n_D$ auxiliary variables

Variable	Lower bound	Upper bound	Box-size
r_D	$0.9r_i$	$1.1r_{i+1}$	0.1 AU
θ	0	2π	10 deg
t_D	0	$T_{i+1} - T_i$	50 day

Table 8.2 Search space bounds, box-sizes, and optimal solution found for the EdM transfer

	T_E	t_{EM}
	MJD 2000	days
L_b	1,000	200
U_b	2,000	650
Δ	50	50
Sol.	1,243.2	606.2

Thus, the maneuver is constrained to lie into an annular region enclosing the planets orbits (If $r_i > r_{i+1}$, then $r_D \in [r_{i+1}, r_i]$). When the maneuver is located between two encounters of the same planet, i.e., $P_i = P_{i+1}$, we set $r_D \in [0.9r_i, 1.1a_{1:2}]$, where $a_{1:2}$ is the semimajor axis of an orbit in 1:2 resonance with the orbit of P_i. We let the partial time of flight, t_D, to vary within $[0, T_{i+1} - T_i]$, where T_i and T_{i+1} are the epochs at the P_i and P_{i+1} encounter, respectively.

8.5.2 EdM

The first test case is an Earth–Mars transfer with one DSM (indicated with d in the planets sequence). The search space is defined in Table 8.2 in terms of bounds on the departure epoch T_E and the transfer time t_{EM}. However, as stated in the previous sections, the pruning is carried out on the search space defined by the epochs of each planet encounter, i.e., T_E and T_M in this case. This observation holds for all test cases. The search space definition is completed by the bounds for the three additional variables in Table 8.1. The last two rows of Table 8.2 report the box-size along each epoch and the optimal solution found, respectively. The GASP–DSM–DA algorithm solves this problem in 253.2 s. Wesummarize below some features of the problem settings and the resultsachieved.

- Problem constraints: $\Delta v_E \leq 3$ km/s, $\Delta v_d \leq 3$ km/s, $\Delta v_M \leq 3$ km/s, $\Delta v_{tot} \leq 7\{\text{km/s}\}$
- Total number of boxes = 388,800
- Boxes remaining after pruning = 1,603 (0.41%)
- Optimal objective function value = 5.632 km/s
- Optimal objective function value without DSM (GASP-DA result) = 5.667 km/s

Table 8.3 Search space bounds, box-sizes, and optimal solution found for the EMdJ transfer

	T_E	t_{EM}	t_{MJ}
	MJD 2000	days	days
L_b	1,000	300	1,000
U_b	3,000	700	2,000
Δ	50	50	100
Sol.	2,804.7	321.9	1,161.9

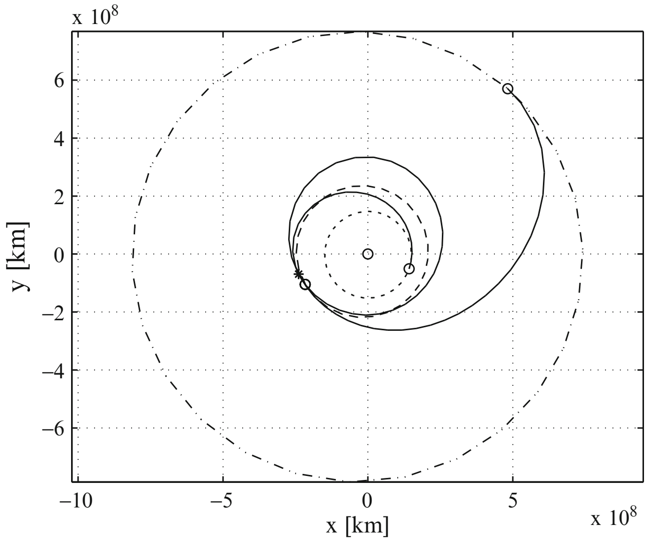


Fig. 8.9 Optimal EMdJ transfer

8.5.3 EMdJ

One planet is added to the transfer. In particular, an Earth–Mars–Jupiter transfer is investigated, with one DSM between Mars and Jupiter. Table 8.3 states the bounds and the box-sizes on the departure epoch and the transfer times, together with the optimal solution found. A purely ballistic Mars gravity assist is imposed by setting the cutoff value for Δv_M to zero in the powered gravity assist model. As in the previous problem, the introduction of a DSM improves the objective function found by GASP-DA. The CPU time is 451 s. The optimal transfer is shown in Fig. 8.9.

- Constraints: $\Delta v_E \leq 5$ km/s, $\Delta v_M \leq 0$ km/s, $\Delta v_d \leq 5$ km/s, $\Delta v_J \leq 5$ km/s, $\Delta v_{tot} \leq$ 15 km/s
- Total number of boxes = 8.52e7
- Boxes remaining after pruning = 323 (3.79e-4%)
- Optimal objective function value = 12.481 km/s
- Optimal objective function value without DSM (GASP-DA result) =13.416 km/s

Table 8.4 Search space bounds, box-sizes, and optimal solution found for the EMdMJ transfer

	T_E	t_{EM}	t_{MM}	t_{MJ}
	MJD 2000	days	days	days
L_b	3,650	30	330	600
U_b	7,300	430	830	2,000
Δ	50	100	100	200
Sol.	4,353.8	371.1	915.8	1,129.5

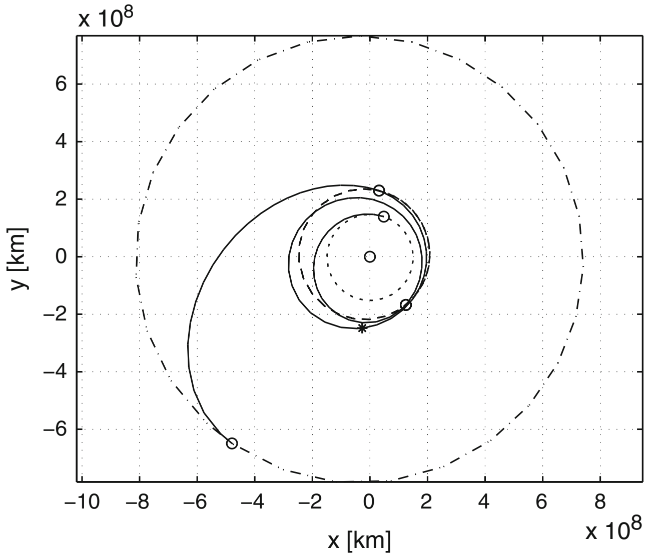


Fig. 8.10 Optimal EMdMJ transfer

8.5.4 EMdMJ

An alternative transfer strategy to Jupiter is investigated. The time domain for the EMdMJ problem is stated in Table 8.4. In this case the maneuver radius is search within $r_D \in [0.\ 9r_M,\ 1.\ 1a_{1:\ 2}]$, where r_M is the mean radius of Mars' orbit whereas $a_{1:\ 2}$ is the semimajor axis of a 1:2 resonant orbit with Mars' orbit. This problem is solved in 144.2 s. The result obtained by GASP-DA for the pure MGA transfer without DSM is once again improved. Figure 8.10 illustrates the resulting optimal transfer.

- Constraints: $\Delta v_E \leq 4$ km/s, $\Delta v_{M,1} \leq 0$ km/s, $\Delta v_d \leq 3$ km/s, $\Delta v_{M,2} \leq 0$ km/s, $\Delta v_J \leq 7$ km/s, $\Delta v_{tot} \leq 12$ km/s
- Total number of boxes = 9.19e7
- Boxes remaining after pruning = 717 (7.8e-3%)
- Optimal objective function value = 10.843 km/s
- Optimal objective function value without DSM (GASP-DA result) = 12.864 km/s

Table 8.5 Search space bounds, box-sizes, and optimal solution found for the EVdVEJ transfer

	T_E	t_{EV}	t_{VV}	t_{VE}	t_{EJ}
	MJD 2000	days	days	days	days
L_b	3,650	80	80	80	600
U_b	7,300	430	830	830	2,000
Δ	50	25	25	50	200
Sol.	3,859.5	119.2	429.6	564.8	1,244.3

Table 8.6 Search space bounds, box-sizes, and optimal solution found for the EVEdEJ transfer

	T_E	t_{EV}	t_{VE}	t_{EE}	t_{EJ}
	MJD 2000	days	days	days	days
L_b	3,650	80	80	80	600
U_b	7,300	430	830	830	2,000
Δ	50	25	50	50	200
Sol.	3,863.4	128.8	288.4	713.3	1,068.2

8.5.5 EVdVEJ

The MGA transfer EVdVEJ to Jupiter is now analyzed. One DSM maneuver is performed between the two consecutive Venus encounters. The five-dimensional domain for the departure epoch and the transfer times is defined in Table 8.5. The domain for DSM characterization is added based on Table 8.1. The resulting search space is relatively large, and 1.85e10 boxes are necessary to run GASP–DSM–DA. Thanks to constraint propagation, this problem is solved in 2,770 s.

- Constraints: $\Delta v_{E,dep} \leq 4.5$ km/s, $\Delta v_{V,1} \leq 0$ km/s, $\Delta v_d \leq 0.5$ km/s, $\Delta v_{V,2} \leq 0$ km/s, $\Delta v_E \leq 0$ km/s, $\Delta v_J \leq 7$ km/s, $\Delta v_{tot} \leq 12$ km/s
- Total number of boxes = 1.85e10
- Boxes remaining after pruning = 3.80e4 (2.06e − 4%)
- Optimal objective function value = 9.304 km/s
- Optimal objective function value without DSM (reference solution [18]) = 10.503 km/s

8.5.6 EVEdEJ

The last transfer strategy to Jupiter is now studied. An EVEdEJ transfer problem is solved using the search space bounds and the box-sizes reported in Table 8.6. Similarly to the previous problem, the search space is relatively large, and a systematic analysis based on a grid sampling would be impossible without an efficient constraint propagation. Thanks to the pruning strategy and the possibility of expanding the constraint functions over subdomains of the search space, GASP–DSM–DA solves this problem in 2,392 s. The main results are listed below, and a plot of the optimal transfer is reported in Fig. 8.11.

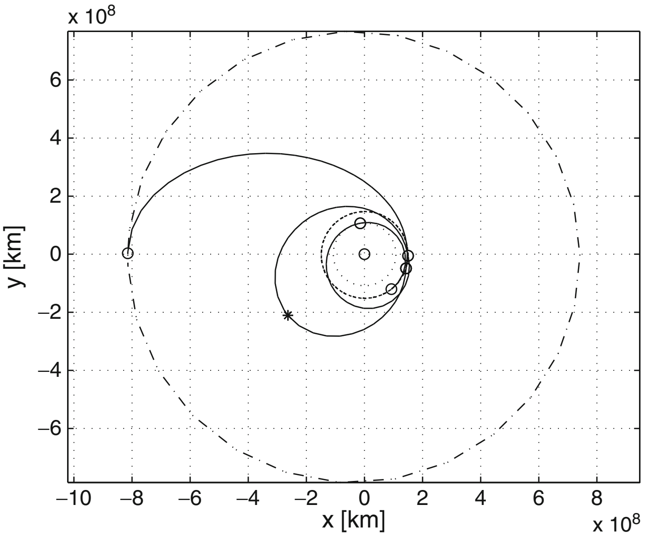


Fig. 8.11 Optimal EVEdEJ transfer

Table 8.7 Search space bounds, box-sizes, and optimal solution found for the EVdVEJS transfer

	T_{E}	t_{EV}	t_{VV}	t_{VE}	t_{EJ}	t_{JS}
	MJD 2000	days	days	days	days	days
L_b	− 1,000	80	200	30	400	800
U_b	0	430	500	180	1,600	2,200
$\varDelta$	50	25	25	50	200	200
Sol.	− 787.0	165.8	427.7	57.7	596.1	2,200

- Constraints: $\varDelta v_{\mathrm{E,dep}} \leq 4$ km/s, $\varDelta v_{\mathrm{V}} \leq 0$ km/s, $\varDelta v_{\mathrm{E,1}} \leq 0$ km/s, $\varDelta v_{d} \leq 3$ km/s, $\varDelta v_{\mathrm{E,2}} \leq 0$ km/s, $\varDelta v_{\mathrm{J}} \leq 7$ km/s, $\varDelta v_{\mathrm{tot}} \leq 12$ km/s
- Total number of boxes = 9.25e9
- Boxes remaining after pruning = 4.84e4 (5.23e − 4%)
- Optimal objective function value = 8.670 km/s
- Optimal objective function value without DSM (GASP-DA result) = 10.09 km/s; reference solution [18] = 8.680 km/s

8.5.7 EVdVEJS

This section is devoted to a Cassini-like transfer with a DSM between the two Venus gravity assists. Saturn is the target planet, which is reached after four gravity assists. Thus, the overall transfer involves six planet encounters and one DSM, leading to a nine-dimensional optimization problem. The search space and the box-sizes are stated in Table 8.7. The computational time required by the DA-based

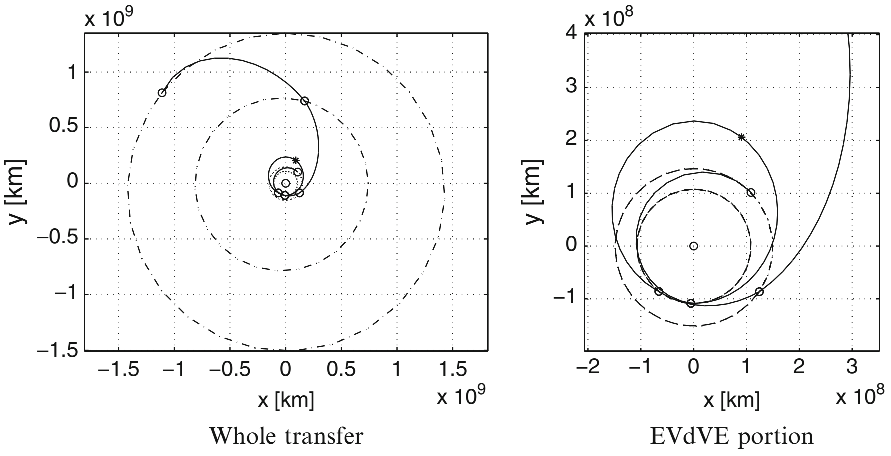


Fig. 8.12 Optimal EVdVEJS transfer

pruning and optimization algorithm is 210 s. Figure 8.12a illustrates the optimal transfer, whereas a detail on the EVdVE portion is reported in Fig. 8.12b.

- Constraints: $\Delta v_{\mathrm{E,dep}} \leq 4$ km/s, $\Delta v_{\mathrm{V},1} \leq 1$ km/s, $\Delta v_d \leq 1$ km/s, $\Delta v_{\mathrm{V},2} \leq 0$ km/s, $\Delta v_{\mathrm{E}} \leq 0$ km/s, $\Delta v_{\mathrm{J}} \leq 0$ km/s, $\Delta v_{\mathrm{S}} \leq 5$ km/s
- Total number of boxes = 3.92e8
- Boxes remaining after pruning = 2,281 (1e − 3%)
- Optimal objective function value = 8.299 km/s
- Optimal objective function value without DSM (GASP-DA result) = 8.619 km/s

8.5.8 EVdVEJdS

An additional DSM is now introduced in the Jupiter–Saturn arc of the Cassini-like transfer of Sect. 8.5.7. The search space is analogous to that reported in Table 8.7, except for t_{JS}, which ranges from 1,600 to 3,000 days. The GASP–DSM–DA algorithm prunes efficiently the twelve-dimensional search space to 1e − 6% of the initial size. The CPU time is 2,000 s. The main results of this problem are listed below.

- Constraints: $\Delta v_{\mathrm{E,dep}} \leq 4$ km/s, $\Delta v_{\mathrm{V},1} \leq 1$ km/s, $\Delta v_{d,\,1} \leq 1$ km/s, $\Delta v_{\mathrm{V},2} \leq 0$ km/s, $\Delta v_{\mathrm{E}} \leq 0$ km/s, $\Delta v_{\mathrm{J}} \leq 0$ km/s, $\Delta v_{d,\,2} \leq 1$ km/s, $\Delta v_{\mathrm{S}} \leq 5$ km/s
- Total number of boxes = 1.41e12
- Boxes remaining after pruning = 2.23e4 (1.6e − 6%)
- Optimal objective function value = 8.276 km/s
- Optimal objective function value without DSM (GASP-DA result) = 8.619 km/s

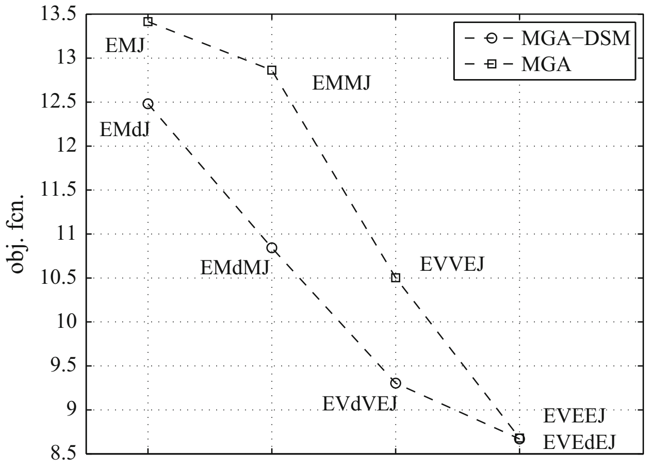


Fig. 8.13 Options for a transfer to Jupiter. Both the optimal MGA (*squares*) and MGA-DSM (*circles*) solutions are shown

8.6 Final Remarks

This chapter described how DSM can be inserted in the search space pruning process of the algorithm GASP. The proposed algorithm takes advantage of differential algebra, which is used to expand the constraint functions in Taylor series of the design variables. The adopted problem formulation limits the maximum functional dependency to eight variables, which is important to accurately bound the constraint functions. The resulting GASP–DSM–DA algorithm has been tested to solve relevant interplanetary transfer problems.

The introduction of the DSM into an MGA transfer deserves a final comment. Figure 8.13 compares the optimal objective function values for a mission to Jupiter, obtained with different transfer strategies. More specifically, GASP-DA and GASP–DSM–DA are used to compute the optimal solutions for pure MGA and MGA transfers with one DSM, respectively. Evidently, different transfer strategies have different costs. It is worth noting that, for the cases presented in Fig. 8.13, and generally for the MGA-DSM transfers, the introduction of DSM improves the optimal solutions in terms of transfer cost. On the other hand, MGA-DSM transfers involve longer overall transfer time.

Acknowledgements This research has been carried out under European space agency (ESA)/ Ariadna scheme, contract number: 20271/06/NL/HI. The authors would like to acknowledge the support of Tamás Vinkó and Dario Izzo from the Advanced Concepts Team of ESA. The authors are also grateful to Kyoko Makino for the support and help.

References

1. Yao, X.: Global optimization by evolutionary algorithms. In: Proceeding Of the Second Aizu International Symposium on Parallel Algorithm Architecture Synthesis, Aizu-Wakamatsu, Japan, IEEE Computer Society Press, pp. 282–291 (1997)
2. Ingberg, L.: Simulated annealing: Practice versus theory. Mathl. Comput. Model. **18**, 29–57 (1993)
3. Sentinella, M.R., Casalino, L.: Cooperative evolutionary algorithm for space trajectory optimization. Celestial Mech. Dyn. Astron. **105** (2009). DOI 10.1007/s10569-009-9223-4
4. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**, 157–181 (1993)
5. Jones, D.R.: A taxonomy of global optimisation methods based on response surfaces. J. Global Optim. **21**, 345–383 (2001)
6. Vasile, M., Summerer, L., De Pascale, P.: Design of Earth–Mars transfer trajectories using evolutionary-branching technique. Acta Astronautica **56**, 705–720 (2005)
7. Myatt, D., Becera, V., Nasuto, S., Bishop, J.: Advanced global optimisation for mission analysis and design. Final Report, Ariadna id: 03/4101, Contract No. 18138/04/NL/MV (2004)
8. Izzo, D., Becerra, V., Myatt, D., Nasuto S., Bishop, J.: Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. J. Global Optim. **38**, 283–296 (2006)
9. Armellin, R., Di Lizia, P., Topputo, F., Lavagna, M., Bernelli-Zazzera, F., Berz, M.: Gravity assist space pruning based on differential algebra. Celestial Mech. Dynam. Astron. **106**, 1–24 (2010)
10. Berz, M.: Modern Map Methods in Particle Beam Physics. Academic, New York (1999)
11. Berz, M.: The new method of TPSA algebra for the description of beam dynamics to high-orders. Technical Report AT-6:ATN-86-16, Los Alamos National Laboratory (1986)
12. Berz, M.: The method of power series tracking for the mathematical description of beam dynamics. Nucl. Instrum. Meth. **A258**, 431–436 (1987)
13. Berz, M.: Differential Algebraic Techniques. In: Entry in Handbook of Accelerator Physics and Engineering. World Scientific, New York (1999)
14. Berz, M., Makino, K.: COSY INFINITY version 9 reference manual. MSU Report MSUHEP-060803, Michigan State University, East Lansing, MI 48824, 1–84 (2006)
15. Giorgini, J.D., Yeomans, D.K., Chamberlin, A.B.., Chodas, P.W., Jacobson, R.A., Keesey, M.S., Lieske, J.H., Ostro, S.J., Standish, E.M., Wimberly, R.N.: Horizons, JPL's On-Line Solar System Data and Ephemeris Computation Service. User's guide (1998)
16. Battin, R.H.: An Introduction to the Mathematics and Methods of Astrodynamics, 2nd Printing. AIAA Education Series, Providence (1987)
17. Bernelli-Zazzera, F., Berz, M., Lavagna, M., Armellin, R., Di Lizia, P., Topputo, F.: Global Trajectory Optimisation: Can We Prune the Solution Space when Considering Deep Space Maneuvers? Final Report, Ariadna id: 06/4101, Contract No. 2007/06/NL/HI (2006)
18. Vasile, M., De Pascale, P.: Preliminary design of multiple gravity-assist trajectories. J. Spacecraft Rockets **43**, 794–805 (2006)

Chapter 9
A Mixed Integer Linear Programming Model for Traffic Logistics Management at the International Space Station

Giorgio Fasano

Abstract The operations of the International Space Station (ISS) pose many challenging issues, including logistics. The on orbit stay of the ISS is to be significantly extended in the near future: there will be an increased experimental activity in microgravity, giving rise to a renewed interest also in the related optimization aspects. A permanent logistic support is necessary to guarantee key ISS operations, as well as the scientific activities performed on-board. A traffic model, based on a mixed integer linear programming (MIP) approach, has been adopted to carry out the requested logistic planes. This chapter discusses the MIP model and provides insights concerning its application context.

Keywords International Space Station (ISS) • Logistic support • European automated transfer vehicle (ATV) • Traffic model • On-board resource resupply • Mixed integer linear programming (MIP) based modeling approach

9.1 Introduction

A major space venture witnessed by the entire world in the last three decades is the International Space Station program (ISS, www.nasa.gov). The ISS research program is conducted by the space agencies of the USA (NASA), Russia (RKA), Europe (ESA), Japan (JAXA), Canada (CSA), and Italy (ASI). Cited from the Canadian Space Agency's web site, "Since the first module of the Station was

G. Fasano (✉)
Thales Alenia Space Italia S.p.A., Str. Antica di Collegno 253, 10146 Turin, Italy
e-mail: giorgio.fasano@thalesaleniaspace.com

launched in 1998, the ISS has circled the globe 16 times per day at 28,000 km/h at an altitude of approximately 400 km, covering a distance equivalent to the Moon and back daily." For details and extensive further references related to the ISS, visit the websites www.nasa.gov, www.roscosmos.ru, www.esa.int, www.jaxa.jp, www.asc-csa.gc.ca, and www.asi.it/en.

The worldwide space environment, involving both national agencies and the whole topical industry, shows a renewed interest in the ISS, as its dismantling is expected to be postponed quite significantly into the future. Regardless of a prediction about the precise time the event will actually occur, a growing momentum is aimed at taking advantage of this new opportunity as much as possible, with its expanding horizons for science and technology.

This prospective scenario has a direct influence on two different aspects: namely, ISS research options and technological support requirements. The first one of these aspects is expected to look into new and promising perspectives in the microgravity experimental field; the second aspect is to provide a significantly improved capability to carry out the requested on-board activities. Both of these are expected to give rise to challenging cost-effectiveness and benefit maximization issues. The logistic support optimization hence becomes a major goal, requiring the use of cutting-edge methodologies.

Benefiting from its optimization expertise in supporting space logistics and flight operations, in the recent past, Thales Alenia Space has developed an ad hoc traffic model [3, 4]. This model, supported also by ESA, investigates the potential contribution of the European automated transfer vehicle (ATV) to the overall ISS traffic logistics scenario, considering also the already existing international fleet of vehicles.

The traffic model has been developed by adopting an optimization (mathematical programming) framework. A mixed integer linear programming (MIP) formulation has been followed, representing the problem in terms of lot sizing in the presence of additional constraints (cf., e.g., [6]). Different versions of the basic mathematical model have been studied, depending on the specific optimization criteria relevant to the current analysis, including also time-scale considerations.

Section 9.2 describes the ISS logistics problem related to introducing the ATV. We will discuss the operational constraints posed by the ISS configuration, its orbit-keeping requirements, and the launcher/vehicle characteristics. Section 9.3 presents the model formulation, and Section 9.4 offers insight related to the application context.

A possible extension of the presented traffic model will be investigated, in order to provide an enhanced analytical environment to look into the increasingly more challenging logistic problems of the ISS, as foreseen in the near future. We are convinced that our logistics modeling approach offers a valuable methodological starting point to tackle the challenges of future manned and unmanned interplanetary missions.

9.2 Logistic Support and Traffic Issues

The ISS has a sizeable infrastructure consisting of pressurized elements such as crew quarters, laboratories, nodes, and service modules, as well as a truss structure to accommodate external locations for experiments, solar panels, and radiators. It also provides facilities to allow externally mounted equipment to perform observations (of the Earth, the Sun, and other stars) and environmental monitoring operations. The ISS is permanently habituated by 6–7 crew members, and it provides a "shirt-sleeve environment" to perform scientific and technological experiments in microgravity conditions. The relative to Earth operational orbit must always be between 410 and 450 km altitude.

The permanent human presence aboard the ISS to maintain its standard operational conditions, to exploit the microgravity conditions as much as possible, to perform scientific experiments, and to keep the allowed orbital altitude and the very limited resource availability has led to the necessity of a continuous flow of materials—including hardware and fluids—between the Earth and the ISS.

The overall operational scenario of the ISS maintenance and utilization requires, time after time, careful logistic planning, based on different timescales, and the flexibility to afford last minute updates and quick rearrangements.

The first key issue is on-orbit resource re-supply (or "upload"), in order to permanently guarantee for the crew a habitable environment and provisions, to execute the requested payload (i.e., experimental facility) operations, as well as to perform overall maintenance. In addition to these, a periodical on-orbit intervention is required to allow for recurrent re-boosting operations to maintain the ISS orbit altitude within its admissible range, since—owing to the atmospheric drag—altitude tends to decrease. The listed issues and requirements necessitate an accurate upload and re-boosting plan. The re-supply material uploaded to the ISS is generally indicated either as pressurized, when transported in a vehicle that ensures a pressurized environment for the load, or unpressurized, when it does not provide any pressurization; a third key category of upload cargo is represented by the fuel.

A similar key issue is "download": that is, the retrieval of the experimental material for post-processing on Earth and the collection of the trash produced on-board that has either to be returned to Earth or destructively deorbited in the atmosphere. Hence, the download cargo consists both of experimental material and trash.

A fleet of launchers and vehicles is available to provide the ISS with the requested logistic support, including the re-boosting capability. Different launchers are associated to the vehicles, and the resulting transportation systems differ from each other with respect to their specific characteristics. Several aspects have to be taken into account such as the necessary equipment for crew transportation, the maximal reachable altitude, the possibility of returning cargos to Earth, the maximum upload and download capacity, the minimum elapsed time between two subsequent launches, the maximum admissible number of flights per year, as well as the maximum on-orbit stay time.

Strong further impacts on the overall planning task come up, as the payload operations under microgravity conditions must be nominally guaranteed for at least a minimum number of periods per year: this implies constraints on the minimum elapsed time between one launch and the following one. In addition to all the above, tight safety requirements have to be met. In case of a skipped vehicle arrival (for whatever reason such as a missed rendezvous) the ISS has to survive up to the next possible launch. This determines at least the provision of the necessary crew re-supply and the fuel amount to guarantee the contingency re-boosting intervention (by activating the onboard motors).

The ISS itself introduces mandatory constraints deriving from its configuration: the quite limited on-board re-supply capacity for each kind of resource and the capability to allow the simultaneous presence of different cargo carriers, strictly depending on the current docking location available on board.

It is obvious that the overall problem to tackle is very complex: the large number of decision alternatives (variables) and constraints to satisfy, as well as the necessity to perform the requested analysis swiftly and efficiently, simply cannot be faced by a "paper-and-pencil" approach.

The traffic model presented below is aimed at accomplishing this task successfully, by determining for each given analysis period a logistic plan that is optimized with respect to a selected criterion relevant to the current scenario to look into. The model's specific scope is to define the following characteristics:

- The Earth-to-orbit vehicle launch and departure times;
- The cargo each such vehicle has to deliver and return, for each load typology;
- The activation time and duration of the re-boosting phases;
- The vehicle to utilize and the relative orbit altitude retrieval;
- The onboard resource availability time profile (in particular, the final state) for each resource typology;
- The onboard trash accumulation time profile (in particular, the final state);
- The ISS altitude time profile (in particular, the final state).

The target of such analysis (time after time) may focus on different aspects. In particular, one of the possible plans can address the determination of feasible operational scenarios at a minimum cost. Other plans can be adopted as well, in order to better fit the perspective to look into the comparison of different solutions.

The ISS operational scenario presented hereinafter refers to the situation occurring when the automated transfer vehicle is introduced, as an additional transportation option together with the existing fleet. The framework is still representative of the current situation, but significant changes and enhancements have to be foreseen from now on, in a perspective of an ever more effective exploitation of the ISS.

The traffic model discussed in this chapter was developed to provide ESA with the capability of investigating the operational scenarios derived from the addition of the ATV to the preexisting fleet. Such a relative late insertion implied several issues. A traffic model had been developed by NASA [1] with flight predictions for vehicles provided by NASA and RKA. The introduction of the ATV within such an established scenario had to prove to be an actual improvement over the previous status.

Different prospective ATV missions had to be considered, in order to tailor properly the vehicle features with respect to the expected ISS needs. The candidate missions should consider different cargo capabilities and typologies, such as the delivery of unpressurized cargo only; the delivery of pressurized cargo only; the delivery of re-boosting fuel only; the delivery of a mix of pressurized and unpressurized cargo and re-boosting fuel.

The ATV design should be goal oriented, depending on the expected vehicle contribution to the ISS, both in terms of mission typology and quantitative support. Specifically, the possible partition between pressurized and unpressurized cargo and fuel had to be determined.

As the ISS program was already in a phase where changes in Earth-to-orbit vehicle performance and parameters, as well as in logistic requirements happened quite often, the need to reiterate the analysis at a very high rate was understood.

9.3 The Mathematical Model

The traffic problem described in the previous section can be considered in terms of optimal control. Applying this point of view, the state variables represent the resource and trash accumulated on board at any time, as well as the ISS altitude; the upload and download mass and the re-boosting activity correspond to the control variables. All of the above model components are subject to transportation and operational constraints implied by the vehicle characteristics, the ISS storage capacity and maintenance requirements. The model formulation can then be carried out by introducing a set of mathematical submodels governed by a state (vector) equation:

$$\frac{\mathrm{d}s}{\mathrm{d}t} = f[s(t), t],$$

defined on each subinterval $\left[t_j, t_{j+1}\right]$ $(j \in \{0, \ldots, n-1\} = J)$ the overall analysis period [0,T] is partitioned into (by assumption, $t_0 = 0$ and $t_n = T$). The terms t_j are control variables, representing the time moments when control actions (i.e. upload, download and re-boosting) $u_0, \ldots, u_{n-1}$ are taken. The vector $s(t)$ represents the state variables as functions of t. For each sub-interval $\left[t_j, t_{j+1}\right]$, the initial conditions below are stated:

$$s(t_j) = s_j + u_j,$$

meaning that the state s_j, of $s(t)$ at the end of the previous sub-interval $\left[t_{j-1}, t_j\right]$, is instantaneously modified by the control action associated to the control variable (vector) u_j. For each $t \in [0,T]$ and $j \in J$, lower and upper bounds are imposed on the state variables:

$$\underline{S} \leq s(t) \leq \overline{S},$$

while further conditions encompassing launch windows, vehicle availability and cargo capacity act as additional constraints for the control variables.

An objective function with the following general form is then defined:

$$\min_{t_1,\ldots,t_{n-1};u_0,\ldots,u_{n-1}} \left\{ \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} p[s(t), u_j, t]\mathrm{d}t + q(t_1, \ldots, t_{n-1}, u_0, \ldots, u_{n-1}, s_n) \right\},$$

where the functions p and q have to be specified, depending on the optimization criterion selected (e.g., integral averages), and the (vector) variable s_n represents the final system state at time T.

The approach described in a simplified form next look upon a discretization of the variables, constraints, and objective function, based on the total time period partition into n subintervals (still denoted as $[t_j, t_{j+1}]$) of prefixed duration (consult, e.g., Tabak and Kuo [5]). This approach is suitable to deal with both continuous and integer variables, implicitly included in the problem. In such a way, the original problem assumes the form of a lot sizing model, in the presence of additional constraints. As a matter of fact, from the logistic point of view, the ISS can be essentially interpreted and modeled as a "warehouse," with given storage capacity for each type of goods to consider, the resource consumption, and the orbit altitude, as the customer demand for each sub-period, the earth-to-orbit vehicles as the transportation means employed, with given load capability for each type of product.

In the following, all variables are assumed to be nonnegative. The state equations concerning the on-board resources (excluding fuel) are described here below:

$$\forall \alpha \in A, \forall j \in J \quad r_{\alpha(j+1)} = r_{\alpha j} - D_\alpha + \sum_{i \in I} r^*_{\alpha ij}, \tag{9.1}$$

where the variables $r_{\alpha j}$ represent for each $\alpha \in A$ (the set of resource types) the amount of the relative resource at time j; D_α the resource α demand per time interval, assumed to be a positive constant and $r^*_{\alpha ij}$ the resource α quantity transported on board by the vehicle $i \in I$ (set of vehicles) at time j.

Similar equations hold for the amount of trash present on-board:

$$\forall j \in J \quad w_{j+1} = w_j + \sum_{\alpha \in A} Q_\alpha - \sum_{i \in I} w^*_{ij}, \tag{9.2}$$

where Q_α , a positive constant, stands for the trash accumulation per time interval deriving from each resource α, and $w^*_{\alpha ij}$ the trash quantity downloaded by vehicle *i*, at time *j*.

As far as the re-boosting activity is concerned, the question is quite tricky, as two different kinds of fuel have to be taken into account. The first can be utilized by the vehicles to perform a re-boosting action, but it has to be stored onboard, always guaranteeing a minimum level, in order to provide the ISS with the necessary attitude control and be available, as possible reserve, to operate re-boosting intervention (by the ISS motors), in case of contingency. In the following, this fuel shall be called extended-use fuel and denoted by $\hat{f}$. The second one, on the contrary, can only be utilized by a vehicle in a (nominal) re-boosting phase. It is denoted as re-boosting fuel and referred to as f. The following equations are then stated for both fuels, respectively:

$$\forall j \in J \quad \hat{f}_{j+1} = \hat{f}_j - C_j + \sum_{i \in I} \hat{f}^*_{ij} - \sum_{i \in I} \hat{u}_{ij}, \tag{9.3}$$

$$\forall i \in I, \forall j \in J \quad f_{i,j+1} = f_{ij} + f^*_{ij} - u_{ij}. \tag{9.4}$$

In Eq. (9.3), referring to the extended-use fuel, for each time j, C_j is the consumption per time interval due to the (nominal) attitude control, $\hat{f}_{ij}^{\,*}$ is the amount uploaded by vehicle *i,* and $\hat{u}_{ij}$is the quantity utilized by vehicle *i* for re-boosting purposes.

In Eq. (9.4), referring to the re-boosting fuel, for each time j, f^*_{ij} is the amount uploaded by vehicle *i* and u_{ij} is the quantity utilized by vehicle *i*, at time *j*, for re-boosting purposes. These equations are stated to take account of the quantity of fuel stored on board the vehicles temporarily docked at the ISS and their relative utilization for re-boosting activity during all periods (time intervals) of permanence as attached modules.

The equations shown below address the ISS altitude trend:

$$\forall j \in J \quad h_{j+1} = h_j - L_j + \sum_{i \in I} (\hat{u}_{ij} \hat{P}_i + u_{ij} P_i), \tag{9.5}$$

where for each time j, h_j represents the altitude, L_j its loss per interval, assumed to be a constant (following an approximate linearization), $\hat{P}_i$ and P_i are the altitude increments per fuel unit, for the two fuel typologies, respectively, as utilized by vehicle *i*.

A quantity of mass, associated to the experimental material that is supposed to be post-analyzed on the ground, is permanently present onboard, prior to be downloaded by a vehicle. It is called processed mass and its trend is described by the following equations:

$$\forall \beta, \forall j \in J \quad e_{\beta,\, j+1} = e_{\beta j} + Y_{\beta j} - \sum_{i \in I} e^*_{\beta ij}, \tag{9.6}$$

where, for each time j, $e_{\beta j}$ represents the quantity of experimental material β, with $\beta \in B$ (set of experimental material typologies), $Y_{\beta j}$ its yield per subinterval (taking into account the transformation of the original mass in processed, consumed, and residual trash), assumed as constant, and $e^{*}_{\beta ij}$ is the processed mass, downloaded by vehicle i.

The following bounds are stated to guarantee, at any time, the minimum allowable level for each resource type:

$$\forall \alpha \in A, \forall j \in J \quad r_{\alpha j} \geq \underline{R}_{\alpha}. \tag{9.7}$$

As mentioned in the previous section, moreover, the ISS orbit altitude must be included, at any time, within a given range, as this is implied by the lower and upper bounds below:

$$\forall j \in J \quad \underline{H} \leq h_j \leq \overline{H}. \tag{9.8}$$

The above conditions, indirectly, in addition to the ISS attitude control requirements, give rise to the following constraints relative to the extended-use fuel available onboard that must hold at any time:

$$\forall j \in J \quad \hat{f}_j \geq \underline{\hat{F}}. \tag{9.9}$$

Upper limits on the ISS onboard storage capacity are given, taking into account the possible presence of attached vehicles during the whole time subinterval $\left[t_j, t_{j+1}\right]$. The 0–1 (indicator) variable σ_{ij} is then introduced, with the following meaning:

$\sigma_{ij} = 1$ if vehicle i is attached during the interval $\left[t_j, t_{j+1}\right]$
$\sigma_{ij} = 0$ otherwise.

The following upper bounds are then stated to take into account, at any time, the on-board storage capacity for each resource typology:

$$\forall \alpha \in A, \forall j \in J \quad r_{\alpha j} \leq \overline{R}_{\alpha} + \sum_{i \in I} R^{*}_{\alpha i} \sigma_{ij}, \tag{9.10}$$

where $R^{*}_{\alpha i}$ is the maximum quantity of mass relative to each resource type α that can be loaded (i.e., stored on orbit) by vehicle i.

Similarly, the following conditions hold, with an obvious meaning of the symbols introduced:

$$\forall j \in J \quad w_j \leq \overline{W}_{\alpha} + \sum_{i \in I} W^{*}_{\alpha i} \sigma_{ij}. \tag{9.11}$$

Analogously, the constraints below are posed concerning the extended-use fuel stored onboard:

$$\forall j \in J \quad \hat{f}_j \leq \overline{\hat{F}} + \sum_{i \in I} \hat{F}_i^* \sigma_{ij}, \tag{9.12}$$

where $\overline{\hat{F}}$ is the maximum fuel loadable by the ISS and $\hat{F}_i^*$ is the maximum loading capacity relative to vehicle i.

For similar reasons, the following conditions are added, with an obvious meaning of the symbols:

$$\forall i \in I, \forall j \in J \quad f_{ij} \leq F_i^* \sigma_{ij} \tag{9.13}$$

A threshold relative to the overall mass loaded, at any time, on the ISS, including the fuel of the vehicles temporary attached, obviously has to be considered. This is achieved by adding the following constraints (based on an approximation):

$$\forall j \in J \quad \sum_{\alpha \in A} r_{\alpha j} + \hat{f}_j + \sum_{i \in I} f_{ij} + \sum_{\beta \in B} e_{\beta j} + w_j \leq M. \tag{9.14}$$

To control the launch and return activities, as well as their relative upload and download actions, the following 0–1 variables are further introduced, with their corresponding logical meanings:

$\lambda_{ij} = 1$ if vehicle i is launched (and reaches the ISS) at time j,
$\lambda_{ij} = 0$ otherwise;
$\rho_{ij} = 1$ if vehicle i performs its reentry from the ISS at time j,
$\rho_{ij} = 0$ otherwise.

A number of constraints have then to be considered, in order to take into account the upload capability relative to the utilization of each vehicle. As a first step, it has to be pointed out that a maximum altitude limit $H_i \in [\underline{H}, \overline{H}]$ is associated to each vehicle i. The constraints below are then introduced to guarantee that if, at a certain time j, the ISS altitude is higher than the limit associated to a vehicle, it cannot be launched within the corresponding subinterval:

$$\forall i \in I, \forall j \in J \quad \lambda_{ij} \leq 2 - \frac{h_j}{H_i}. \tag{9.15}$$

A further set of constraints, whose meaning is obvious, involves both the launch, reentry, and on-board stay decisional variables:

$$\forall i \in I \quad \sum_{j \in J} \lambda_{ij} \leq 1, \tag{9.16}$$

$$\forall i \in I \quad \sum_{j \in J} \rho_{ij} \leq 1, \tag{9.17}$$

$$\forall i \in I \quad \sum_{j \in j} \lambda_{ij} \leq 1 - \Lambda_i, \tag{9.18}$$

$$\forall i \in I, \forall j \in J/j \geq 1 \quad \rho_{ij} \leq \sum_{j' \leq j} \lambda_{ij}{}' - \Lambda_i, \tag{9.19}$$

$$\forall i \in I, \forall j \in J \quad \sigma_{ij} \leq \sum_{j' \leq j} \lambda_{ij}{}' + \Lambda_i, \tag{9.20}$$

$$\forall i \in I, \forall j \in J \quad \sigma_{ij} \leq 1 - \sum_{j' \leq j} \rho_{ij}{}', \tag{9.21}$$

$$\forall i \in I, \forall j \in J \quad \sigma_{ij} \geq \sum_{j' \leq j} \lambda_{ij}{}' - \sum_{j' \leq j} \rho_{ij}{}' + \Lambda_i, \tag{9.22}$$

where Λ_i is a parameter equal to one, if at the initial time ($t = 0$) vehicle i is attached to the ISS and zero otherwise. Further conditions, representing the docking rules, should also be introduced, but they are not reported here, for the sake of simplicity.

The following constraints express the conditions that if at time j the vehicle i is not launched, its cargo overall capacity is null:

$$\forall i \in I, \forall j \in J \quad \sum_{\alpha \in A} r^*_{\alpha ij} + \hat{f}^*_{ij} + f^*_{ij} \leq M_i \lambda_{ij}, \tag{9.23}$$

where M_i is the maximum mass capacity associated to vehicle i, corresponding to the altitude (the minimum admissible) $\underline{H}$. When vehicle i is, however, supposed to reach a higher altitude, its maximum capacity decreases (as an approximation) linearly, with rate $K_i > 0$. The following constraints are then posed (noting that they become redundant when the corresponding λ_{ij} is zero):

$$\forall i \in I, \forall j \in J \quad \sum_{\alpha \in A} r^*_{\alpha ij} + \hat{f}^*_{ij} + f^*_{ij} \leq M_i - (h_j - \underline{H}) K_i. \tag{9.24}$$

Considering the different loading capacity characteristics, with reference to each single cargo typology, the following constraints are then stated:

$$\forall \alpha \in A, \forall i \in I, \forall j \in J \quad r^*_{\alpha ij} \leq R^*_{\alpha ij} \lambda_{ij}, \tag{9.25}$$

$$\forall i \in I, \forall j \in J \quad \hat{f}^*_{ij} \leq \hat{F}^*_i \lambda_{ij}, \tag{9.26}$$

$$\forall i \in I, \forall j \in J \quad f^*_{ij} \leq F^*_i \lambda_{ij}. \tag{9.27}$$

Furthermore, since the vehicles have generally a limited pressurized cargo capacity, the following constraints hold:

$$\forall i \in I, \forall j \in J \quad \sum_{\alpha \in A'} r^*_{\alpha ij} \leq M'_i \lambda_{ij}, \tag{9.28}$$

where $A' \subset A$ is the subset of pressurized resource typologies and M_i' is the relative loading upper bound, corresponding to vehicle i.

Similar cargo constraints have to be formulated as well, in order to include in the model the loading limitations occurring during the reentry phases. They are, however, omitted here for the sake of a more concise exposition.

As mentioned in the previous section, a number of microgravity periods, ranging from a minimum and a maximum denoted by $\underline{N}, \overline{N}$ respectively, have to be foreseen. The following 0–1 variables are then aimed at determining the microgravity/re-boosting state associated to each subinterval:

$\mu_j = 1$ if the sub-interval $[j, j+1]$ is a microgravity period,
$\mu_j = 0$ otherwise.

The condition below expresses that the number of microgravity periods has to be in compliance with its given range:

$$\underline{N} \leq \sum_{j \in J} \mu_j \leq \overline{N}. \tag{9.29}$$

The constraints shown next have thus the scope of guaranteeing that during each microgravity period, no launch, re-enter, or re-boosting activity can be executed:

$$\forall i \in I, \forall j \in J \quad \lambda_{ij} \leq 1 - \mu_j, \tag{9.30}$$

$$\forall i \in I, \forall j \in J \quad \rho_{ij} \leq 1 - \mu_j, \tag{9.31}$$

$$\forall i \in I, \forall j \in J \quad \hat{u}_{ij} \leq \hat{F}_i^*(1 - \mu_j), \tag{9.32}$$

$$\forall i \in I, \forall j \in J \quad u_{ij} \leq \hat{F}_i^*(1 - \mu_j). \tag{9.33}$$

The values of the state variables, corresponding to the initial time, have to be taken into account. They are represented, for each resource type, the trash, the external-use fuel, and the altitude, respectively, by the terms $R_{\alpha 0}$, W_0, $\hat{F}_0$ and H_0. The overall initial state is then expressed by the following boundary conditions:

$$\forall \alpha \in A \quad r_{\alpha 0} = R_{\alpha 0}, \tag{9.34.1}$$

$$w_0 = W_0, \tag{9.34.2}$$

$$\hat{f}_0 = \hat{F}_0, \tag{9.34.3}$$

$$h_0 = H_0. \tag{9.34.4}$$

On the other hand, the final state conditions corresponding to time T relative to all the resources present on board the ISS, the extended use fuel, as well as the altitude represent, generally, a major concern of the various analysis scenarios.

Quite a frequently requested overall restriction is that at the end of the time period, the ISS resources (including the fuel) and altitude must not result at their minimum admissible level (and similarly the trash must not be at its on-board storage upper bound). For such a reason proper lower limits have to be set from time to time, in compliance to the specific context to investigate. They are summarized here below:

$$\forall \alpha \in A \quad r_{\alpha T} \geq \underline{R}_{\alpha T}, \tag{9.35.1}$$

$$w_T \leq \underline{W}_T, \tag{9.35.2}$$

$$\hat{f}_T \geq \underline{\hat{F}}_T, \tag{9.35.3}$$

$$h_T \geq \underline{H}_T. \tag{9.35.4}$$

As far as the objective function is concerned, different choices can be made depending on the specific analysis task under consideration. We report here one of the possible options that proved to be quite useful at a preliminary level, when some of the characteristics both of the ISS and even the main ATV features, such as, for instance, the load capacity relative to the different cargo typologies, were not yet consolidated. For this purpose, some of the model constraints were relaxed by introducing (non-negative) slack variables representing the relative constraint violation. The corresponding version of constraints (9.23), for instance, assumes then the form:

$$\forall i \in I, \forall j \in J \quad \sum_{\alpha \in A} r^*_{\alpha ij} + \hat{f}^*_{ij} + f^*_{ij} \leq M_i \lambda_{ij} + \tilde{m}_{ij}, \tag{9.36.1}$$

$$\forall i \in I, \forall j \in J \quad 0 \leq \tilde{m}_{ij} \leq \tilde{M}_i \lambda_{ij}, \tag{9.36.2}$$

where $\tilde{M}_i$ represents for each vehicle i the maximum relaxation admitted for the relative overall mass capacity. Since similar readjustments can obviously involve several other constraints of the model, the objective function, to be minimized, simply consists of the sum of the slack variables taken into account, time after time, on the basis of the current analysis task and properly weighted.

As mentioned beforehand, different criteria could be looked into depending each time on the specific merit perspective the problem is intended to be investigated from. Even if some interesting insights in this regard could be provided, they are not discussed here.

Let us mention that a more refined formulation of the problem, interpreted in terms of lot sizing with additional constraints, could be implemented adopting the approach of Eppen and Martin [2], in order to tighten the corresponding MIP model by restricting the relative LP-relaxation's convex hull. In the traffic model considered here, we limit ourselves to pointing out that each upload variable $r^*_{\alpha ij}$ could thus be substituted with $r^*_{\alpha ijl}$, associated to the relative resource mass uploaded at time j and utilized onboard at time l, in compliance with the condition:

$$\forall \alpha \in A, \forall i \in I, \forall j \in J \quad \sum_{l \geq j} r^*_{\alpha ijl} = r^*_{\alpha ij}.$$

It is of course understood that analogous substitutions could be applied also to the remaining upload and download variables. Although Eppen and Martin's formulation of the lot sizing problem is sharper than the classical one, it should also be observed that it increases the number of variables involved quite dramatically. As a consequence, when dealing with very large-scale instances, a trade-off analysis that considers the advantages derived from a tighter model vs. the actual model size is strongly recommended.

As a final topic of this chapter, it is noted that each analysis task refers to a specific time scale. When the sub-intervals are sufficiently short, it can be assumed that all control actions (i.e., upload, download, and re-boosting activities) occur either at their initial or final times. When, however, this assumption does not hold any longer, due to the length of the sub-intervals, then it is more suitable to consider in some way that control actions can occur during the entire subperiod. A set of necessary conditions can then be posed in an integral form. Adopting this approach, the following inequalities can, for instance, replace constraints (9.10):

$$\forall \alpha \in A, \forall j \in J \quad \frac{\int_{\Delta T} r_\alpha(t)\mathrm{d}t}{\Delta T} \leq \overline{R}_\alpha + \frac{\sum_{i \in I} R^*_{\alpha i} \tau_{ij}}{\Delta T},$$

$$\forall i \in I, \forall j \in J \quad \tau_{ij} \leq \Delta T \sigma_{ij},$$

where ΔT represents the duration of each sub-interval and $\tau_{ij}$the actual permanence of vehicle i during the whole sub-interval $\left[t_j, t_{j+1}\right]$. Similar substitutions can involve all the on-board storage capacity conditions. The integral term can easily be approximated by the trapezoidal rule as follows:

$$\int\limits_{\Delta T} r_\alpha(t)\mathrm{d}t \approx \frac{r_{\alpha j} + r_{\alpha (j+1)}}{2} \Delta T.$$

9.4 Application Aspects

This section provides some information concerning the overall operational scenario, beginning from the state-of-the-art context to deal with when the traffic model was initially developed to investigate the introduction of the ATV option within the preexisting vehicle fleet.

Table 9.1 Upload cargo categories

Pressurized cargo	Unpressurized cargo	Fuel
Crew supplies (food, clothes, etc.)	Logistic items (for maintenance)	Extended-use fuel
Logistic items (for maintenance)	Scientific equipment	Re-boosting fuel
Scientific equipment		
Air		
Water		
Oxygen (for extra-vehicular activity)		

Table 9.2 Re-supply requirements (preliminary scenario)

Cargo type	RKA cargo requirements (kg)	Western cargo requirements (kg)	Total
Pressurized cargo			
Crew supplies (food, clothes, etc.)	49,331	86,876	136,207
Logistic items (for maintenance)	23,415	60,520	83,935
Science equipment	34,500	155,408	189,908
Air	3,671	-	3,671
Water	7,773	15,257	23,030
Oxygen for extra-vehicular activity	9,125	-	9,125
Oxygen fuel	6,156	-	6,156
Others	33,538	-	33,538
Unpressurized cargo			
Logistic items (for maintenance)	10,880	64,585	75,465
Science equipment	5,500	25,900	31,400
Fuel	106,672	-	106,672

Table 9.1 provides the relative upload cargo categories, and Table 9.2 shows the projections of the material to be delivered to orbit over the subsequent 15 years. An important characteristic of the resources involved is that most of them are “flagged” with the relevant nationality and all are classified either as Western (i.e., owned by NASA, ESA, CSA, or JAXA) or Russian (owned by RKA). As a general rule, NASA has been supposed to deliver Western re-supply, while RKA the Russian one. At the same time, ESA was the only agency expected to transport to orbit, using the ATVs, both Western and Russian re-supply.

Table 9.3 reports for each Earth-to-orbit systems (i.e. launcher plus transfer vehicle) the type of cargo and the maximum upload capabilities that were expected at the average ISS altitude. Both the Mini Pressurized Logistic Module (MPLM, ASI) and the Unpressurized Logistic Carrier (ULC, NASA) are associated to the Shuttle (NASA); the ATV (ESA) utilizes Ariane 5 as its launcher, Progress-X (RKA) utilizes the Soyuz (RKA), HTV (JAXA) utilizes the H2 (JAXA), respectively.

Table 9.3 Earth-to-orbit transportation system upload characteristics (preliminary scenario)

ETO transportation system	Nationality	Maximum pressurized/ unpressurized upload (kg)	Maximum fuel (kg)	Total maximum (kg)
Shuttle + MPLM	USA	9,040 (pressurized only)	–	9,040
Shuttle + ULC	USA	9,833 (unpressurized only)	–	9,833
Soyuz + Progress-X	Russia	4,750	2,300	4,750
Ariane + ATV	Europe	6,000	5,000	9,000
H2 + HTV	Japan	6,000	–	6,000

The overall outcome of the analysis carried out to evaluate the possible contribution deriving from the ATV introduction within the previous fleet composition, can be summarized as follows.

The ATV mission that could better support the re-supply and return scenario was the one up to delivering to orbit the so-called mixed cargo (pressurized cargo plus fuel).

The introduction of a properly designed ATV mixed-cargo vehicle is seen as beneficial to the re-supply and return scenario, since such an ATV mission could replace two Progress-X missions or, alternatively, one single Progress-X if higher resource re-supply capacity were required. This replacement could alleviate the high Progress-X flight rate (from up to sever missions per year to five missions per year) that was considered a potential criticality for the whole ISS program. Such a replacement was also beneficial with respect to the number of the extended microgravity periods.

An on-orbit stay of 4 months was considered the shortest period for a proper exploitation of the ATV utilization; while a mission duration exceeding 6 months would have led to an overdesign of its pressurized module. The optimal share of cargo resulted in the following partition:

- 5,000–6,000 kg of pressurized cargo;
- 1,000 kg of extended-use fuel;
- 3,000–4,000 kg of re-boosting fuel.

Several scenarios were analyzed, focusing on different time-scales, operational options, and specific objective functions. Most of the cases proved to be quite challenging from the computational point of view. Instances dealing with an analysis period covering 1–3 years involved up to 100,000 constraints, 150,000 continuous variables, and 3,000 binary variables: solving such models required more than 10 h of computational time within a standard state-of-the-art MIP solver environment.

In the light of the prospective scenarios expected for the ISS exploitation, a new trend is oriented towards less precise but much quicker computational analyses, implying the development of a "smarter" version of the previous traffic model. A dedicated research effort is currently ongoing, focusing both on the model formulation and MIP strategy improvement.

Table 9.4 Case study overall launch/reentry plan

	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13
P03		🟩												
P05	🟩													
P08										🟩	🟩	🟩	🟩	🟩
P11					🟩	🟩								
AT1		🟩	🟩	🟩	🟩									
AT3						🟩	🟩	🟩	🟩	🟩				

	t14	t15	t16	t17	t18	t19	t20	t21	t22	t23	t24	t25	t26
P06												🟩	🟩
P07				🟩	🟩	🟩							
P09	🟩	🟩											
P10			🟩	🟩									
P12									🟩	🟩	🟩	🟩	
AT2							🟩	🟩	🟩				

	t27	t28	t29	t30	t31	t32	t33	t34	t35	t36	t37	t38	t39
P01								🟩	🟩				
P02	🟩	🟩	🟩	🟩									
P04									🟩				
AT4					🟩	🟩	🟩						

The reference frame for the experimental analysis under study consists essentially of the following environment, in terms of platform and MIP solver: personal computer(s) equipped with Core 2 Duo P8600, 2.40 GHz, 1.93 GB RAM, and running under Windows XP Professional Service Pack 2; IBM ILOG CPLEX Optimizer 12.3.

A case study, referring to the mathematical model currently under study, is shown next to give a qualitative representation of the result typologies. The analysis covers a 3-year period, partitioned in 36 months (sub-intervals). Since most of the information concerning the near future scenarios is still quite uncertain, in particular, with respect to the new fleet that is going to be set up, the previous one has been kept as basic reference, even if some of its transportation systems have already been dismissed (as the Shuttle, for example) or are quite close to their last mission (as the ATV).

The objective function consists of the sum of the two relaxation slack variables concerning the maximum ATV capacity associated to both kinds of fuel. The relative model instance contains about 51,000 rows, 17,000 continuous variables, and 1,200 binary variables.

Some of the overall case study results obtained in about 30 CPU minutes are summarized by the tables reported below. Table 9.4 illustrates the launch and reentry plan for each vehicle considered (in several cases they remain on orbit for more than one month). Table 9.5 reports the ISS pressurized cargo: the light bars indicate unprocessed mass, while the dark ones the processed mass. Table 9.6 refers

Table 9.5 Case study ISS on-board pressurized cargo trend (kg)

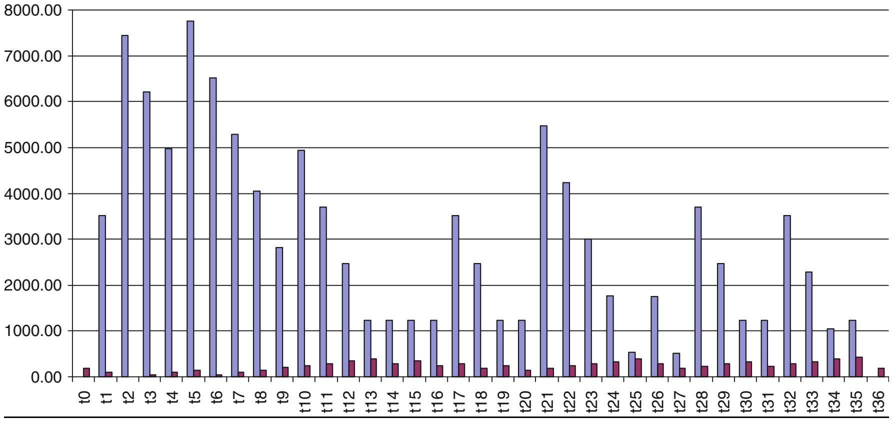


Table 9.6 Case study ISS on-board trash trend (kg)

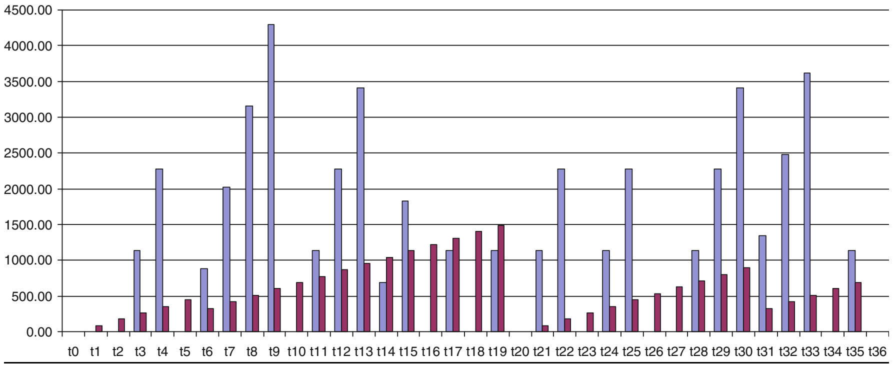


Table 9.7 Case study ISS on-board fuel trend (kg)

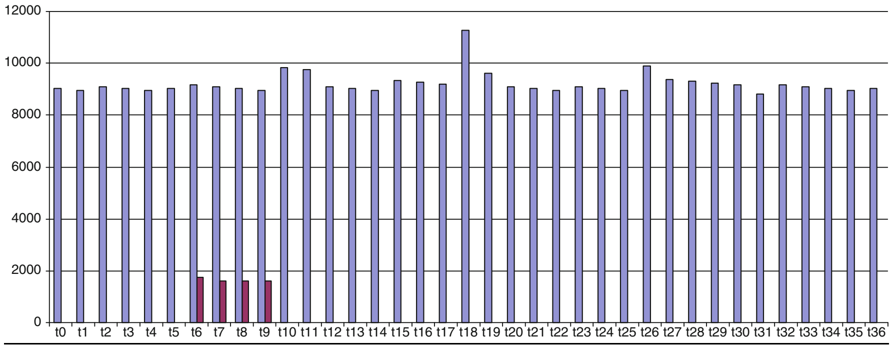


Table 9.8 Case study ISS altitude trend (km)

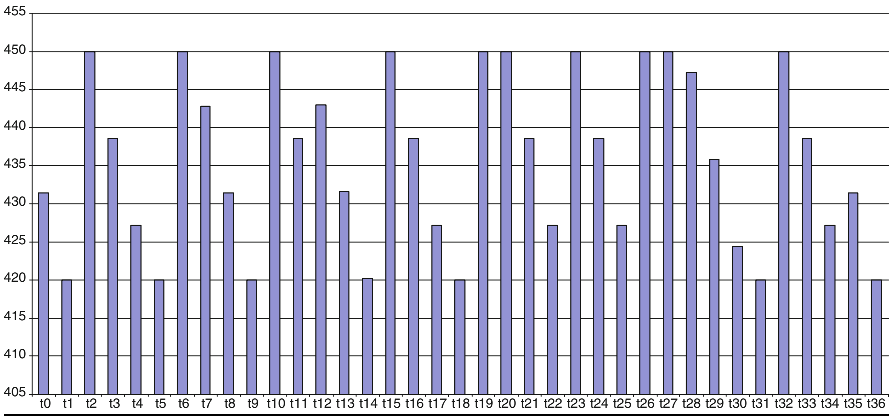


Table 9.9 Case study upload/download provided by a single vehicle (kg)

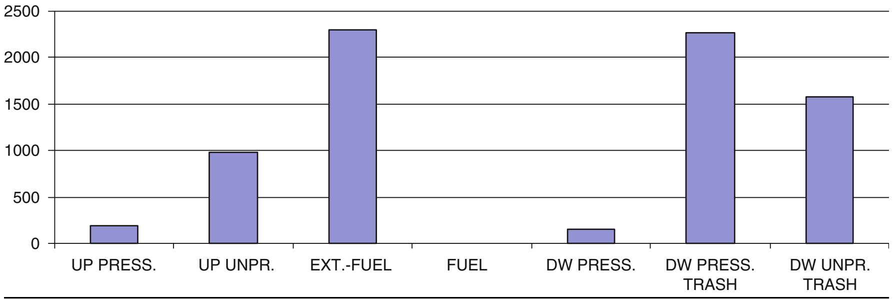
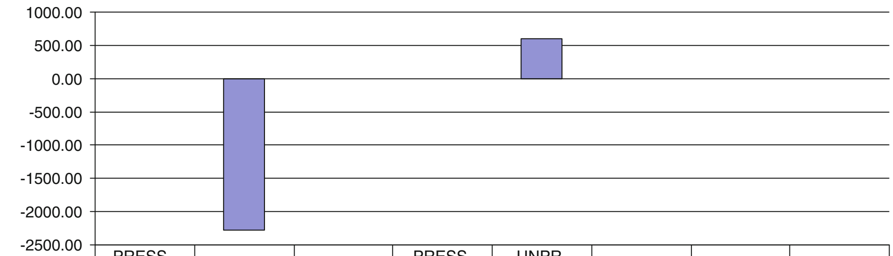


Table 9.10 Case study gap between initial and final ISS states



	PRESS.-NON-US.	PRESS.-US.	UNPR.	PRESS. TRASH	UNPR. TRASH	EXT.-FUEL	FUEL	ALTITUDE
▨ GAP (kg, km)	0.00	-2279.61	0.00	0.00	599.00	0.00	0	-11.40

to the trash trend. Table 9.7 shows the fuel present onboard the ISS: the light bars represent the external-use fuel, while the dark ones indicate the re-boosting one. Table 9.8 illustrates the trend relative to the ISS altitude. Table 9.9 points out the upload/download provided by a single vehicle and Table 9.10 the gap between the initial and final ISS states, during the whole analysis period.

9.5 Concluding Remarks

This chapter has been devoted to logistics and traffic issues that the International Space Station has given rise to, up to now and in the near future (with increasing complexity, as nowadays, its dismantling plan has been put off for some time). In the recent past, the issue was looked into, in particular by ESA in relation to the introduction of the Automated Transfer Vehicle (ATV) within the already existing international fleet of transportation systems. This challenging problem has been successfully tackled by developing and solving an MIP model in different context-specific versions. As the traffic problem embeds the flow of material to and from the ISS, as well as the sequence of missions of the various earth-to-orbit vehicles, including the re-boosting phases, the traffic model has been used in order to determine, for each given analysis period, an optimized logistic plan.

In this chapter we have presented a simplified version of the actual formulation and omitted a number of technical details. The model needs to be properly updated and extended as soon as the ongoing operational scenario, including a new fleet, has been determined. Specific research is foreseen both from the modeling and computational strategy points of view. Our discussion also offers practical insights, to interpret the problem in its real-world context.

References

1. Boeing: ISS Integrated Traffic Model Report—DAC 4 Final Report. www.boeing.com (1996)
2. Eppen, G.D., Martin, R.K.: Solving multi-item capacitated lot-sizing using variable redefinition. Oper. Res. **35**(6), 832–848 (1987)
3. Fasano, G.: Mixed integer linear approach for the space station on-orbit resources re-supply. In: AIRO 96 Annual Conference Proceedings, Perugia, 16–20 Sept 1996
4. Fasano, G., Provera, R.: A traffic model for the space station on-orbit re-supply problem. In: The International Society of Logistics Engineering (SOLE), Logistics National Conference Proceedings, Turin, 7–10 Oct 1997
5. Tabak, D., Kuo, B.: Optimal Control by Mathematical Programming. Prentice-Hall, New Jersey (1971)
6. Van den Heuvel, W.: Lot-sizing. In: Cochran, J.J. (ed.) Wiley Encyclopedia of Operations Research and Management Science. Wiley, New York (2010)

Chapter 10
Global Optimization Approaches to Sensor Placement: Model Versions and Illustrative Results

Giorgio Fasano and János D. Pintér

Abstract We investigate the optimized configuration of sensor cameras to be placed in a suitably defined three-dimensional cubic region E, in order to maximize the coverage of a completely embedded cube $C \subset E$. The sensing regions associated with each of the cameras are convex, but not necessarily identical. In order to handle this important practical problem, we present mixed integer linear programming (MILP) and mixed integer nonlinear programming (MINLP) model formulations and propose corresponding solution approaches. Illustrative numerical results are presented, and certain application aspects are also discussed.

Keywords Sensor camera network design • Continuous set covering problems in three-dimensions • Global optimization • Mixed integer linear and nonlinear model forms • Solution strategies • Illustrative numerical results

10.1 Introduction

The topic discussed here originates from a case study carried out in a space engineering context, within the framework of future Mars exploration studies. We consider a drone scale model aimed at stimulating the landing phase of a robot vehicle that has to be permanently monitored by a set of sensor cameras. This scenario leads to challenging camera placement optimization issues: in this

MSC Classification (2000) 68 T20, 90 C11, 90 C30, 90 C59, 90 C90

G. Fasano (✉)
Thales Alenia Space Italia S.p.A., Str. Antica di Collegno 253, 10146 Turin, Italy
e-mail: giorgio.fasano@thalesaleniaspace.com

J.D. Pintér
Pintér Consulting Services Inc., Halifax, Canada
e-mail: janos.d.pinter@gmail.com

study we describe a modeling and algorithmic solution approach followed by illustrative results.

The drone is supposed to move inside a three-dimensional (3D) virtual cube denoted by C. Our general objective is to cover C optimally by the union of the fields of view (FOW) of the set of cameras. The cameras can be of different types: they have to be placed outside of C but inside an external cube denoted by E that embeds C and has parallel facets to C. We assume that the (truncated) FOW of each camera is a rectangular-based pyramid, without considering sensor orientation constraints at this point. Then, in a possible modeling approach, our objective can be to minimize the total number (or total cost) of the sensors needed to assure complete coverage of C. Alternatively, we can maximize the volume (a proper subset of C) that is covered by a given set of devices; a total equipment cost constraint could also be considered. In this work we present optimization models related to the latter formulation.

Although the case study discussed in this chapter is related to a specific simulation framework, many similar optimization problems arise in the context of space engineering and in other areas. Related space engineering applications include future interplanetary explorations, such as rover remote control and the monitoring of surfaces or regions. Similar sensor layout optimization problems are considered also in the context of environmental quality monitoring, industrial process tracking, robot movement control, security systems design, telematic and telecommunication applications, and sports events broadcasting. In all these application areas, the monitoring or coverage efficiency, the collection, retrieval and interpretation of information, the available equipment, and the related setup and operational costs are the key driving factors.

The professional literature related to sensor networks and camera arrays is extensive. Without going into details here, we refer, e.g., to Akyildiz et al. [3]; Barbosa et al. [5]; Cardei et al. [10]; Cerpa et al. [11]; Chakrabarty et al. [12]; Dhillon et al. [15]; Dhillon and Chakrabarty [16]; Erdem and selaroff [17]; Fleishman et al. [18]; Fusco and Gupta [19]; Hörster and lienhart [25]; Howard et al. [26]; Kang and Golay [29]; Meguerdichian et al. [33]; Meguerdichian and Potkonjak [34]; Mittal and Davis [36]; Molina et al. [38]; Onur et al. [40]; Pottie and Kaiser [45]; Sahni and Xu [47]; Shen et al. [48]; Xing et al. [55]; Yao et al. [57]; Zhang et al. [59]; Zou and Chakrabarty [60–62]. A large variety of studies consider two-dimensional (2D) applications, but some discuss 3D applications: see e.g. Alam and Haas [4]; Huang et al. [27]; and Chen et al. [13]. The regions considered can be of circular (in 2D), spherical, conic, polyhedral, or arbitrary shape (in 3D): consult, e.g., Gao and Shi [20] for the general case.

Solution approaches range from deterministic to stochastic coverage optimization methods: see, for instance, Abrams et al. [1]; Buczak et al. [9]; Harada et al. [22]; Lazos and Poovendran [30]; Meguerdichian et al. [33]; Miorandi and Altman [35]; Rowe and Wettergren [46]; and Wang et al. [56].

The specific problem type considered here is a 3D continuous set covering problem [14]. Mathematical programming (optimization based) formulations are discussed, e.g., by Altınel et al. [2]; Biswand Ye [7]; Gentile [21]; So and Ye [49]; Sorokin et al. [50]; Venkatesh and Buehrer [54]; and Berry et al. [6].

Following this brief introduction, Sect. 10.2 discusses the solution approach chosen, and then describes two mathematical models developed to represent the problem at different levels of approximation. Section 10.3 discusses illustrative numerical results and some practically important application aspects. The current experimental results are related to the first mixed integer linear programming (MILP) model type; in future work we will address some useful model extensions.

10.2 Optimized Sensor Placements: MILP and MINLP Formulations

10.2.1 A Global Optimization Approach to Sensor Placement

The general solution approach proposed here is based on a global optimization (GO) point of view. The objective of GO is to find the absolutely best solution of optimization models that have also local optima. The general sensor network design problem clearly falls into this model category: hence, it is natural to consider GO-based solution strategies.

In recent decades, global optimization has grown into a mature field of mathematical programming. Consult, for instance, Bomze et al. [8]; Horst and Pardalos [23]; Horst and Tuy [24]; Liberti and Maculan [32]; Pardalos and Romeijn [41]; Pintér [42–44]; Strongin and Sergeyev [52]; and Zabinsky [58].

The sensor network design problem will be tackled in two steps, consisting of the solution of two optimization models. The first of these is formulated as a MILP (mixed integer linear programming) model, while the second one is a MINLP (mixed integer non-linear programming) model. Regarding the latter topic, consult, e.g., Mockus et al. [37]; Nowak [39]; and Tawarmalani and Sahinidis [53]. Li and Sun [31] discuss nonlinear integer programming, with pertinent notes also on GO and MINLP.

The MILP model formulation is aimed at finding a suboptimal solution that can be improved, if necessary, by solving a related MINLP. In fact, even if the problem could be formulated directly as an MINLP problem, the MILP step is highly recommended, since the numerical efficiency of GO algorithms often strongly depends on some well-chosen simplified model statement(s) and the resulting solution(s). In other words, the optimization results based on a reduced MILP formulation can provide an informative starting solution to the quite possibly even more challenging MINLP model and its solution process.

In the MILP model described next (Sect. 10.2.2), all cameras are supposed to be pre-oriented, although not necessarily in an orthogonal arrangement. As a consequence, the resulting solution (even if optimal in the MILP framework) may well be suboptimal when considering a more detailed MINLP point of view. The pre-orientation assumption is dropped in the MINLP model version (Sect. 10.2.3), allowing the search to proceed towards the global optimal solution(s) of the original problem.

In both mathematical models, the convex domain to cover by sensors is discretized by a set of its internal points. Then, given the maximal number of cameras that can be utilized, the overall coverage of these internal points is maximized. Different weights can be assigned to these internal points if such an approach is dictated by practical needs: the weights will then indicate the relative importance of the individual points (or of the regions represented by these points).

10.2.2 An MILP Model with Pre-oriented Sensor Configurations

We shall consider a convex domain C defined with respect to a predefined orthonormal (main) reference frame: the latter is given by its origin O and coordinate axes w_1, w_2, w_3. The internal volume of C is discretized by a suitably chosen set of n points $P = \{P_1,\ldots, P_n\}$. In the given reference frame, point P_i is represented by the coordinates $p_{i\beta}$ for $i \in \{1,\ldots,n\}$ and $\beta \in \{1,2,3\}$.

Next, we introduce a model for the set of sensor devices $J = \{1,\ldots,r\}$. The sensing capability of each device is characterized by a convex domain V_j. In other words, V_j represents the sensing region of device j. In our case, we will assume that V_j can be adequately modeled by a truncated pyramid. A local orthonormal reference frame with origin O_j and axes $w'_{j\beta}$, oriented in parallel with respect to the corresponding main frame coordinate axes w_β, is associated to each device j. Each local reference frame origin, identified by its coordinates $o_{j\beta}$ with respect to the main reference frame, is supposed to be contained in an external convex domain E that embeds C: $O_j \in E$ and $C \subseteq E$. In this local frame, the coordinates $V_{j\alpha\beta}$ identify the sensing region vertices $V_{j\alpha}$, for $\alpha \in \{1,..,s\}$. (Recall that in the case studied here both C and E are 3D cubes.)

For each device j, we define a parallelepiped C_j; then $C_{j\gamma\beta}$ will denote the coordinates of its vertices with respect to its local frame. We shall assume that each parallelepiped C_j embeds the corresponding sensing domain V_j. Moreover, for any position of O_j inside E, each point P_i of C is covered by C_j so that we have

$$V_j \subseteq C_j \text{ , and } C \subseteq C_j \text{ for } j \in \{1,\ldots,r\}.$$

Introducing next the non-negative variables $\lambda_{ij\alpha}, \lambda^*_{ij\gamma}$, the following convexity conditions hold:

$$\forall i, \forall j, \forall \beta \quad p_{i\beta} = o_{j\beta} + \sum_{\alpha} V_{j\alpha\beta}\lambda_{ij\alpha} + \sum_{\gamma} C_{j\gamma\beta}\lambda^*_{ij\gamma}, \tag{10.1}$$

$$\forall i, \forall j \quad \sum_{\alpha} \lambda_{ij\alpha} + \sum_{\gamma} \lambda^*_{ij\gamma} = 1. \tag{10.2}$$

From Eqs. (10.1) and (10.2) it follows that $\sum_{\gamma} \lambda^*_{ij\gamma} = 0$ implies $P_i \in V_j$.

Next, we introduce the indicator variables $\delta_{ij} \in \{0, 1\}$: if $\delta_{ij} = 1$, then $P_i \in V_j$. Note that $P_i \in C_j$ always holds, but $\delta_{ij} = 0$ does not imply the relation $P_i \in C_j - V_j$. The conditions stated above are thus expressed by the following equations:

$$\forall i, \forall j \quad \sum_{\alpha} \lambda_{ij\alpha} = \delta_{ij}. \tag{10.3}$$

In the application area discussed here, evidently there is no advantage of covering any internal point of C more than once, since such "overcovering" could negatively affect the coverage of other points. This aspect can be expressed by the additional conditions:

$$\forall i \quad \sum_{j} \delta_{ij} \leq 1. \tag{10.4}$$

Next, in our model, a threshold $R < r$ is set to limit the total number of devices used to cover C as well as possible. For this purpose, the new binary indicator variables $\chi_j \in \{0, 1\}$ are associated with each device j, and the conditions below are stated:

$$\forall i, \forall j \quad \delta_{ij} \leq \chi_j. \tag{10.5}$$

$$\sum_{j} \chi_j \leq R. \tag{10.6}$$

Recall that in the model introduced here the embedding convex domain E is assumed to be a parallelepiped. (E is actually a cube in the specific drone simulation case; this assumption facilitates the discussion and the numerical solution procedure, without loss of generality.) We can then set the following bounds on the position of each local reference frame O_j that needs to be inside E:

$$\forall i, \forall \beta \quad 0 \leq o_{i\beta} \leq E_\beta. \tag{10.7}$$

The conditions (10.1)–(10.7) provide a comprehensive formulation of the problem constraints. In order to tighten the model itself by limiting the relative search region for the solution process, the auxiliary conditions shown below can be also introduced:

$$\forall j \quad \chi_j \leq \sum_{i} \delta_{ij}. \tag{10.8}$$

This way we assume that the sensing region of each device selected covers at least one internal point of C. (In the final solution, all unselected devices can be simply dropped.) On the basis of the domain discretization gap adopted to generate the internal points P_i, an estimate of a lower bound R_o for the number of devices needed can be introduced by the following constraint:

$$\sum_j \chi_j \geq R_o. \tag{10.9}$$

Similarly, introducing a suitable upper bound R_j for the number of points that can be covered by device j, the following conditions can be added:

$$\forall j \quad \sum_i \delta_{ij} \leq R_j \chi_j. \tag{10.10}$$

Our eventual objective is to maximize the coverage of the set of points: this can be expressed by

$$\max \sum_{i,j} \delta_{ij}. \tag{10.11}$$

Remark 1 The variables χ_j could be considered as continuous in the interval [0,1], since conditions (10.5) and (10.8) imply that they are actually binary. If they are defined explicitly as binary variables, then their processing priority with respect to the other binary variables could give rise to different MILP solution strategies.

Remark 2 Conditions (10.4) imply that the cardinality n of the internal points set $\{P_i\}$ is an upper bound for the objective function (10.11).

Remark 3 Weight factors associated to each δ_{ij} could be simply introduced as coefficients in the objective function (10.11), if there is a requirement to differentiate the internal points based on their (assigned or perceived) importance.

10.2.3 An MINLP Model for Sensor Configurations Without Pre-orientation

If the assumptions regarding a prefixed sensor orientation are dropped, then the local reference frames of each sensor can have arbitrary orientation. Hence, although the components (10.2)–(10.11) of the MILP model introduced above are all kept, the relations (10.1) have to be properly extended to consider not only the translation of the local reference frames but also their possible rotation. To obtain this model extension, we introduce the following notation. For each discretization point P_i, as well as for the origins O_j and vertices $V_{j\alpha}, C_{j\gamma}$, a corresponding vector, is defined as shown below by (10.12.1)–(10.12.4).

$$P_i = \begin{pmatrix} P_{i1} \\ P_{i2} \\ P_{i3} \end{pmatrix} \tag{10.12.1}$$

$$O_j = \begin{pmatrix} o_{j1} \\ o_{j2} \\ o_{j3} \end{pmatrix} \tag{10.12.2}$$

$$V_{j\alpha} = \begin{pmatrix} V_{j\alpha 1} \\ V_{j\alpha 2} \\ V_{j\alpha 3} \end{pmatrix} \tag{10.12.3}$$

$$C_{j\gamma} = \begin{pmatrix} C_{j\gamma 1} \\ C_{j\gamma 2} \\ C_{j\gamma 3} \end{pmatrix} \tag{10.12.4}$$

For each local reference frame j, we shall denote by $q_{j\beta\beta'}$ the direction cosines of the local reference frame axes $w'_{j\beta}$ with respect to the main frame axes $w_{j\beta}$. Then, the following orthogonal rotation matrix is defined:

$$Q_j = \begin{pmatrix} q_{j11} & q_{j12} & q_{j13} \\ q_{j21} & q_{j22} & q_{j23} \\ q_{j31} & q_{j32} & q_{j33} \end{pmatrix} \tag{10.12.5}$$

Applying these notations, the relations (10.1) are extended by the vector equations shown below:

$$\forall i, \forall j \quad P_i = o_j + \sum_{\alpha} Q_j V_{j\alpha} \lambda_{ij\alpha} + \sum_{\gamma} Q_j C_{j\gamma} \lambda^*_{ij\gamma}. \tag{10.13}$$

Observe that the extended mathematical model becomes an MINLP, since the terms $\sum\limits_{\alpha} Q_j V_{j\alpha} \lambda_{ij\alpha}$ and $\sum\limits_{\gamma} Q_j C_{j\gamma} \lambda^*_{ij\gamma}$ are nonlinear.

10.3 Applications

10.3.1 Modeling Considerations in the MILP Formulation

Given n internal points and r possible devices of which at most R can be selected, the corresponding MILP model instance is characterized by the following key model size descriptors:

$O(nr)$, δ item-device assignment binary variables;
$O(r)$, χ device selection binary variables;
$O(3nr)$, convexity constraints.

If we consider practically inspired model instances, then even for moderate scale sensor location problems, the (simpler) MILP model described in Sect. 10.3.1 quickly becomes extremely hard to solve. Therefore, obtaining just a *satisfactory* solution within a reasonable computational time frame can be a challenging task.

A promising MILP strategy is currently under development to speed up the MILP solution process. Its basic concept consists of performing a recursive procedure that at each step perturbs the objective function by *suggesting* the solution found at the previous one. In other words, the objective function (10.11) is modified by adding the perturbation terms $k_{ij}\delta_{ij}$, and it is thus substituted by the following expression:

$$\max \sum_{i,j} (\delta_{ij} + k_{ij}\delta_{ij}) \tag{10.14}$$

In our experimentation framework, we set at each step $k_{ij} = 1$ if in the previous step point i has been assigned to device j and $k_{ij} = 0$ otherwise. (Alternative weights k_{ij} could also be considered to achieve different solution strategies, e.g., using $k_{ij}>1$.) More precisely, at the initial step, all k_{ij} terms are equal to 0, and then a first (not proven optimal) solution is looked for. At the second step, these terms are set to 1 if the corresponding δ_{ij} variables (in the solution of the initial step) have been given value 1; otherwise, they are kept at 0. Denoting by N_o the total number of points that in the initial solution step are associated to a camera (i.e., the corresponding $\delta_{ij} = 1$), the following constraint is added:

$$\sum_{i,j} \delta_{ij} \geq N_o. \tag{10.15}$$

This way, in the second step, the nonzero δ_{ij} variables are induced to maintain their active "decision-making" status, and the new solution will not be "too far" from the previous one. Notice that k_{ij} can be set to 1, even if the corresponding δ_{ij} is 0, provided that point i is actually covered by camera j; in any case, in the presence of several possible candidates, for each i, at most one k_{ij} is set to 1. Moreover, considering (10.15), the new solution cannot become inferior to the previous one. This way, a recursive process (adopting a depth-first strategy) is performed, until the number of points corresponding to non-zero δ_{ij} variables keeps on increasing.

10.3.2 An Illustrative Case Study

In the case study presented here, we consider two concentric cubes corresponding to the domains C and E referred to earlier: see Fig. 10.1. Recall that the field of view of each sensor device is modeled by a truncated pyramid which has a rectangular

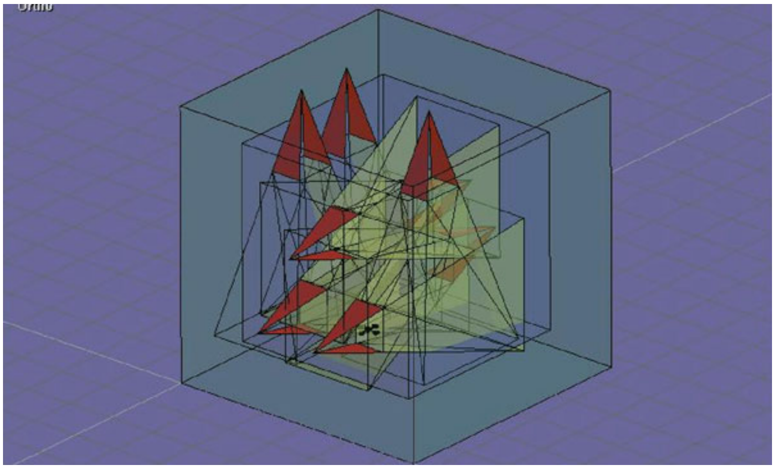


Fig. 10.1 Case study topology

basis. The key information for our numerical example is summarized by the following data:

Side length of C: 10 units;
Side length of E: 13 units;
Maximum number of cameras admitted: $R = 8$;
Number of internal points to observe: $n = 343$.

The MILP model presented in Sect. 10.3.1 has been applied with these data, taking into account a set of 12 identical sensor devices, each one with a prefixed orientation (that is parallel to the main reference frame). Figure 10.1 depicts an example of the possible solution topology: the configuration shown is similar to the actual case studied.

Our numerical experiments show that a suboptimal solution to the MILP model in which 339 points (out of 343) are covered can be obtained in about 1 CPU hour, by adopting the MILP solution strategy described in Sect. 10.3.1. An optimal solution (in which all 343 points covered) has been found in about 12 CPU hours. In these numerical experiments, a personal computer with a Core 2 Duo P8600, 2.40 GHz processor, and 2 GB RAM was used, running under Windows XP Professional (updated with Service Pack 2). The MILP solver used is the IBM ILOG CPLEX Optimizer Version 12.3 [28].

10.4 Concluding Remarks and Future Work

In this chapter a research activity related to an important space engineering application has been summarized. We consider the problem of sensor network placement, in order to cover a 3D convex region as well as possible. Dictated by the multimodal structure of this general problem type, our modeling and solution approach is driven by a global optimization point of view. Two corresponding models are introduced: one of these is a sizeable MILP, while the second one is an even more challenging MINLP. The simplified MILP model is aimed at finding high-quality—yet, most typically, suboptimal—solutions to the original problem. Such solutions can be improved, if necessary, by initializing and solving a related (extended) MINLP model. Further solution refinement could be profitably obtained, e.g., by adopting the *Phi*-function formulation introduced by Stoyan and Romanova [51].

While the literature devoted to sensor and camera networks is fairly extensive, the less frequently used GO-based modeling and solution approach deserves further investigations. Although our approach currently can address (numerically) only relatively small-scale realistic applications, it is quite promising. We think that GO methodology will effectively support the solution of a significant class of similar problems when the globally optimal solution (or at least a near-optimal numerical solution based on truly global scope search) is required.

In future studies, further experimental analysis will be carried out to handle real-world-level problem instances, applying both the MILP and the MINLP models. Extensions to include more complex scenarios that involve, e.g., internal obstacles in the region to observe represent another area of significant practical interest, implying new modeling and computational challenges. We plan to utilize also other solver engine options depending on the model formulation used.

Acknowledgments Thanks are due to Stefano Gliozzi (Business Consulting Services, IBM) for useful discussions related to the subject and for his valuable support of our related experimental analysis and results.

References

1. Abrams, Z., Goel, A., Plotkin, S.: Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, Berkeley, CA, USA, 26–27 April 2004, pp. 424–432 (2004)
2. Altınel, I.K., et al.: Effective coverage in sensor networks: binary integer programming formulations and heuristics. In: Proceedings of the IEEE International Conference on Communications (ICC'06), vol. 9, pp. 4014–4019. Istanbul, Turkey, 11–15 June 2006
3. Akyildiz, I.F., et al.: Wireless sensor networks: a survey. Comput. Netw.: Int. J. Comput. Telecommun. Netw. **38**(4), 393–422 (2002)
4. Alam, S.M., Haas, Z.J.: Coverage and connectivity in three-dimensional networks. In: Proceedings of MobiCom'06, Los Angeles, CA, USA, 23–29 Sept 2006
5. Barbosa, J., et al.: Optimal camera placement for total coverage. In: Proceedings of the 2009 I.E. international conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009, pp. 3672–3676 (2009)
6. Berry, J., et al.: Sensor placement in municipal water networks with temporal integer programming models. J. Water Resour. Plann. Manage. **132**(4), 218–224 (2006)

7. Biswas, P., Ye, Y.: Semidefinite programming for ad hoc wireless sensor network localization. In: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04), pp. 46–54. Berkeley, CA, USA, 26–27 April 2004. (2009)
8. Bomze, I.M., et al. (eds.): Developments in Global Optimization. Kluwer, Dordrecht (1997)
9. Buczak, A.L., et al.: Genetic algorithm convergence study for sensor network optimization. Inf. Sci. Inf. Comput. Sci. **133**(3–4), 267–282 (2001)
10. Cardei M., et al.: Energy-efficient target coverage in wireless sensor networks. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM), vol. 3, pp. 1976–1984. Miami, FL, USA, 13–17 March 2005
11. Cerpa, A., et al.: Habitat monitoring: application driver for wireless communications technology. Comput. Comm. Rev. **31**(2) (2001). Supplement
12. Chakrabarty, K., et al.: Grid coverage for surveillance and target location in distributed sensor networks. IEEE Trans. Comput. **51**(12), 1448–1453 (2002)
13. Chen, F., Jiang, P., Xue, A., et al.: An algorithm of coverage control for wireless sensor networks in 3d underwater surveillance systems. In: Huang, D.S. (ed.) ICIC 2008, LNCS 5226, pp. 1206–1213. Springer, Berlin (2008)
14. Drezner, Z., Suzuki, A.: Covering continuous demand in the plane. J. Oper. Res. Soc. **61**, 878–881 (2010)
15. Dhillon, S.S., Chakrabarty, K., Iyengar, S.S.: Sensor placement for grid coverage under imprecise detections. In: Proceedings of the International Conference on Information Fusion, pp. 1581–1587. Annapolis, MD, USA, 8–11 July 2002
16. Dhillon, S.S., Chakrabarty, K.: Sensor placement for effective coverage and surveillance in distributed sensor networks. In: Proceedings of the IEEE Wireless Communications and Networking Conference, pp 1609–1614. New Orleans, LA, USA, 16–20 March 2003
17. Erdem, U., Sclaroff, S.: Optimal placement of cameras in floor-plans to satisfy task requirements and cost constraints. In: OMNIVIS Workshop, pp. 30–41. Prague, Czech Republic, 16 May 2004
18. Fleishman, S., Cohen-Or, D., Lischinski, D.: Automatic camera placement for image-based modeling. Comput. Graphics Forum **19**(2), 101–110 (2000)
19. Fusco, G., Gupta, H.: Selection and orientation of directional sensors for coverage maximization. In: Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pp. 556–564. Rome, Italy, 22–26 June (2009)
20. Gao, J.F., Shi, Y.J.: A simple coverage-evaluating approach for wireless sensor networks with arbitrary sensing areas. Inform. Process. Lett. **106**(4), 159–161 (2008)
21. Gentile, C.: Distributed sensor location through linear programming with triangle inequality constraints. IEEE Trans. Wireless Comm. **6**(7), 2572–2581 (2007)
22. Harada, J., Shioda, S., Saito, H., Path: Coverage properties of randomly deployed sensors with finite data-transmission ranges. Comput. Netw.: Int. J. Comput. Telecommun. Netw. **53**(7), 1014–1026 (2009)
23. Horst, R., Pardalos, P.M. (eds.): Handbook of Global Optimization, 1st edn. Kluwer, Dordrecht (1995)
24. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Berlin (1996)
25. Hörster, E., Lienhart, R.: On the optimal placement of multiple visual sensors. In: Proceedings of the 4th ACM international Workshop on Video Surveillance and Sensor Networks (VSSN '06), Santa Barbara, California, USA, 27 Oct 2006. ACM, New York (2002)
26. Howard, A., Mataric, M.J., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem. In: Proceedings of the International Symposium on Distributed Autonomous Robotics Systems (DARS02), pp 299–308. Fukuoka, Japan, 25–27 June 2002
27. Huang CF, Tseng YC, Lo LC (2004) The coverage problem in three-dimensional wireless sensor networks. In: Global Telecommunications Conference 2004 (GLOBECOM '04 IEEE), vol. 5, pp. 3182–3186. Dallas, Texas, Nov 29–Dec 3, 2004

28. IBM Corporation: ILOG CPLEX Optimizer. In: High Performance Mathematical Optimization Engines. IBM Corporation Software Group, Route 100 Somers, NY, USA. WSD14044-USEN-01 (2010)
29. Kang, C.W., Golay, M.W.: An integrated method for comprehensive sensor network development in complex power plant systems. Reliability Eng. Syst. Safety **67**, 17–27 (2000)
30. Lazos, L., Poovendran, R.: Stochastic coverage in heterogeneous sensor networks. ACM Trans. Sensor Netw. **2**(3), 325–328 (2006)
31. Li, D., Sun, X.: Nonlinear Integer Programming. Springer, New York (2006)
32. Liberti, L., Maculan, N. (eds.): Global Optimization: From Theory to Implementation. Springer, New York (2005)
33. Meguerdichian, S., et al.: Coverage problems in wireless ad-hoc sensor networks. In: Proceedings of the INFCOM 2001 Conference, pp. 1380–1387. Anchorage, Alaska, USA, 22–26 April 2001
34. Meguerdichian, S., Potkonjak, M.: Low power 0/1 coverage and scheduling techniques in sensor networks. Technical Report 030001, Computer Science Department, University of California Los Angeles (2003)
35. Miorandi, D., Altman, E.: Coverage and connectivity of ad hoc networks in presence of channel randomness. In: Proceedings of IEEE INFOCOM '05, pp. 491–502. Miami, USA, 13–17 March 2005
36. Mittal, A., Davis, L.: Visibility analysis and sensor planning in dynamic environments. In: 8th European Conference on Computer Vision (ECCV 2004), vol. 1, pp. 175–189. Prague, Czech Republic, 11–14 May 2004
37. Mockus, J., et al.: Bayesian Heuristic Approach to Discrete and Global Optimization. Kluwer, Dordrecht (1996)
38. Molina, A., Athanasiadou, G.E., Nix, A.R.: The automatic location of base-stations for optimised cellular coverage: a new combinatorial approach. In: IEEE 49th Vehicular Technology Conference, vol. 1, pp. 606–610. Houston, Texas, USA, 16–20 May 1999
39. Nowak, I.: Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming. Springer, New York (2005)
40. Onur, E., et al.: Coverage in sensor networks when obstacles are present. In: Proceedings of the IEEE International Conference on Communications (ICC '06), vol. 9, pp. 4077–4082. Istanbul, Turkey, 11–15 June 2006
41. Pardalos, P.M., Romeijn, H.E.: Handbook of Global Optimization, 2nd edn. Kluwer, Dordrecht (2002)
42. Pintér, J.D.: Global Optimization in Action. Kluwer, Dordrecht (1996)
43. Pintér, J.D.: Global optimization: software, test problems, and applications. In: Pardalos, P.M., Romeijn, H.E. (eds.) Handbook of Global Optimization, 2nd edn, pp. 515–569. Kluwer, Dordrecht (2002))
44. Pintér, J.D.: Software development for global optimization. In: Pardalos, P.M., Coleman, T.F., (eds.) Global Optimization: Methods and Applications, pp. 183–204. Fields Institute Communications vol. 55. Published by the American Mathematical Society, Providence, RI (2009)
45. Pottie, G.J., Kaiser, W.J.: Wireless integrated network sensors. Comm. ACM **43**, 51–58 (2000)
46. Rowe, E.G., Wettergren, T.A.: Coverage and reliability of randomly distributed sensor systems with heterogeneous detection range. Int. J. Distrib. Sensor Netw. **5**(4), 303–320 (2009)
47. Sahni, S., Xu, X.: Algorithms for wireless sensor networks. Int. J. Distrib. Sensor Netw. **1**(1), 35–56 (2005)
48. Shen, C., Srisathapornphat, C., Jaikaeo, C.: Sensor information networking architecture and applications. IEEE Personal Comm. **8**(4), 52–59 (2001)
49. So, A.M., Ye, Y.: Theory of semidefinite programming for sensor network localization. Math. Program. **109**(2), 367–384 (2007)
50. Sorokin, A., et al.: Mathematical programming techniques for sensor networks. Algorithms **2**, 565–581 (2009)

51. Stoyan, Y.G., Romanova, T.Y.: Mathematical models of placement optimisation: two- and three-dimensional problems and applications. In: Fasano, G., Pintér, J.D. (eds.) Modeling and Optimization in Space Engineering. Springer, New York (2013)
52. Strongin, R.G., Sergeyev, Y.D.: Global Optimization with Non-convex Constraints. Kluwer, Dordrecht (2000)
53. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming. Kluwer, Dordrecht (2002)
54. Venkatesh, S., Buehrer, R.M.: A linear programming approach to NLOS error mitigation in sensor networks. In: Proceedings of the 5th International Conference on Information processing (IPSN '06), pp. 301–308. Nashville, TN, USA, 19–21 April 2006
55. Xing, G., et al.: Integrated coverage and connectivity configuration for energy conservation in sensor networks. ACM Trans. Sensor Netw. **1**(1), 36–72 (2005)
56. Wang, H.B., Yao, K., Erstrin, D.: Information-theoretic approaches for sensor selection and placement in sensor networks for target localization and tracking. J. Commun. Networks **7**(4), 438–449 (2005)
57. Yao, Y., et al.: Can you see me now? Sensor positioning for automated and persistent surveillance. IEEE Trans. Syst. Man Cybernetics, Part B: Cybernetics **40**(1), 101–115 (2010)
58. Zabinsky, Z.B.: Stochastic Adaptive Search for Global Optimization. Kluwer, Dordrecht (2003)
59. Zhang, M., Du, X., Nygard, K.: (2005) Improving coverage performance in sensor networks by using mobile sensors. In: Proceedings of the Military Communications Conference (MILCOM 2005), vol. 5, pp. 3335–3341. Atlantic City, NJ, USA, 17–20 Oct 2005
60. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization in distributed sensor networks. ACM Trans. Embed. Comput. Syst. **3**(1), 61–91 (2004)
61. Zou, Y., Chakrabarty, K.: Uncertainty-aware and coverage-oriented deployment for sensor networks. J. Parallel Distrib. Comput. **64**(7), 788–798 (2004)
62. Zou, Y., Chakrabarty, K.: A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks. IEEE Trans. Comput. **54**(8), 978–991 (2005)

Chapter 11
Space Module On-Board Stowage Optimization by Exploiting Empty Container Volumes

Giorgio Fasano and Maria Chiara Vola

Abstract This chapter discusses a research activity recently carried out by Thales Alenia Space, to support International Space Station (ISS) logistics. We investigate the issue of adding a number of virtual items (i.e. items not given a priori) inside partially loaded containers, in order to exploit the volume still available on board as much as possible. Items already accommodated are supposed to be *tetris*-like, while the additional virtual items are assumed to be parallelepipeds. A mixed-integer non-linear programming (MINLP) model is introduced first, then possible linear (MILP) approximations are discussed, and a corresponding heuristic solution approach is proposed. Guidelines for future research are highlighted, and experimental insights are provided to show the efficiency of the proposed approach.

Keywords Space cargo accommodation • Container loading problem • Non-standard three-dimensional packing • Virtual items • *Tetris*-like items • MINLP models • MILP approximations • Heuristics

11.1 Introduction

This study has been motivated by the challenging issue of cargo accommodation in space vehicles and modules. Specifically, we focus on the on-board stowage problem of the International Space Station (ISS, http://www.nasa.gov) with particular reference to the European Columbus Laboratory (http://www.esa.int). The laboratory also provides logistic support for the ISS (consult [6]).

G. Fasano (✉)
Thales Alenia Space Italia S.p.A., Str. Antica di Collegno 253, 10146 Turin, Italy
e-mail: giorgio.fasano@thalesaleniaspace.com

M.C. Vola
Altran Italia S.p.A., Turin, Italy
e-mail: mariachiara.vola@altran.it

A fleet of vehicles is used for transportation, on the basis of the cargo plan provided by NASA, to control the upload/download flow logistic to/from the ISS. The cargo plan is supposed to be repeatedly updated over time, as the whole logistic process continues. If not all planned upload cargo can be accommodated at a given time, then part of it can be temporary crossed off the list and taken into account in further launches.

To meet (at least approximately) a given cargo plan, a number of non-trivial three-dimensional packing problems arise. Once the optimal packing solution (that offers the best loading of the listed items) at the current step is obtained, the cargo engineer is still asked to execute a further demanding job. How could the residual space (volume) be suitably exploited? More precisely, we assume that it is allowed, for each container not yet fully loaded, to add a certain number of virtual items (that are not known a priori), by reallocating the ones already accommodated, if necessary. What sort of virtual items could be properly added, in order to maximize the loaded volume of each container? A further optimization process is then performed, and possible "hole-filling" (virtual) items are suggested, thereby making the achievement of further cargo planning objectives significantly easier.

In this chapter we discuss the issue of optimizing the volume exploitation of a partially loaded single container by adding up to a maximum number of virtual items, repositioning, if necessary the objects already loaded. From the analytical point of view, this problem is quite similar to the one consisting of creating internal separation elements of filling material (such as foam), to protect the items loaded inside the containers and to prevent their collision especially during the launch phase.

Although our study has originated in order to tackle the on-board stowage of space modules, it can be applied also in many other contexts. Examples include the use of autonomous robots in future space exploration (e.g. to identify accessibility zones and to support loading and assembling activities); another area—not in the space engineering context—is that of the very large system integration (VLSI).

The literature related to packing issues is extensive (consult, e.g. [3]). A major subject area within this broad context is the orthogonal packing of two- and three-dimensional objects inside convex domains (cf., e.g. [1, 7, 12, 13, 15]). A number of mixed-integer (linear) programming (MILP) formulations of two- and three-dimensional orthogonal packing problems are available (see, e.g. [2, 14, 16]). The topic discussed here extends a previous MIP-based approach (see [4]) devoted to the three-dimensional packing of *tetris*-like items within a convex domain, in presence of additional conditions. The term *tetris*-like refers either to parallelepipeds or, more generally, to properly connected clusters of mutually orthogonal parallelepipeds as illustrated by Fig. 11.1.

In the following discussion, we assume that the container is a convex domain. Already loaded items are assumed to be *tetris*-like, while the additional virtual items are parallelepipeds of various sizes. Our objective is to create a number of virtual items (without exceeding a given bound) and to determine their sizes so that the total loaded volume inside the container is maximized.

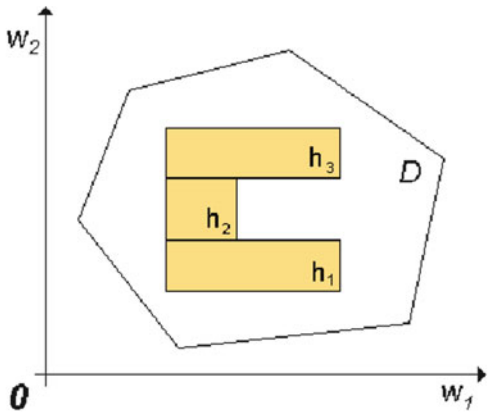


Fig. 11.1 *Tetris*-like item (two-dimensional representation)

The problem is stated in Sect. 11.2. Section 11.3 formulates it in terms of mixed-integer non-linear programming (MINLP) (consult, e.g. [8–11, 17, 18, 22]). A simplified approach, based on MILP approximations, is discussed in Sect. 11.4. A review of some real-world applications is given in Sect. 11.5, including the discussion of a heuristic approach that has been quite efficient in practice.

11.2 Problem Statement

To formulate the problem studied, we shall consider a convex domain D which is a polyhedron (or it can be approximated by a polyhedron). Inside D, we have a set of n *tetris*-like items (TLI), each one consisting of a cluster of mutually orthogonal parallelepipeds, called components, see Fig. 11.1.

We shall also suppose that the TLIs have been positioned within D, in compliance with the following packing rules:

- each TLI has to be positioned parallel to an axis of a pre-fixed orthonormal domain reference frame (orthogonality conditions);
- each TLI has to be contained within D (domain conditions);
- components of different TLIs cannot overlap (non-intersection conditions).

In particular, this means that at least a feasible solution to the orthogonal packing of the given set of TLIs exists. We are then asked to add up to $\overline{N}$ virtual items, consisting of parallelepipeds with sides of variable length (but not less than a given minimal threshold), so that the total volume occupied by the TLIs plus the virtual items is maximized. We are not requested to keep the initial feasible solution for the TLIs, i.e. we are allowed to move them by arbitrary feasible rotations and translations. However, we must take into account the following rules for each virtual item (VI):

- each VI side has to be parallel to an axis of the pre-fixed orthonormal domain reference frame (orthogonality conditions);

- each VI has to be contained within D (domain conditions);
- VIs cannot overlap with the packed TLIs or with other VIs (non-intersection conditions).

11.3 An MINLP Model Formulation

A MINLP formulation of the outlined optimization problem is described next. First of all, let us point out that the packing rules expressed in Sect. 11.2 can be grouped as follows:

- orthogonality, domain and non-intersection conditions for TLIs only;
- orthogonality, domain and non-intersection conditions for VIs only;
- non-intersection conditions between TLIs and VIs.

The orthogonality, domain and non-intersection conditions for TLIs have been discussed in detail in [4]. For the sake of completeness, we will briefly discuss these conditions here. An orthogonal main reference frame with origin O and axes w_β, $\beta \in \{1,2,3\}$ is defined, and a local reference frame is associated to each TLI. Each local reference frame is chosen so that all TLI components lie within its first (positive) octant. Its origin coordinates, with respect to the main reference frame axes, are denoted by $o_{\beta i}$. Next, we introduce the set Ω of all possible orthogonal rotations, admissible for any TLI's local reference frame, with respect to the main one. Under the orthogonality assumptions there are 24 orthogonal rotations since the TLIs are in general asymmetric objects.

For each TLI i, we introduce the set C_i of its (parallelepiped) components. E_{hi} indicates, for each component h of TLI i, the set of the eight vertices associated to it, and an extension of this set is obtained by adding to E_{hi} the geometrical centre of component h. This extended set is denoted by E^*_{hi} and its generic element by η; $\eta = 0$, in particular, represents the geometrical centre. For each TLI i and each possible (orthogonal) orientation $\omega \in \Omega = \{1,\ldots,24\}$, the binary variables $\vartheta_{\omega i} \in \{0,1\}$ are defined, with the meaning: $\vartheta_{\omega i} = 1$ if TLI i has the orthogonal orientation $\omega \in \Omega$, and $\vartheta_{\omega i} = 0$ otherwise.

The orthogonality conditions for TLIs (only) are expressed as follows:

$$\forall i \quad \sum_{\omega} \vartheta_{\omega i} = 1, \tag{11.1}$$

$$\forall \beta, \forall i, \forall h \in C_i, \forall \eta \in E^*_{hi}, \quad w_{\beta\eta hi} = o_{\beta i} + \sum_{\omega} W_{\omega\beta\eta hi}\vartheta_{\omega i}, \tag{11.2}$$

where $w_{\beta\eta hi}$ are the vertex coordinates of component h or its geometrical centre, relative to TLI i, with respect to the reference frame axes w_β; $W_{\omega\beta\eta hi}$ are the coordinate differences between points $\eta \in E^*_{hi}$ and the origin of the local reference frame, with $o_{\beta i}$ coordinates, projected along the w_β axis of the main reference frame, corresponding to the TLI i orientation ω.

Next, the domain conditions for TLIs (only) are expressed as follows:

$$\forall \beta, \forall i, \forall h \in C_i, \forall \eta \in E_{hi} \quad w_{\beta\eta hi} = \sum_{\gamma} V_{\beta\gamma} \lambda_{\gamma\eta hi}, \tag{11.3}$$

$$\forall i, \forall h \in C_i, \forall \eta \in E_{hi} \quad \sum_{\gamma} \lambda_{\gamma\eta hi} = 1, \tag{11.4}$$

where $\lambda_{\gamma\eta hi}$ are non-negative variables; (V_{11},V_{21},V_{31}),..., (V_{1u},V_{2u},V_{3u}) are the vertices of D (whose coordinates, in the main reference frame, are assumed as non-negative, without loss of generality).

The non-intersection conditions (again, for TLIs only) are represented by the constraints below:

$$\begin{gathered}\forall \beta, \forall i, \forall j, i<j, \forall h \in C_i, \forall k \in C_j \\ w_{\beta 0hi} - w_{\beta 0kj} \geq \frac{1}{2} \sum_{\omega} \left(L_{\omega\beta hi}\vartheta_{\omega i} + L_{\omega\beta kj}\vartheta_{\omega j}\right) - D_\beta\left(1 - \sigma^{+}_{\beta hkij}\right),\end{gathered} \tag{11.5.1}$$

$$\begin{gathered}\forall \beta, \forall i, \forall j, i<j, \forall h \in C_i, \forall k \in C_j \\ w_{\beta 0kj} - w_{\beta 0hi} \geq \frac{1}{2} \sum_{\omega} \left(L_{\omega\beta hi}\vartheta_{\omega i} + L_{\omega\beta kj}\vartheta_{\omega j}\right) - D_\beta\left(1 - \sigma^{-}_{\beta hkij}\right),\end{gathered} \tag{11.5.2}$$

$$\forall i, \forall j, i<j, \forall h \in C_i, \forall k \in C_j \quad \sum_{\beta} \left(\sigma^{+}_{\beta hkij} + \sigma^{-}_{\beta hkij}\right) \geq 1, \tag{11.6}$$

where D_β are the sides (parallel to the main reference frame axes) of the parallelepiped of minimum volume that encloses D; $w_{\beta 0hi}$ and $w_{\beta 0hj}$ are the centre coordinates of components h and k; $L_{\omega\beta hi}$ and $L_{\omega\beta kj}$ are their sides, parallel to the w_β axis, corresponding to the orientation ω and $\sigma^{+}_{\beta hkij}$ and $\sigma^{-}_{\beta hkij} \in \{0, 1\}$. If either in (11.5.1) or (11.5.2) a variable σ is set to one, then the corresponding constraint is called a relative position constraint.

Constraints (11.1)–(11.6) represent the necessary and sufficient conditions for placing the given n TLIs in compliance with the packing rules stated above and hold when only TLIs are involved (constraints 11-6 can be expressed in terms of equations, in order to make the model formulation *tighter*). The orthogonality, domain and non-intersection conditions for VIs only, as well as the non-intersection ones involving both TLIs and VIs, are formulated next.

As the number of VIs actually used may be smaller than the maximum allowable $\overline{N}$, the binary variable $\chi_j \in \{0,1\}, j \in \{1,...,\overline{N}\}$, is introduced: χ_j =1 if VI j is used, and $\chi_j = 0$ otherwise.

The orthogonality conditions for VIs are then implicitly stated by the following domain conditions and hold for each vertex η of VI j:

$$\forall \beta, \forall j, \forall \eta \in E_j \quad w_{\beta \eta j} = \sum_{\gamma} V_{\beta \gamma} \lambda_{\gamma \eta j}, \tag{11.7}$$

$$\forall j, \forall \eta \in E_j \quad \sum_{\gamma} \lambda_{\gamma \eta j} = \chi_j, \tag{11.8}$$

where E_j is the set of vertices associated to VI j and $w_{\beta \eta j}$ are their coordinates; both $w_{\beta \eta j}$ and $\lambda_{\gamma \eta j}$ are non-negative variables.

The following inequalities represent the non-intersection conditions between the generic TLI i and VI j:

$$\forall \beta, \forall i, \forall j, \forall h \in C_i$$
$$w_{\beta 0 hi} - w_{\beta 0 j} \geq \frac{1}{2} \sum_{\omega} \left(L_{\omega \beta hi} \vartheta_{\omega i} \right) + \frac{1}{2} l_{\beta j} - D_{\beta} \left(1 - \sigma^{+}_{\beta hij} \right), \tag{11.9.1}$$

$$\forall \beta, \forall i, \forall j, \forall h \in C_i$$
$$w_{\beta 0 j} - w_{\beta 0 hi} \geq \frac{1}{2} \sum_{\omega} \left(L_{\omega \beta hi} \vartheta_{\omega i} \right) + \frac{1}{2} l_{\beta j} - D_{\beta} \left(1 - \sigma^{-}_{\beta hij} \right), \tag{11.9.2}$$

$$\forall i, \forall j, \forall h \in C_i \quad \sum_{\beta} \left(\sigma^{+}_{\beta hij} + \sigma^{-}_{\beta hij} \right) = \chi_j, \tag{11.10}$$

where $w_{\beta 0 j}$ are the centre coordinates of VI j, $l_{\beta j}$ the side parallel to the w_{β} axis, $\sigma^{+}_{\beta hij} \in \{0, 1\}$ and $\sigma^{-}_{\beta hij} \in \{0, 1\}$ are binary variables.

The condition $\sigma^{+}_{\beta hij} = 1$ implies that the side projection of h and j respectively do not overlap along the w_{β} axis and h precedes j, while $\sigma^{+}_{\beta hij} = 0$ makes the corresponding non-intersection constraint redundant. An essentially identical consideration holds for $\sigma^{-}_{\beta hij}$

Analogous constraints are stated when dealing with the non-intersection conditions relative to each pair of VIs:

$$\forall \beta, \forall j, \forall j', j<j' \quad w_{\beta 0 j} - w_{\beta 0 j'} \geq \frac{1}{2} \left(l_{\beta j} + l_{\beta j'} \right) - D_{\beta} \left(1 - \sigma^{+}_{\beta j j'} \right), \tag{11.11.1}$$

$$\forall \beta, \forall j, \forall j', j<j' \quad w_{\beta 0 j'} - w_{\beta 0 j} \geq \frac{1}{2} \left(l_{\beta j} + l_{\beta j'} \right) - D_{\beta} \left(1 - \sigma^{-}_{\beta j j'} \right), \tag{11.11.2}$$

$$\forall j, \forall j', j<j' \quad \sum_{\beta} \left(\sigma^{+}_{\beta j j'} + \sigma^{-}_{\beta j j'} \right) \geq \chi_j + \chi_{j'} - 1. \tag{11.12}$$

The lower bound $\underline{L}$ is further introduced for all VI sides, in order to obtain valid solutions from a practical point of view (since too small VIs would be useless), together with additional upper bounds:

$$\forall j \quad \underline{L}\chi_j \le l_{\beta j} \le D_\beta \chi_j. \tag{11.13}$$

The objective function to maximize is simply defined by the total volume of the VIs added:

$$\max\left\{\sum_j \prod_\beta l_{\beta j}\right\}. \tag{11.14}$$

Remark 1 In [4], a number of additional conditions such as a given minimum gap between items, the pre-fixed position or orientation of some of them, the presence of structural elements, forbidden zones (*holes*) inside the domain and separation planes (movable within given ranges) have been taken into account. Such conditions can be easily added to the model under consideration here; in fact, such conditions are taken into account in some of the tests reported in Sect. 11.5).

Remark 2 As far as the balancing conditions are concerned, the concept expressed in the previous work [4] could also be easily added to the present MINLP model by associating to each (added) VI a constant (average) density r. The previous balancing expressions are rewritten:

$$\forall \gamma, \psi_\gamma * \ge 0, \sum_{\gamma=1}^{r} \psi_\gamma * = m$$

$$\forall \beta \quad \sum_{i=1}^{n} M_i w_{\beta i} + \sum_{j=1}^{N} \left(r w_{\beta j} \prod_\beta l_{\beta j} \right) = \sum_{\gamma=1}^{r} V_{\beta\gamma} \psi_\gamma *$$

where

$$m = \sum_{i=1}^{n} M_i + \sum_{j=1}^{N} \left(r \prod_\beta l_{\beta j} \right).$$

Differently from the case discussed in [4], the above conditions are not linear.

Remark 3 The presence of the binary variable χ_j in the expressions (11.13) guarantees that if a certain *virtual* item is not added, its contribution to the total volume computation is zero.

11.4 MILP Model Approximations

The MINLP model formulated in Sect. 11.3 is (most likely) hard to solve to optimality. A first simple linear approximation is generated easily by adopting the surrogate objective function below:

$$\max\left\{\sum_{\beta,j} l_{\beta j}\right\}. \tag{11.15}$$

An alternative approach consists of substituting the original objective function (11.14) by a separable one. This can be easily achieved by adopting the (equivalent) objective function:

$$\max\left\{\sum_{j,\beta} \ln l_{\beta j}\right\}. \tag{11.16}$$

A linear piece-wise approximation of each single-variable term then reduces the original MINLP model to a much simpler (approximate) MILP (cf., e.g. [20]). A straightforward formulation to obtain this can be achieved as follows (consult [21]). For each β, we introduce an arbitrary partition $\{A_{\beta 0},\ldots,A_{\beta\alpha},\ldots\}$ of $[\underline{L}, D_\beta]$ and then state the conditions

$$\forall\beta,\forall j \quad l_{\beta j} = \sum_{\alpha} \lambda_{\beta\alpha j} A_{\beta\alpha}, \tag{11.17}$$

$$\forall\beta,\forall j \quad \ln l_{\beta j} \approx \sum_{\alpha} \lambda_{\beta\alpha j} \ln A_{\beta\alpha}, \tag{11.18}$$

$$\forall\beta,\forall j \quad \sum_{\alpha} \lambda_{\beta\alpha j} = \chi_j, \tag{11.19}$$

where, for each β and j, the $\lambda_{\alpha j}$ variables are subject to the further SOS2 (special ordered set of type 2, [21]) restriction: at most two consecutive $\lambda_{\beta\alpha j}$ can be non-zero. The objective function then becomes

$$\max\left\{\sum_{\alpha,\beta,j} \lambda_{\beta\alpha j} \ln A_{\beta\alpha}\right\}. \tag{11.20}$$

Remark 4 The *SOS2* restriction shown above can be tackled either algorithmically or by introducing additional binary variables and proper constraints in the model [19, 21]. If could also be seen that such restriction in our specific case could be dropped tout court, but this is beyond the scope of the chapter.

Remark 5 A further possibility of converting a sum of products into a separable function could be considered [21]. In the case of the product of two variables $q_1 q_2$ it is sufficient to introduce new variables s_1,s_2, by performing the transformation

$$s_1 = \frac{1}{2}(q_1 + q_2), \quad s_2 = \frac{1}{2}(q_1 - q_2),$$

and substituting $q_1 q_2$ with ${s_1}^2 - {s_2}^2$. The method can be easily extended when the products involve more than two variables and a linear piece-wise approximation of the quadratic terms obtained can be achieved, as in the case of the logarithmic functions discussed above.

Remark 6 The MILP approximations described in this section could well be considered to provide a good starting solution to an MINLP solution process.

11.5 Practical Applications

On the basis of the MILP approximation relative to the surrogate objective function 11.15 mentioned in Sect. 11.4, a heuristic approach has been investigated to obtain quick (satisfactory, but typically suboptimal) solutions to the original problem: this approach will be reviewed next.

An appropriate lower bound, as stated by conditions (11.13), is generated each time, depending on the specific model instance analysed. In addition to this point, it is recommended to get rid of solutions with strongly asymmetric, "non-cube-shaped" VIs. (The surrogate objective function tends to give rise to such VIs, as matter of fact.) Figure 11.2 illustrates an example with just three VIs. In the image on the left no additional limitation was posed, unlike in the case shown on the right.

The heuristic approach proposed here poses further conditions on the possibility of reallocating the already loaded TLIs. While orthogonal rotations of the TLIs are allowed (except for the pre-oriented ones, if any), their translations are partially restricted on the basis of the relative positions they already have (refer to constraints (11.5.1) and (11.5.2)). This restriction is linked to the concept of *abstract configuration* [4, 5]: given a number of couples of components (belonging to different TLIs) an *abstract configuration* consists of a set of variables σ set to one (one and only one for each couple), so that, in any unbounded domain, the subspace delimited by the corresponding *non-intersection* constraints is a feasible region (see Fig. 11.3).

In the heuristic approach proposed here, as a first step, an *abstract configuration* compliant with the solution of the already loaded TLIs is selected. This can be done as suggested in [5]. The *abstract configuration* is then imposed to the MILP model, so that only the relative non-intersection constraints are considered, while all the remaining ones are neglected. This gives rise to a reduced MILP model that dramatically decreases the computational effort of the solution process.

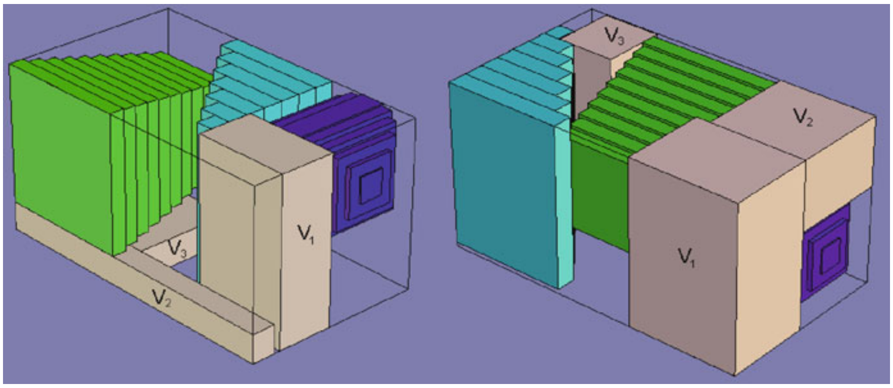


Fig. 11.2 Example of solutions without/with extra limitations: compare the left/right side solutions shown

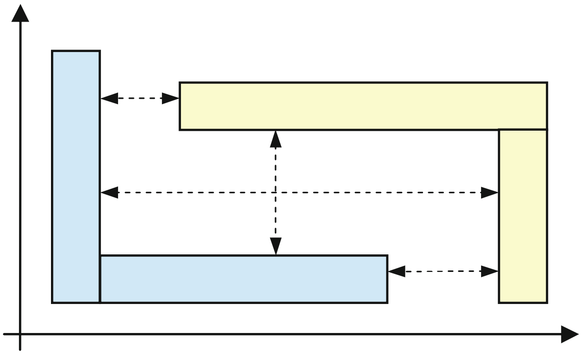


Fig. 11.3 Example of an *abstract configuration*: two-dimensional representation

Since the addition of several VIs all together would represent a significant computational job, the heuristic procedure proceeds, step by step, adding one VI after the other, until a satisfactory solution is obtained or their maximum number is reached. The heuristic proposed is based on the algorithm outlined here below.

```
Algorithm. Incremental introduction of virtual items
Input $n$ tetris-like items, convex domain
Output tetris-like items repositioning, positions and dimensions of the added virtual items ($<\overline{N}$)
Begin
Generate an abstract configuration corresponding to the given tetris-like item accommodation;
    $j \leftarrow 0$; //numer of virtual items
```

```
      bsort ← false;
      free_volume ← container_volume-loaded_Item_volume;
      **While** ( $j < \overline{N}$ & free_volume>$\underline{L}^3$ & !bsort) **do**
      Min_vrt_dim ← (√ free_volume) /2;
      Sol. ← false;
      **While** (MinVrtDim >$\underline{L}$ & !Sol) **do**
        Execute MIP model
        (corresponding to the chosen *abstract configuration*);
           **If** integer solution **then**
              Sol ← true;
              $j \leftarrow j + 1$;
              free_volume ← free_volume - virtual_item_volume;
              Generate new *abstract configuration*;
           **Else**
              **If** min_vrt_dim >$\underline{L}$ **then**
                 min_vrt_dim ← max (min_vrt_dim /2, $\underline{L}$);
              **Else**
                 bsort ← true;
              **End If**
           **End If**
        **End** //close loop while
     **End** //close loop while
   **Return** last item allocation;
   **End**
```

A currently ongoing experimental analysis has been carried out by using the IBM ILOG CPLEX Optimizer 12.3 on a personal computer with a Core 2 Duo P8600, 2.40 GHz processor and 1.93 GB of RAM, running under MS Windows XP Professional with Service Pack 2. Table 11.1 shows some of our experimental results. In the tests reported here a maximum threshold of ten VIs is imposed. The minimum dimension for each of them is assumed to be at least two units. A runtime limit of three CPU hours has been used. Tables 11.2, 11.3, 11.4, 11.5, 11.6, 11.7, 11.8, 11.9, 11.10, 11.11 report the input and output information for case studies 5, 29 and 33, respectively. Figures 11.4, 11.5, 11.6 represent the corresponding graphical solutions. Next, we report some details concerning case studies 5, 29 and 33.

In case study 5, the domain is a parallelepiped of dimensions 16, 18 and 10 units, containing seven TLIs, one of which consists of a single component. Figure 11.4 shows the ten VIs added.

The domain relative to case study 29 is a parallelepiped of dimensions 42, 49 and 74.4 units, containing four structural elements (i.e. TLIs with pre-fixed position and orientation) and a separation plane (grey coloured). Five TLIs (one of them consisting of a single component) are present. In the solution obtained (see Fig. 11.5), ten VIs have been introduced (in compliance to the given relative gap of 1.5 units).

Table 11.1 Experimental analysis

Test ID	Total number of *tetris*-like items	Total number of comp.	Items GAP	Already loaded item volume %	Total number of virtual items	Virtual item volume %	Residual free volume %	CPU time (H:MM:SS)
Case study 1	6	11	0.0	76.78	5	18.44	4.78	0:01:32
Case study 2	6	9	0.0	45.54	10	30.16	24.30	0:04:16
Case study 3	8	13	0.0	69.64	10	23.25	5.33	0:03:08
Case study 4	4	34	0.0	36.87	10	45.62	17.51	0:05:19
Case study 5	7	15	0.0	56.94	10	27.72	15.33	0:04:07
Case study 6	6	16	0.0	49.78	10	33.16	17.06	0:04:41
Case study 7	5	37	0.0	29.00	10	49.28	21.72	0:04:02
Case study 8	5	44	0.0	29.95	10	57.08	12.97	0:03:23
Case study 9	4	32	0.0	51.40	10	22.39	26.21	0:05:23
Case study 10	5	19	0.0	39.21	10	33.81	26.98	0:04:02
Case study 11	7	12	2.5	63.70	1	0.35	35.94	0:00:24
Case study 12	4	29	0.0	44.10	10	38.09	17.81	0:03:53
Case study 13	6	12	2.5	29.66	10	38.84	31.50	0:03:33
Case study 14	5	33	1.5	49.98	10	12.68	37.34	0:03:18
Case study 15	3	28	1.5	47.55	10	26.75	25.70	0:03:09
Case study 16	5	23	2.5	48.15	10	12.49	39.16	0:03:26

Case study 17	3	27	1.0	33.42	10	25.90	40.68	0:38:20
Case study 18	4	27	0.5	33.50	10	35.58	30.92	0:04:21
Case study 19	3	30	2.5	33.79	10	17.29	48.92	0:06:18
Case study 20	9	49	1.0	46.66	10	39.16	14.17	0:03:32
Case study 21	8	31	0.0	56.72	10	24.93	18.35	0:05:13
Case study 22	5	33	0.0	43.75	10	34.02	22.23	0:03:59
Case study 23	5	63	0.0	42.09	10	27.82	30.08	0:03:45
Case study 24	5	51	0.0	45.27	10	32.31	22.41	0:03:34
Case study 25	6	47	0.0	40.32	10	32.51	27.17	0:03:37
Case study 26	4	21	1.5	46.49	8	15.13	38.38	0:04:44
Case study 27	6	27	2.5	41.09	10	21.96	36.95	0:05:39
Case study 28	5	26	1.5	49.85	10	9.98	40.18	0:04:11
Case study 29	5	22	1.5	42.02	10	17.07	40.91	0:03:47
Case study 30	5	25	2.5	42.88	10	26.75	30.37	0:03:39
Case study 31	22	54	0.0	54.37	10	23.94	21.69	0:05:51
Case study 32	3	15	0.0	47.93	10	35.90	20.03	0:04:18
Case study 33	5	26	0.5	49.85	10	19.09	31.07	0:04:43

Table 11.2 Case study 5: input

		Component projection			Component centre coordinates (with respect to the local reference frame)		
Items	Components	X	Y	Z	x	y	z
It 1 (H-shaped)	C1	2	2.5	8	1	5	4
	C2	2	2.5	8	7	5	4
	C3	4	2.5	2	4	5	4
It 2 (H-shaped)	C1	2	2.5	8	1	5	4
	C2	2	2.5	8	7	5	4
	C3	4	2.5	2	4	5	4
It 3 (T-shaped)	C1	3	2.5	6	1.5	5	3
	C2	8	2.5	3	7	5	3
It 4 (T-shaped)	C1	3	2.5	6	1.5	5	3
	C2	8	2.5	3	7	5	3
It 5 (T-shaped)	C1	3	2.5	6	1.5	5	3
	C2	8	2.5	3	7	5	3
It 6 (T-shaped)	C1	3	2.5	6	1.5	5	3
	C2	8	2.5	3	7	5	3
It 7 (single comp.)	–	5	8	10			

Table 11.3 Case study 5: *tetris*-like item output

	Local reference frame origin coordinates (with respect to the main reference frame)			Local reference frame orientation (with respect to the main reference frame)		
Items	X	Y	Z			
It 1 (H-shaped)	7.98	9.98	10	cos(Xx) = 1	cos(Yz) = 1	cos(Zy) = −1
It 2 (H-shaped)	15.98	1.98	8	cos(Xy) = −1	cos(Yz) = 1	cos(Zx) = −1
It 3 (T-shaped)	15.98	17.96	5	cos(Xx) = −1	cos(Yz) = −1	cos(Zy) = −1
It 4 (T-shaped)	0	17.98	10	cos(Xz) = 1	cos(Yx) = −1	cos(Zy) = −1
It 5 (T-shaped)	9	0	10	cos(Xz) = −1	cos(Yx) = 1	cos(Zy) = −1
It 6 (T-shaped)	0	14.48	0	cos(Xx) = 1	cos(Yz) = −1	cos(Zy) = 1
It 7 (single comp.)	0	0	0	cos(Xz) = 1	cos(Yx) = 1	cos(Zy) = 1

Table 11.4 Case study 5: virtual item output

	Item oriented dimension			Item centre coordinates (with respect to the main reference frame)		
Virtual Items	X	Y	Z	X	Y	Z
Vrt_it 1	4.98	3.48	6.98	9.24	6.49	7.51
Vrt_it 2	4.98	3.98	4.98	13.51	2.49	3.99
Vrt_it 3	4.98	3.48	4.98	2.49	16.26	2.49
Vrt_it 4	4.48	3.98	4.98	2.24	4.99	7.51
Vrt_it 5	3.98	3.48	4.98	6.24	12.99	7.51
Vrt_it 6	4.98	2.98	4.98	13.51	9.47	3.49
Vrt_it 7	3.98	2.98	4.98	11.99	16.51	7.51
Vrt_it 8	3.98	2.98	4.98	11.99	11.49	7.51
Vrt_it 9	2.98	2.98	4.98	1.49	1.49	7.51
Vrt_it 10	4.98	1.98	9.98	13.51	4.99	9.01

Table 11.5 Case study 29: input

		Component projection			Component centre coordinates (with respect to the local reference frame)		
Items	Components	X	Y	Z	x	y	Z
It 1 (camera-shaped)	C11	20	12	10	15	8	7
	C21	2	2	3	22	8	13.5
	C31	2	2	3	15	8	13.5
	C41	20	5	5	19	8	17.5
	C51	5	16	14	2.5	8	7
It 2 (cone-shaped)	C12	42	42	7	21	21	3.5
	C22	33.6	33.6	7	21	21	10.5
	C32	25.2	25.2	7	21	21	17.5
	C42	16.8	16.8	7	21	21	24.5
	C52	8.4	8.4	7	21	21	31.5
It 3 (sphere-shaped)	C13	15	15	2.01	15	15	1.00
	C23	25.98	25.98	5.49	15	15	4.75
	C33	30	30	15	15	15	15
	C43	25.98	25.98	5.49	15	15	25.25
	C53	15	15	2.01	15	15	29.00
It 4 (L-shaped)	C14	26	20	8	13	10	4
	C24	8	20	26	4	10	21
It 5 (structural element)	–	3	3	74.4			
It 6 (structural element)	–	3	3	74.4			
It 7 (structural element)	–	3	3	74.4			
It 8 (structural element)	–	3	3	74.4			
It 9 (single comp.)	–	5	7	9			
Separation plane		42	49	2			

Table 11.6 Case study 29: *tetris*-like item and separation plane output

	Local reference frame origin coordinates (with respect to the main reference frame)			Local reference frame orientation (with respect to the main reference frame)		
Items	X	Y	Z			
It 1 (camera-shaped)	26	29	0	cos(Xy) = 1	cos(Yz) = 1	cos(Zx) = 1
It 2 (cone-shaped)	4.5	0.06	0	cos(Xz) = 1	cos(Yx) = 1	cos(Zy) = 1
It 3 (sphere-shaped)	12	14.5	44.4	cos(Xx) = 1	cos(Yy) = 1	cos(Zz) = 1
It 4 (L-shaped)	0.02	4.52	54.4	cos(Xz) = 1	cos(Yx) = 1	cos(Zy) = 1
It 5 (structural element)	0	3	0	cos(Xy) = 1	cos(Yx) = −1	cos(Zz) = 1
It 6 (structural element)	39	3	0	cos(Xy) = 1	cos(Yx) = −1	cos(Zz) = 1
It 7 (structural element)	0	49	0	cos(Xy) = 1	cos(Yx) = −1	cos(Zz) = 1
It 8 (structural element)	39	49	0	cos(Xy) = 1	cos(Yx) = −1	cos(Zz) = 1
It 9 (single comp.)	35	21.66	37.38	cos(Xx) = 1	cos(Yz) = −1	cos(Zy) = 1
Separation plane	21	24.5	43.38	cos(Xx) = 1	cos(Yy) = 1	cos(Zz) = 1

Table 11.7 Case study 29: virtual item output

	Items oriented dimension			Item centre coordinates (with respect to the main reference frame)		
Virtual Items	X	Y	Z	X	Y	Z
Vrt_it 1	10.48	11.16	42.38	32.26	5.58	21.19
Vrt_it 2	11.48	9.62	42.38	18.74	44.19	21.19
Vrt_it 3	10.48	12.48	30	5.24	38.26	59.4
Vrt_it 4	10.48	25.82	11.48	32.26	36.09	36.64
Vrt_it 5	14.98	14.82	11.08	34.51	20.07	5.54
Vrt_it 6	41.98	8.5	8.48	21.01	8.75	48.62
Vrt_it 7	10.48	16.02	8.5	5.24	22.51	48.65
Vrt_it 8	13.48	9	7.28	26.74	17.16	38.74
Vrt_it 9	6.48	8.48	20.02	38.76	8.74	64.39
Vrt_it 10	6.98	9	9.16	38.51	17.16	31.3

Table 11.8 Case study 33: convex domain vertex coordinates

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
X	0	0	0	0	0	0	0	0	0
Y	1.3	31.1	32.2	28	14.4	13.6	1.3	0	0
Z	0	0	1.7	13.4	25.2	25.5	25.5	24.3	1.3
	V_{10}	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}	V_{17}	V_{18}
X	42.9	42.9	42.9	42.9	42.9	42.9	42.9	42.9	42.9
Y	1.3	31.1	32.2	28	14.4	13.6	1.3	0	0
Z	0	0	1.7	13.4	25.2	25.5	25.5	24.3	1.3

The convex domain vertex coordinates (with respect to the main reference frame), corresponding to case study 33, are shown below (Table 11.8):

Inside this container there are two structural elements and a separation plane. Six TLIs (two of which consist of single parallelepipeds) were already present. In the solution obtained (see Fig. 11.6), ten VIs have been accommodated, in compliance to the given relative gap of 0.5 units.

Table 11.9 Case study 33: input

		Component projection			Component centre coordinates (with respect to the local reference frame)		
Items	Components	X	Y	Z	x	y	z
It 1 (sphere-shaped)	C11	6.50	6.50	0.87	6.50	6.50	0.44
	C21	11.26	11.26	2.38	6.50	6.50	2.06
	C31	13.00	13.00	6.50	6.50	6.50	6.50
	C41	11.26	11.26	2.38	6.50	6.50	10.94
	C51	6.50	6.50	0.87	6.50	6.50	12.56
It 2 (wedge-shaped)	C12	13.00	28.00	2.25	6.50	14.00	1.13
	C22	13.00	24.50	2.25	6.50	12.25	3.38
	C32	13.00	21.00	2.25	6.50	10.50	5.63
	C42	13.00	17.50	2.25	6.50	8.75	7.88
	C52	13.00	14.00	2.25	6.50	7.00	10.13
	C62	13.00	10.50	2.25	6.50	5.25	12.38
	C72	13.00	7.00	2.25	6.50	3.50	14.63
	C82	13.00	3.50	2.25	6.50	1.75	16.88
It 3 (cylinder-shaped)	C13	5.00	42.00	0.67	5.00	21.00	0.33
	C23	8.66	42.00	1.83	5.00	21.00	1.58
	C33	10.00	42.00	5.00	5.00	21.00	5.00
	C43	8.66	42.00	1.83	5.00	21.00	8.42
	C53	5.00	42.00	0.67	5.00	21.00	9.67
It 4 (cylinder-shaped)	C14	5.00	42.00	0.67	5.00	21.00	0.33
	C24	8.66	42.00	1.83	5.00	21.00	1.58
	C34	10.00	42.00	5.00	5.00	21.00	5.00
	C44	8.66	42.00	1.83	5.00	21.00	8.42
	C54	5.00	42.00	0.67	5.00	21.00	9.67
It 6 (single comp.)	–	4.00	6.00	8.00			
It 7 (single comp.)	–	4.00	6.00	8.00			
It 8 (structural element)	–	4.00	4.00	25.50			
It 9 (structural element)	–	4.00	4.00	25.50			
Separation plane	–	42.90	32.20	1.30			

Table 11.10 Case study 33: *tetris*-like item/separation plane output

	Local reference frame origin coordinates (with respect to the main local reference frame)			Local reference frame orientation (with respect to the main reference frame)		
Items	X	Y	Z			
It 1 (sphere-shaped)	0	0.04	0	cos(Xx) = 1	cos(Yy) = 1	cos(Zz) = 1
It 2 (cuneo)	42.9	0.13	14.17	cos(Xy) = −1	cos(Yz) = 1	cos(Zx) = −1
It 3 (cylinder-shaped)	0.9	30.12	0	cos(Xy) = 1	cos(Yx) = −1	cos(Zz) = 1
It 4 (cylinder-shaped)	42.9	0	25.47	cos(Xy) = −1	cos(Yz) = 1	cos(Zx) = −1
It 6 (single comp.)	31.1	10.5	15.47	cos(Xy) = 1	cos(Yz) = 1	cos(Zx) = 1
It 7 (single comp.)	16.80	10.50	15.47	cos(Xy) = 1	cos(Yz) = 1	cos(Zx) = 1
It 8 (structural element)	12.3	14.1	0	cos(Xx) = 1	cos(Yy) = 1	cos(Zz) = 1
It 9 (structural element)	26.6	14.1	0	cos(Xx) = 1	cos(Yy) = 1	cos(Zz) = 1
Separation plane	21.45	16.1	14.82	cos(Xx) = 1	cos(Yy) = 1	cos(Zz) = 1

Table 11.11 Case study 33: virtual item output

	Items oriented dimension			Item centre coordinates (with respect to the main reference frame)		
Virtual items	X	Y	Z	X	Y	Z
Vrt_it 1	8.08	12.24	14.16	20.86	13.50	7.09
Vrt_it 2	11.78	6.08	14.16	5.89	16.58	7.08
Vrt_it 3	11.78	8.04	6.08	5.89	14.52	18.51
Vrt_it 4	7.28	8.04	6.08	39.26	14.52	18.51
Vrt_it 5	4.28	5.48	14.16	33.24	16.87	7.08
Vrt_it 6	4.78	10.4	4.02	23.71	15.7	17.48
Vrt_it 7	4.78	5.2	4.02	23.71	13.1	22.00
Vrt_it 8	42.88	7.14	3.52	21.46	23.69	12.26
Vrt_it 9	4.38	4	14.16	15.71	4.88	7.08
Vrt_it 10	3.66	6.22	3.92	14.49	10.49	12.21

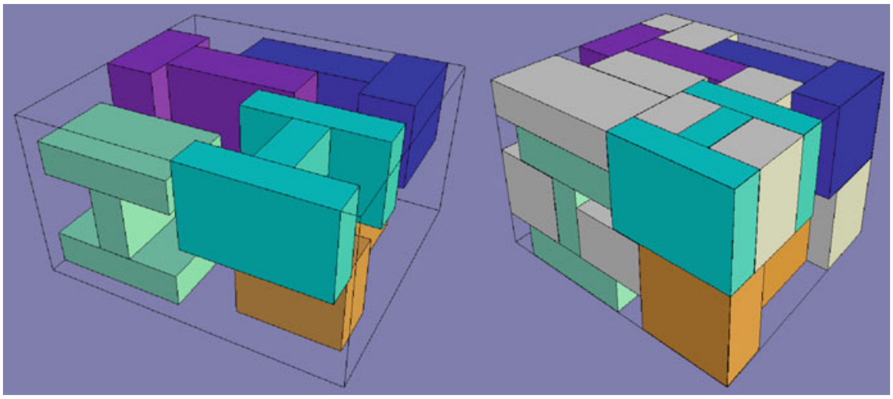


Fig. 11.4 Case study 5: solution shown with the added virtual items (see right-hand side image)

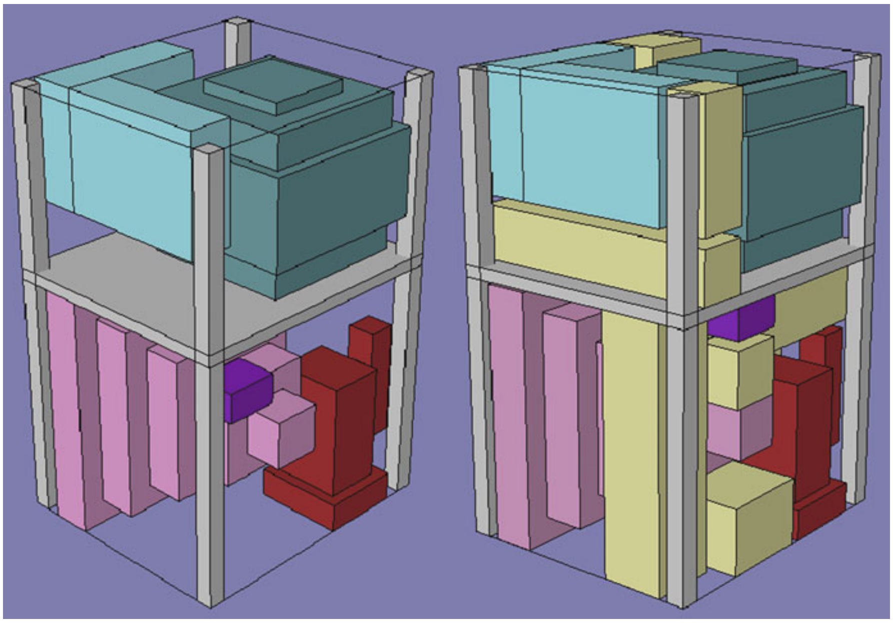


Fig. 11.5 Case study 29: solution shown with the added virtual items (see right-hand side image)

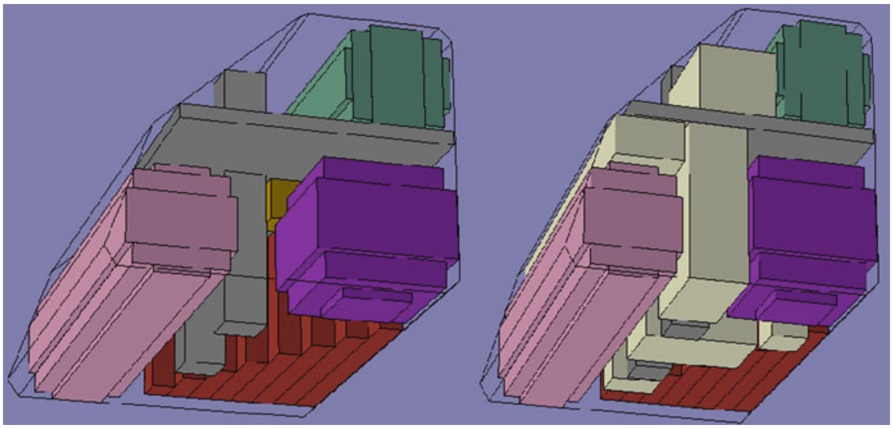


Fig. 11.6 Case study 33: solution shown with the added virtual items (see right-hand side image)

11.6 Concluding Remarks

In this study we consider the issue of maximizing the volume exploitation of a partially loaded container, by adding virtual items. The items already inside the container are supposed to be *tetris*-like, placed orthogonally with respect to a given reference frame. By assumption, the container is a convex domain, and the virtual items are parallelepipeds of various dimensions (not smaller than a given minimum size). *Tetris*-like items can be repositioned, and the number of virtual items that may be added cannot exceed a given threshold. The problem discussed here originates from the field of space engineering, but similar important applications arise in a number of different contexts.

To handle this problem type, an MINLP formulation has been introduced. Next, possible MILP approximations have been taken into account, and an MIP-based heuristic approach aimed at finding quick satisfactory solutions in practice has been described.

In future research, an in-depth experimental analysis can be carried out to compare the heuristic procedure proposed with the separable programming-based MILP approximation approach. The MILP solution can also been used to initialize the MINLP model based optimization process.

References

1. Birgin, E., Martinez, J.M., Nishihara, F.H., Ronconi, D.P.: Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization. Comput. Oper. Res. **33**(12), 3535–3548 (2006)
2. Chen, C.S., Lee, S.M., Shen, Q.S.: An analytical model for the container loading problem. Eur. J. Oper. Res. **80**, 68–76 (1995)
3. Dyckhoff, H., Scheithauer, G., Terno, J.: Cutting and packing. In: Dell'Amico, M., Maffioli, F., Martello, S. (eds.) Annotated Bibliographies in Combinatorial Optimization. Wiley, Chichester (1997)
4. Fasano, G.: MIP-based heuristic for non-standard 3D-packing problems. 4OR **6**, 291–310 (2008)
5. Fasano, G.: A global optimization point of view to handle non-standard object packing problems. J. Global Optim. (2012) DOI: 10.1007/s10898-012-9865-8
6. Fasano, G., Saia, D., Piras, A.M.: Columbus stowage optimization by CAST (cargo accommodation support tool). Acta Astronautica **67**(3), 489–496 (2010)
7. Fekete, S., Schepers, J.: A combinatorial characterization of higher-dimensional orthogonal packing. Math. Oper. Res. **29**, 353–368 (2004)
8. Floudas, C.A., Pardalos, P.M. (eds.): Encyclopedia of Optimization. Kluwer, Dordrecht (2001)
9. Kallrath, J.: Mixed-integer nonlinear applications. In: Ciriani, T., Gliozzi, S., Johnson E.L., (eds.) Operations Research in Industry, pp. 42–76. McMillan Press, Great Britain (1999)
10. Kallrath, J.: Modeling difficult optimization problems. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, 2nd edn, pp. 2284–2297. Springer, New York (2008)
11. Liberti, L., Maculan, N. (eds.): Global Optimization: From Theory to Implementation. Springer, New York (2005)

12. Martello, S., Pisinger, D., Vigo, D.: The three-dimensional bin packing problem. Oper. Res. **48**, 256–267 (2000)
13. Martello, S., Pisinger, D., Vigo, D., den Boef, E., Korst, J.: Algorithms for general and robot-packable variants of the three-dimensional bin packing problem. ACM Trans. Math. Software **33**(1) (2007). article 7
14. Padberg, M.: Packing small boxes into a big box. Office of Naval Research, N00014-327, New York University, New York (1999)
15. Pisinger, D., Sigurd, M.M.: The two-dimensional bin packing problem with variable bin sizes and costs. Discrete Optim. **2**, 154–167 (2005)
16. Pisinger, D., Sigurd, M.M.: Using decomposition techniques and constraints programming for solving the two-dimensional bin packing problem. INFORMS J. Comput. **19**(1), 36–51 (2006). Winter 2007
17. Rebennack, S., Kallrath, J., Pardalos, P.M.: Column enumeration based decomposition techniques for a class of non-convex MINLP problems. J. Global Optim. **43**(2–3), 277–297 (2009)
18. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming. Kluwer, Dordrecht (2002)
19. Vielma, J.P., Nemhauser, G.L.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Math. Program. **128**(1–2), 49–72 (2009)
20. Wang, P.C., Tsai, J.F.: A superior piecewise linearization approach for assortment problems. In: EURO XXIV Conference, Lisbon, 11–14 July 2010
21. Williams, H.P.: Model Building in Mathematical Programming. Wiley, London (1993)
22. Zhu, Y., Kuno, T.: Global optimization of non-convex minlp by a hybrid branch-and-bound and revised general Benders decomposition approach. Ind. Eng. Chem. Res. **42**(3), 528–539 (2003)

Chapter 12
Optimization Models for the Three-Dimensional Container Loading Problem with Practical Constraints

Leonardo Junqueira, Reinaldo Morabito, Denise Sato Yamashita, and Horacio Hideki Yanasse

Abstract The last decades have seen an increasing emergence of solution approaches to three-dimensional container loading problems. Starting from simple constructive algorithms and passing through sophisticated metaheuristics, the container loading literature offers a range of solving options. However, few authors have engaged themselves in proposing optimization models to deal with container loading problems that aim to pack the largest volume (or value) of rectangular boxes orthogonally inside a single container. In this sense, studies are even scarcer when practical constraints are considered. Cargo stability, load bearing strength of the boxes, and multi-drop situations, among others, are constraints that have important practical claim and should be considered in order to model more realistic situations. In this chapter we are concerned with presenting mixed integer linear programming models for container loading problems that consider vertical and horizontal stability of the cargo, load bearing strength of the boxes, and multi-drop situations, besides the non-overlapping of boxes. Computational results achieved with a modeling language and an optimization solver, comparing the performance of the models on instances from the literature, are also presented and discussed. Finally, we discuss some potential directions for future works in this area.

Keywords Three-dimensional container loading • Mathematical modeling • Cargo stability • Load bearing • Multi-dropping

L. Junqueira • R. Morabito (✉) • D.S. Yamashita
Departamento de Engenharia de Produção, Universidade Federal de São Carlos, Rodovia Washington Luís, km 235−SP-310, 13565-905 São Carlos−São Paulo, Brazil
e-mail: leo_junqueira@yahoo.com; morabito@ufscar.br; denisesy@dep.ufscar.br

H.H. Yanasse
Instituto Nacional de Pesquisas Espaciais−INPE/LAC, Avenida dos Astronautas 1758, Jardim da Granja, 12227-010 São José dos Campos−São Paulo, Brazil
e-mail: horacio@lac.inpe.br

12.1 Introduction

The cutting and packing literature provides a variety of names for the problem we address in this chapter. Besides *container loading problem*, other common names that indicate the same problem are: *container packing problem*, *three-dimensional cargo-loading problem*, *three-dimensional packing problem*, *three-dimensional knapsack packing problem*, *single container loading problem*, *three-dimensional pallet loading problem*, *three-dimensional palletization problem*, *three-dimensional cutting problem*, etc. The problem, in its basic form, consists of finding the best three-dimensional packing pattern for loading a set of boxes into a container so that the total volume (or value) of the boxes loaded is maximized and the boxes do not overlap (e.g., [16, 17, 20, 27, 29, 30, 34]). It is assumed that the boxes and the container are of rectangular shape and that the boxes can only be placed orthogonally inside the container. The problem considered can be broadly characterized as being of type 3/B/O/-, according to the typology presented in Dyckhoff [11], or as being of a type 3D-R-IIPP/SLOPP/SKP, according to the typology presented in Wäscher et al. [35].

Besides the non-overlapping of boxes, 12 practical considerations that might play an important role when addressing more real-world container loading problems were presented in Bischoff and Ratcliff [4]. This list of considerations includes: cargo stability, load bearing strength of the boxes, multi-dropping, box orientation, box handling, box grouping, box separation, complete shipment of box groups, box priorities, complexity of the cargo arrangement, container weight limit, and weight distribution within the container. These assumptions have motivated the study and development of various heuristics and metaheuristics in the literature to solve the container loading problem (e.g., [6, 15, 18, 21, 26, 28, 36]). However, not many works have presented mathematical models for this problem with additional constraints. Some papers, such as Beasley [1], Hadjiconstantinou and Christofides [19], and Beasley [2], present formulation for two-dimensional cutting and packing problems that can be easily extended to the container loading problem. Other formulation for the container loading problem are presented in Tsai et al. [33] and Chen et al. [8]. Nevertheless, none of these papers handle the practical considerations mentioned, limiting only to avoid that the boxes do not overlap inside the container.

In this chapter, we present mixed integer linear programming models for three-dimensional container loading problems that consider, in addition to the non-overlapping of boxes, other constraints like the vertical and horizontal stabilities of the cargo, the load bearing strength of the boxes, and the multi-dropping. Cargo stability (e.g., [12, 22, 27, 28, 30, 32]) refers to the support of the bottom faces of boxes, in the case of vertical stability (i.e., the boxes must have their bottom faces supported by other box top faces or the container floor), and the support of the lateral faces of boxes, in the case of horizontal stability. Load bearing strength (e.g., [3, 5, 22, 31]) refers to the maximum number of boxes that can be stacked one above each other or, more generally, to the maximum pressure that can be applied over the top face of a box. Multi-dropping (e.g., [9, 23, 24]) refers to cases where boxes that are delivered to the same customer (destination) must be placed close to each other inside the container (or truck, as it is more common in practice), and the loading pattern must take into account the delivery route of the vehicle and the sequence in which the boxes are unloaded.

The practical importance of incorporating these constraints to the problem is to avoid loading patterns where boxes are “floating in mid air” inside the container, where products are damaged due to deformation of the boxes that contain them, or where an unnecessary additional handling is incurred at each drop-off point of the route. In the case of space applications, the presence of additional constraints, such as the ones of balancing, and loading issues involving both vehicles and modules, is a peculiarity of this context. Requirements on vertical and horizontal stability, as well as load bearing strength, can also be imposed in some cases, due to the launch phase criticality. Mass stability constraints may also be taken into account, due to the specific loading facilities. This occurs, for instance, when clusters of bags have to be fastened together (e.g., on a panel) by means of belts. Moreover, on board unloading operations, multi-drop constraints may arise.

The proposed models can also be useful as reference models to deal with extended situations where nonstandard 3D-packing issues are present, such as in space engineering, aeronautical, naval and transportation systems, logistics, manufacturing, engineering of complex systems, etc. Some examples can be found in cargo accommodation of space vehicles and modules [13, 14].

The models were coded in the modeling language GAMS to evaluate their performance, and the CPLEX solver was used to solve different instances from the literature.

This work is organized as follows. In Sect. 12.2, we present two modeling paradigms for container loading problems, in which geometrical constraints play the main role. In Sect. 12.3, we incorporate additional practical three-dimensional loading constraints into one of the formulation of Sect. 12.2. In Sect. 12.4, we analyze the results of some computational tests for the model presented, solved by GAMS/CPLEX. Finally, in Sect. 12.5, we present concluding remarks and some perspectives for future research.

12.2 Modeling Paradigms for Container Loading Problems

In this section, we revisit some of the studies that made an effort to model the container loading problem or that can be easily extended for this purpose. The formulation can be broadly categorized as belonging to two modeling paradigms. The main difference between these two modeling paradigms is closely related to the possible positions where the boxes may be placed inside the container. In the following, we restate one instance of each of these two modeling paradigms. As mentioned, the presented formulation can be used to describe container loading problems of type 3/B/O/- or 3D-R-IIPP/SLOPP/SKP.

12.2.1 A Position-Free Paradigm

This modeling paradigm encompasses formulation in which the possible positions where the boxes may be placed inside the container are not defined a priori. In this category, the variables related to the position of the boxes inside the container are usually of real type. The formulation proposed by Tsai et al. [33], Chen et al. [8], and Beasley [2] (in this case, its direct extension) may be included in this modeling paradigm. Chen et al. [8] present a 0-1 mixed integer linear programming model for the multiple container loading problem, in which a subset of containers of different types is to be selected and loaded with all available boxes, and the objective is to minimize the total unused space of the selected containers. The indices, parameters, and variables used to formulate the single container loading problem are presented as follows:

12.2.1.1 Indices

i, j: indices for the boxes

12.2.1.2 Parameters

M: a sufficiently large number
m: the number of available boxes
v_{i}: the value of box i
(l_{i}, w_{i}, h_{i}): the length, width, and height, respectively, of box i
(L, W, H): the length, width, and height, respectively, of the container
($\{X^{\mathrm{o}}\}$,$\{Y^{\mathrm{o}}\}$,$\{Z^{\mathrm{o}}\}$): the position of the front-left-bottom corner of the container in a Cartesian coordinate system, respectively, along axes x, y *and* z

12.2.1.3 Decision Variables

p_{i}: binary variable that is equal to 1 if box i is packed inside the container, $i = 1, \ldots, m$, and it is equal to 0 otherwise.
(x_i, y_i, z_i): real variables that indicate the position of the front-left-bottom corner of box i, respectively, along axes x, y *and* z, *$i = 1, \ldots, m$ (see Fig. 12.1).*
$a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij}$: binary variables that indicate the relative positions between the boxes. For example, a_{ij} is equal to 1 if box i is placed entirely to the left of box j, i, $j = 1, \ldots, m$, and it is equal to 0 otherwise. Analogously, the other variables indicate if box i is placed entirely to the right of, in front of, behind, below, or above box j, respectively.

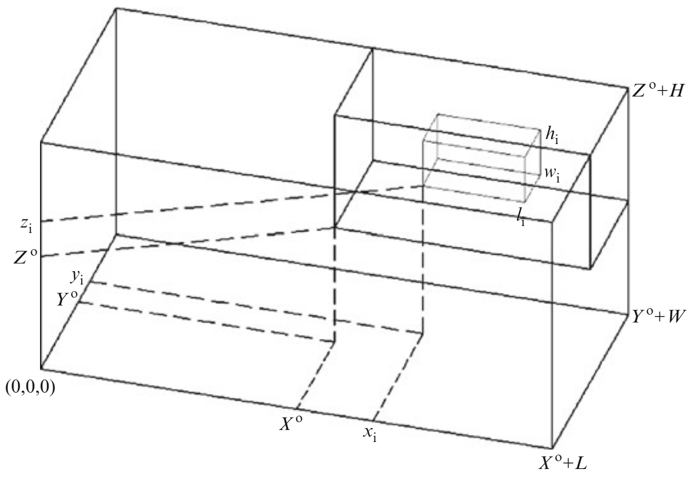


Fig. 12.1 Container position in the Cartesian coordinate system

The problem of loading rectangular boxes (with fixed orientation) inside a single container can be written as a particular case of the formulation proposed by Chen et al. [8].

12.2.1.4 Model

$$\max \sum_{\{i=1,\dots,m\}} v_i \cdot p_i \tag{12.1}$$

$$x_i + l_i \leq x_j + (1 - a_{ij}) \cdot \mathrm{M} \qquad i,j = 1,\dots,m, i<j \tag{12.2}$$

$$x_j + l_j \leq x_i + (1 - b_{ij}) \cdot \mathrm{M} \qquad i,j = 1,\dots,m, i<j \tag{12.3}$$

$$y_i + w_i \leq y_j + (1 - c_{ij}) \cdot \mathrm{M} \qquad i,j = 1,\dots,m, i<j \tag{12.4}$$

$$y_j + w_j \leq y_i + (1 - d_{ij}) \cdot \mathrm{M} \qquad i,j = 1,\dots,m, i<j \tag{12.5}$$

$$z_i + h_i \leq z_j + (1 - e_{ij}) \cdot \mathrm{M} \qquad i,j = 1,\dots,m, i<j \tag{12.6}$$

$$z_j + h_j \leq z_i + (1 - f_{ij}) \cdot \mathrm{M} \qquad i,j = 1,\dots,m, i<j \tag{12.7}$$

$$a_{ij} + b_{ij} + c_{ij} + d_{ij} + e_{ij} + f_{ij} \geq 1 \qquad i,j = 1,\dots,m, i<j \tag{12.8}$$

$$x_i \geq X^o \cdot p_i \qquad i = 1, ..., m \tag{12.9}$$

$$y_i \geq Y^o \cdot p_i \qquad i = 1, ..., m \tag{12.10}$$

$$z_i \geq Z^o \cdot p_i \qquad i = 1, ..., m \tag{12.11}$$

$$x_i + l_i \leq (X^o + L) \qquad i = 1, ..., m \tag{12.12}$$

$$y_i + w_i \leq (Y^o + W) \qquad i = 1, ..., m \tag{12.13}$$

$$z_i + h_i \leq (Z^o + H) \qquad i = 1, ..., m \tag{12.14}$$

$$x_i, y_i, z_i \geq 0 \qquad i = 1, ..., m \tag{12.15}$$

$$p_i \in \{0, 1\} \qquad i = 1, ..., m \tag{12.16}$$

$$a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij} \in \{0, 1\} \qquad i,j = 1, ..., m \tag{12.17}$$

In formulation (12.1)–(12.17), the objective function (12.1) aims at maximizing the total value of the boxes packed inside the container (if $v_i = (l_i \cdot w_i \cdot h_i)$, then (12.1) maximizes the total volume of the boxes), constraints (12.2)–(12.8) ensure the non-overlapping of the boxes packed, constraints (12.9)–(12.14) ensure that the packed boxes are completely inside the container, and constraints (12.15)–(12.17) are the decision variables domain constraints. Note that the values of (X^o, Y^o, Z^o) should be sufficiently large, so that all available boxes can fit into the fictitious container with dimensions $(X^o + L, Y^o + W, Z^o + H)$. This condition is necessary since, although some variables p_i are equal to 0, not all variables (x_i, y_i, z_i) can be equal to 0, that is, the non-overlapping constraints still apply for the boxes left out of the loading, and these boxes must be placed in a valid coordinate.

Note that the model grows substantially with the number of *boxes* to be loaded. Figure 12.2 shows two perspectives of an optimal solution obtained with formulation (12.1)–(12.17) and GAMS/CPLEX (versions 23.0/11.0). In this example we considered a container with dimensions $(L, W, H) = (12, 8, 8)$ units, and 15 boxes, with dimensions $(l_i, w_i, h_i) = (6, 3, 2)$ units for boxes $i = 1, 2$, $(l_i, w_i, h_i) = (6, 4, 3)$ units for boxes $i = 3, ..., 7$, $(l_i, w_i, h_i) = (8, 3, 2)$ units for boxes $i = 8, ..., 10$, $(l_i, w_i, h_i) = (4, 3, 2)$ units for boxes $i = 11, 12$, and $(l_i, w_i, h_i) = (4, 4, 3)$ units for boxes $i = 13, ..., 15$. We also defined v_i as the percentage of the container volume used by each box i, that is, $v_i = [(l_i \cdot w_i \cdot h_i)/(L \cdot W \cdot H)]$, for boxes $i = 1, ..., 15$. GAMS/CPLEX took 0.99 s to solve this instance. The solution depicted packs 13 boxes in 89.06% of the container volume (the colors differ the boxes with different dimensions). Boxes 2 and 9 are left out of the loading. Note in this figure that some of the packed boxes are floating inside the container (for instance, the darker box does not support the lighter box above it), which indicates a lack of practical considerations, particularly the vertical stability of the cargo.

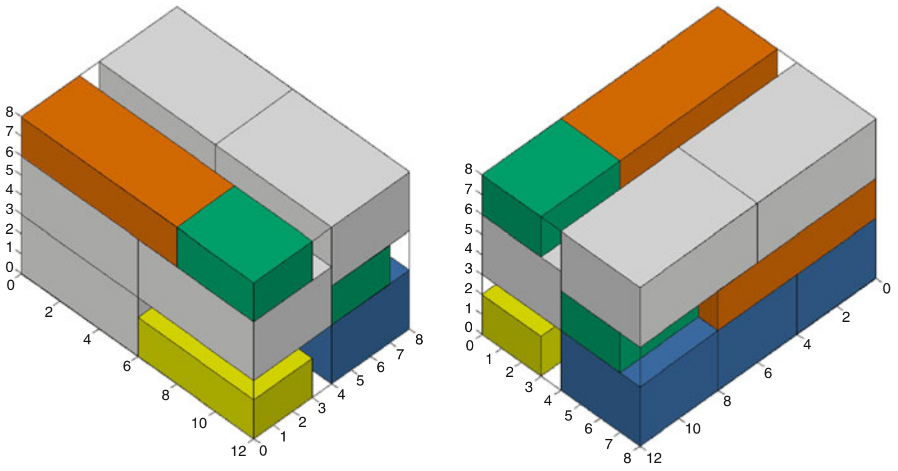


Fig. 12.2 Example of a solution obtained with formulation (12.1)–(12.17)

12.2.2 *A Grid-Based Position Paradigm*

This modeling paradigm encompasses formulation in which the possible positions where the boxes may be placed inside the container are defined a priori. In this category, the variables related to the position of the boxes inside the container are usually of integer type. The formulation proposed by Beasley [1] and Hadjiconstantinou and Christofides [19] (in this case, their direct extensions) may be included in this modeling paradigm. Beasley [1] presents a 0-1 integer linear programming model for the two-dimensional non-guillotine cutting problem, in which rectangular pieces of different types are to be cut from a single large rectangle, and the objective is to maximize the total value of the cut pieces. The indices, parameters, and variables used to formulate the single container loading problem are presented as follows:

12.2.2.1 Indices

i: index for the box types
p, s: indices for the possible positions along the *x-axis*
q, t: indices for the possible positions along the *y-axis*
r, u: indices for the possible positions along the *z-axis*

12.2.2.2 Parameters

m: the number of available box types
v_i: the value of box of type i

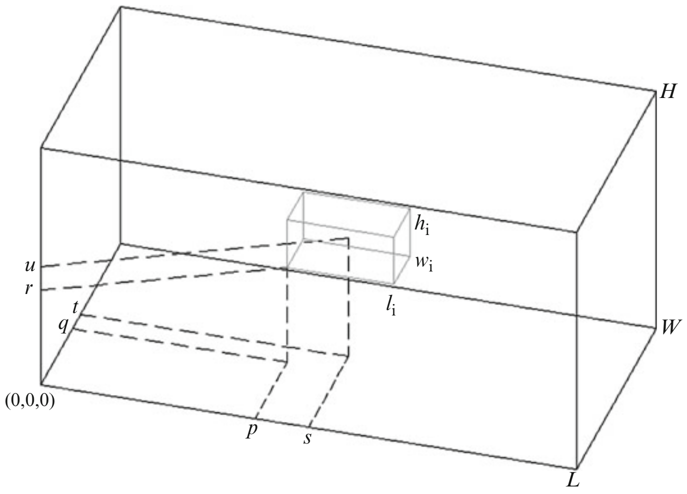


Fig. 12.3 Position of a box inside the container

b_i: the number of available boxes of type i
(l_i, w_i, h_i): the length, width, and height, respectively, of box of type i
(L, W, H): the length, width, and height, respectively, of the container

12.2.2.3 Decision Variable

x_{ipqr}: binary variable that is equal to 1 if a box of type i is placed with its front-left-bottom corner at position $(p, q, r), i = 1, \ldots, m, 0 \leq p \leq L - l_i, 0 \leq q \leq W - w_i,$ and $0 \leq r \leq H - h_i$, and it is equal to 0 otherwise (see Fig. 12.3) (the possible coordinates (p, q, r) where the front-left-bottom corner of a box can be placed, along axes x, y *and* z *of the container, belong to the sets* $X = \{p | 0 \leq p \leq L - \min_i(l_i)\}$, $Y = \{q | 0 \leq q \leq W - \min_i(w_i)\}$, *and* $Z = \{r | 0 \leq r \leq H - \min_i(h_i)\}$, *respectively); let* $X_i = \{p \in X | 0 \leq p \leq L - l_i\}$, $Y_i = \{q \in Y | 0 \leq q \leq W - w_i\}$, *and* $Z_i = \{r \in Z | 0 \leq r \leq H - h_i\}$, $i = 1, \ldots, m$*)*

The problem of loading rectangular boxes (with fixed orientation) inside a single container can be written as a direct extension of the formulation proposed by Beasley [1].

12.2.2.4 Model

$$\max \sum_{\{i=1,\ldots,m\}} \sum_{p \in X_i} \sum_{q \in Y_i} \sum_{r \in Z_i} v_i \cdot x_{ipqr} \qquad (12.18)$$

$$\sum_{\{i=1,\ldots,m\}} \sum_{\{p\in X_i|s-l_i+1\leq p\leq s\}} \sum_{\{q\in Y_i|t-w_i+1\leq q\leq t\}} \sum_{\{r\in Z_i|u-h_i+1\leq r\leq u\}} x_{ipqr} \leq 1 \qquad s\in X, t\in Y, u\in Z \tag{12.19}$$

$$\sum_{p\in X_i}\sum_{q\in Y_i}\sum_{r\in Z_i} x_{ipqr} \leq b_i \quad i=1,\ldots,m \tag{12.20}$$

$$x_{ipqr}\in\{0,1\} \quad \begin{array}{l} i=1,\ldots,m \\ p\in X_i, q\in Y_i, r\in Z_i \end{array} \tag{12.21}$$

In formulation (12.18)–(12.21), the objective function (12.18) aims at maximizing the total value of the boxes packed inside the container (if $v_i=(l_i\cdot w_i\cdot h_i)$, then (12.18) maximizes the total volume of the boxes), constraints (12.19) ensure the non-overlapping of the boxes packed, constraints (12.20) ensure a limit to the maximum number of boxes packed, and constraints (12.21) are the decision variables domain constraints. In case there is also a limit to the minimum number of boxes to be packed in the container, additional constraints similar to (12.20) can be included into the model.

For a given cutting or packing pattern, each packed box could be moved down and/or forward and/or to the left, until its bottom, front, and left-hand face are adjacent to other boxes or to the container. These patterns, called normal patterns, allow us, without loss of generality, to restrict the sets X, Y, and Z to [10]:

$$X=\left\{p|p=\sum\nolimits_{i=1}^{m}\varepsilon_i\cdot l_i, 0\leq p\leq L-\min_i(l_i), 0\leq\varepsilon_i\leq b_i \text{ and integer}, i=1,\ldots,m\right\} \tag{12.22}$$

$$Y=\left\{q|q=\sum\nolimits_{i=1}^{m}\varepsilon_i\cdot w_i, 0\leq q\leq W-\min_i(w_i), 0\leq\varepsilon_i\leq b_i \text{ and integer}, i=1,\ldots,m\right\} \tag{12.23}$$

$$Z=\left\{r|r=\sum\nolimits_{i=1}^{m}\varepsilon_i\cdot h_i, 0\leq r\leq H-\min_i(h_i), 0\leq\varepsilon_i\leq b_i \text{ and integer}, i=1,\ldots,m\right\} \tag{12.24}$$

Figure 12.4 shows, for the two-dimensional case (for the sake of ease of viewing), an example of the use of these sets as described previously (i.e., the normal patterns) when compared to the use of the sets as described in the decision variable statement (i.e., the full sets). In this example we considered a

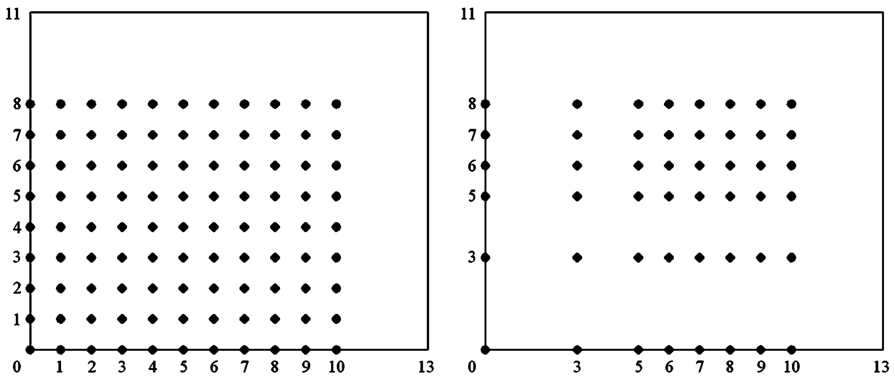


Fig. 12.4 Example with the use of the full sets and the normal patterns, respectively

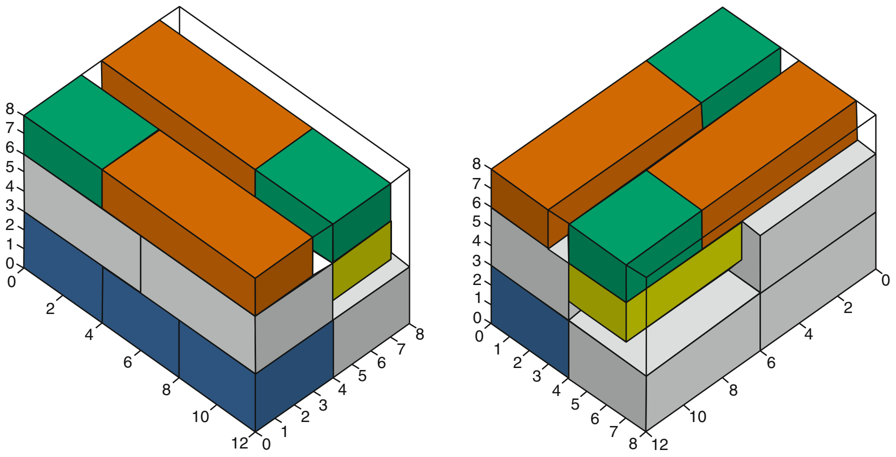


Fig. 12.5 Example of a solution obtained with formulation (12.18)–(12.21)

rectangular plate with dimensions $(L,W) = (13,11)$ units and 4 box types with dimensions $(l_1,w_1) = (3,3)$ units, $(l_2,w_2) = (5,3)$ units, $(l_3,w_3) = (5,7)$ units, and $(l_4,w_4) = (7,5)$ units. On the left are the sets obtained with the full sets, while on the right are the sets obtained with the normal patterns. Note that the size of the former is $|X| \cdot |Y| = 11 \cdot 9 = 99$, while the size of the latter is $|X| \cdot |Y| = 8 \cdot 6 = 48$, which is significantly fewer.

Note that the model grows substantially with the number of box *types* and with the number of *possible positions* along axes x, y and z. *Figure 12.5 shows, for the same instance of Fig. 12.2, two perspectives of an optimal solution obtained with formulation (12.18)–(12.21) (with sets X, Y, and Z defined as in (12.22), (12.23), and (12.24), respectively) and GAMS/CPLEX (versions 23.0/11.0). The 15 boxes* *are now grouped in 5 box types with dimensions $(l_1, w_1, h_1) = (6, 3, 2)$ units, $(l_2, w_2, h_2) = (6, 4, 3)$ units, $(l_3, w_3, h_3) = (8, 3, 2)$ units, $(l_4, w_4, h_4) = (4, 3, 2)$ units, and $(l_5, w_5, h_5) = (4, 4, 3)$ units, and amounts $b_1 = 2$, $b_2 = 5$, $b_3 = 3$, $b_4 = 2$, and $b_5 = 3$, respectively. The percentages of the container volume used v_i are now also redefined by each box type i, that is, $v_i = [(l_i \cdot w_i \cdot h_i)/ (L \cdot W \cdot H.)]$, for box types $i = 1, \ldots, 5$. GAMS/CPLEX took 0.22 s to solve this instance. As expected, the solution depicted also packs 13 boxes in 89.06% of the container volume (the colors differ the box types). One box of type 1 and one box of type 3 are left out of the loading. Once more, note in this figure that some of the packed boxes are floating inside the container.*

12.3 Embedding Additional Constraints

In this section, we show how other additional practical three-dimensional loading considerations can be embedded into formulation (12.18)–(12.21) (with sets X, Y, and Z defined as in (12.22), (12.23), and (12.24), respectively) or how this formulation can be adapted to consider such aspects [22, 23]. We begin addressing cargo vertical and horizontal stability constraints, followed by load bearing constraints and finally multi-drop constraints.

12.3.1 Vertical and Horizontal Stability Constraints

Cargo stability can be considered in terms of vertical and horizontal conditions. Vertical (or static) stability is related to the capacity of the loaded boxes to withstand the gravity force over them, that is, they are not displaced with respect to the *z-axis*. Vertical stability constraints prevent the boxes from falling over each other or on the container's floor. Horizontal (or dynamic) stability, on the other hand, is related to the capacity of the loaded boxes to withstand the inertia of their own bodies, that is, they are not displaced with respect to the x and y axes. Horizontal stability constraints prevent the boxes from moving around inside the container, due to variations in the speed of the displacement. The vertical stability constraints can be stated as follows:

$$\sum_{\{j=1,\ldots,m|r-h_j\geq 0\}} \sum_{\{p'\in X_j|p-l_j+1\leq p'\leq p+l_i-1\}} \sum_{\{q'\in Y_j|q-w_j+1\leq q'\leq q+w_i-1\}} L_{ij}\cdot W_{ij}\cdot x_{jp'q'(r-h_j)} \geq l_i\cdot w_i\cdot x_{ipqr}$$

$$\text{where}\begin{cases} L_{ij} = \min(p+l_i, p'+l_j) - \max(p,p') & i = 1,\ldots,m \\ W_{ij} = \min(q+w_i, q'+w_j) - \max(q,q') & p\in X_i, q\in Y_i, r\in Z_i\backslash\{0\} \end{cases} \tag{12.25}$$

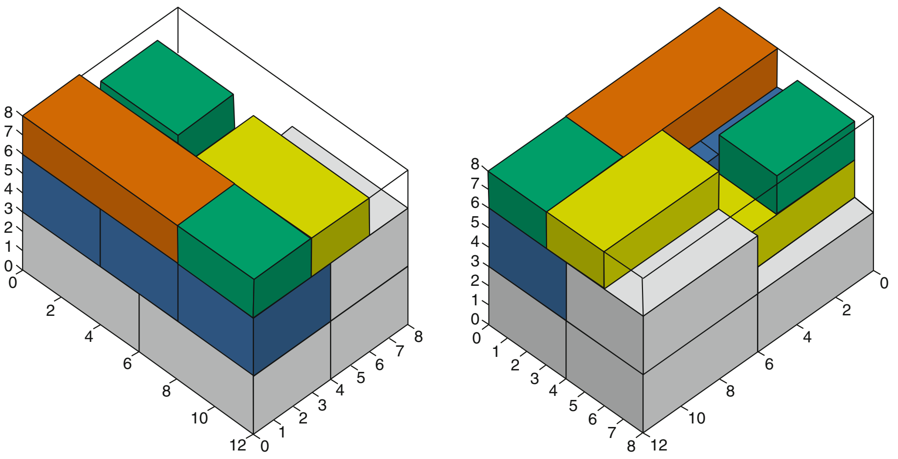


Fig. 12.6 Example of a solution obtained with formulation (12.18)–(12.21) with constrains (12.25)

that is, the area of the bottom face of a box of type *i* must be completely supported by the area of the top faces of one or more boxes placed immediately below it, or by the container's floor. Figure 12.6 shows, for the same instance of Figs. 12.2 and 12.5, two perspectives of an optimal solution obtained with formulation (12.18)–(12.21) with constraints (12.25) and GAMS/CPLEX (versions 23.0/11.0). GAMS/CPLEX took 0.29 s to solve this instance. The solution depicted packs 13 boxes in 87.50% of the container volume (the colors differ the box types). Two boxes of type 3 are left out of the loading. Note in this figure that all packed boxes are now completely stable.

Analogous formulation for the horizontal stability can be stated as follows:

$$\sum_{\{j=1,\ldots,m|p-l_j\geq 0\}}\ \sum_{\{q'\in Y_j|q-w_j+1\leq q'\leq q+w_i-1\}}\ \sum_{\{r'\in Z_j|r-h_j+1\leq r'\leq r+h_i-1\}} W_{ij}\cdot H_{ij}\cdot x_{j(p-l_j)q'r'} \geq w_i\cdot h_i\cdot x_{ipqr}$$
$$\text{where}\begin{cases} W_{ij}=\min(q+w_i,q'+w_j)-\max(q,q') & i=1,\ldots,m\\ H_{ij}=\min(r+h_i,r'+h_j)-\max(r,r') & p\in X_i\backslash\{0\},q\in Y_i,r\in Z_i\end{cases} \tag{12.26}$$

$$\sum_{\{j=1,\ldots,m|q-w_j\geq 0\}}\ \sum_{\{p'\in X_j|p-l_j+1\leq p'\leq p+l_i-1\}}\ \sum_{\{r'\in Z_j|r-h_j+1\leq r'\leq r+h_i-1\}} L_{ij}\cdot H_{ij}\cdot x_{jp'(q-w_j)r} \geq l_i\cdot h_i\cdot x_{ipqr}$$
$$\text{where}\begin{cases} L_{ij}=\min(p+l_i,p'+l_j)-\max(p,p') & i=1,\ldots,m\\ H_{ij}=\min(r+h_i,r'+h_j)-\max(r,r') & p\in X_i,q\in Y_i\backslash\{0\},r\in Z_i\end{cases} \tag{12.27}$$

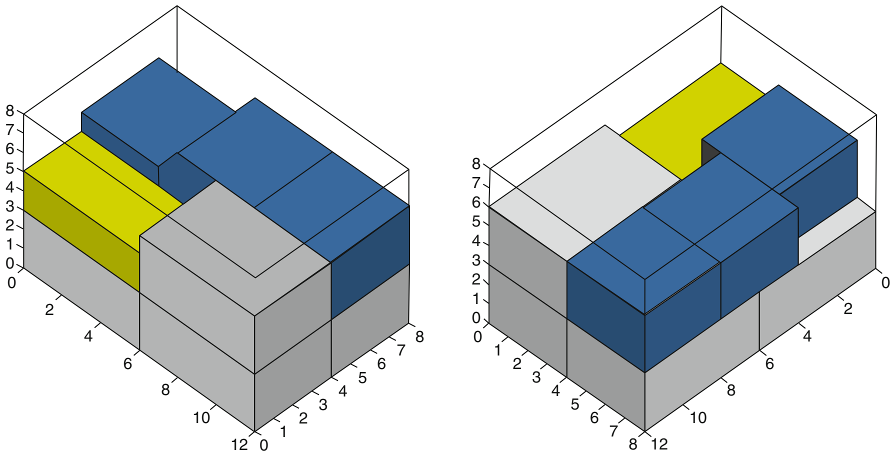


Fig. 12.7 Example of a solution obtained with formulation (12.18)–(12.21) with constrains (12.25) and (12.28)

that is, the area of the left (front) face of a box of type *i* must be completely supported by the area of the right (back) faces of one or more boxes placed immediately to the left (in front) of it or by the container's walls. Clearly, depending on the type of application, it is also possible to model analogous constraints to consider stability in an opposite direction along the same *axis* (e.g., constraints similar to constraints (12.25) to prevent the boxes from crashing against the container's roof).

12.3.2 Load Bearing Constraints

Load bearing strength is related to the maximum number of boxes that can be stacked one above each other or, more generally, to the maximum pressure that can be applied over the top face of a box. These constraints prevent damaging the products due to deformation of the boxes that contain them. We denote by P_i the weight of a box of type *i*, and by σ_i the maximum admissible pressure (given in units of force per unit of area) that each point at the top face of a box of type *i* can stand. The load bearing constraints can be stated as follows:

$$\sum_{\{j=1,\ldots,m\}} \sum_{\{p'\in X_j|s-l_j+1\le p'\le s\}} \sum_{\{q'\in Y_j|t-w_j+1\le q'\le t\}} \sum_{\{r'\in Z_j|u+1\le r'\le H-h_j\}} \left(\frac{P_j}{l_j\cdot w_j}\right)\cdot x_{jp'q'r'}$$
$$\le \sum_{\{i=1,\ldots,m\}} \sum_{\{p\in X_i|s-l_i+1\le p\le s\}} \sum_{\{q\in Y_i|t-w_i+1\le q\le t\}} \sum_{\{r\in Z_i|u-h_i+1\le r\le u\}} \sigma_i\cdot x_{ipqr}$$
$$s\in X, t\in Y, u\in Z \tag{12.28}$$

that is, for a certain point (s, t, u) of the container; if there is a box of type i that contains this point, then the boxes stacked over that box (not necessarily in direct contact to it) must not exceed the maximum admissible pressure σ_i (in terms of units of force per unit of area) that each point at the top face of that box can stand. If the top face of a box of type i cannot bear any kind of pressure, it is sufficient to set $\sigma_i = 0$ on the right-hand side of constraints (12.28). We observe that the vertical stability constraints (12.25) must be embedded into formulation (12.18)–(12.21) together with the load bearing constraints (12.28), in order to avoid empty spaces or "holes" in the loading pattern. Figure 12.7 shows, for the same instance of Figs. 12.2 and 12.5, two perspectives of an optimal solution obtained with formulation (12.18)–(12.21) with constraints (12.25) and (12.28) and GAMS/CPLEX (versions 23.0/11.0). The weight P_i was defined as the volume of a box of type i, since we assumed, for the sake of simplicity, that all boxes have the same density: $P_i = (l_i \cdot w_i \cdot h_i)$, $i = 1, ..., m$. The maximum admissible pressure σ_i that a box of type i can bear in any point of its top face was set as $\sigma_1 = 0, \sigma_2 = 3, \sigma_3 = 5, \sigma_4 = 0$, and $\sigma_5 = 3$, respectively, for the 5 box types. Note that boxes of types 1 and 3 do not bear any kind of pressure on their top faces, that is, they cannot have another box placed on top of them. GAMS/CPLEX took 0.06 s to solve this instance. The solution depicted packs 9 boxes in 70.31% of the container volume (the colors differ the box types). One box of type 1, three boxes of type 3, and two boxes of type 4 are left out of the loading. Note in this figure that all packed boxes are completely stable and that the presence of boxes that cannot bear any kind of pressure on their top faces incurred in a decrease of the number of packed boxes.

12.3.3 Multi-drop Constraints

Multi-dropping addresses situations where boxes that are delivered to the same customer (destination) must be placed close to each other inside the container (or truck, as it is more common in practice). The loading pattern, therefore, must take into account the delivery route of the vehicle and the sequence in which the boxes are unloaded. Multi-drop constraints prevent spending an unnecessary additional handling at each drop-off point of the route (unloading and reloading boxes). For each destination k $(k = 1, ..., n)$, there are b_{ik} boxes of type i $(i = 1, ..., m)$, so that $\sum_{k=1}^{n} b_{ik} = b_i$. We define δ_{ik} as the maximum reach (given in units of the container length) of a worker tasked to manually arrange a box of type i required by destination k inside the container (in some cases we may have $\delta_{ik} = \delta_i = \delta$ for all i and k). This parameter shows how many units of length, beyond the "border" of the boxes already packed, the worker is allowed to surpass in order to arrange the boxes of a customer that is visited earlier in the route. The "border" is a plane (or virtual wall) of type $(p, 0, 0)$ that is defined after all boxes of a certain customer (and other customers that are visited later in the route) are packed inside the container. For instance, if $\delta_{ik} = 0$, the worker cannot take advantage of any empty spaces left behind by the boxes of customers already packed to arrange the boxes of other customers. Note that this parameter can also represent the arm's reach of the worker, or even a piece of equipment used to load/unload the boxes, for instance, a forklift truck. Let us also define $L_k^{'}$ as an auxiliary real variable that indicates the length needed to pack all boxes of customer k (plus the boxes of other customers that are visited later in the route), and M as a sufficiently large number. Note that this variable defines the "border" aforementioned. We assume that the delivery route of the container is already known in advance. We note that $k = 1$ refers to the boxes that are loaded first and unloaded last and $k = n$ refers to the boxes that are loaded last and unloaded first. The multi-drop constraints can be stated in an adaptation of formulation (12.18)–(12.21):

$$\max \sum_{\{i=1,\dots,m\}} \sum_{\{k=1,\dots,n\}} \sum_{x\in X_i} \sum_{y\in Y_i} \sum_{z\in Z_i} v_i \cdot x_{ikpqr} \tag{12.29}$$

$$\sum_{\{i=1,\dots,m\}} \sum_{\{k=1,\dots,n\}} \sum_{\{p\in X_i|s-l_i+1\le p\le s\}} \sum_{\{q\in Y_i|t-w_i+1\le q\le t\}} \sum_{\{r\in Z_i|u-h_i+1\le r\le u\}} x_{ikpqr} \le 1 \quad s\in X, t\in Y, u\in Z \tag{12.30}$$

$$\sum_{p\in X_i} \sum_{q\in Y_i} \sum_{r\in Z_i} x_{ikpqr} \le b_{ik} \quad i=1,\dots,m, k=1,\dots,n \tag{12.31}$$

$$\sum_{\{j=1,\dots,m|r-h_j\ge 0\}} \sum_{\{k'=1,\dots,k\}} \sum_{\{p'\in X_j|p-l_j+1\le p'\le p+l_i-1\}} \sum_{\{q'\in Y_j|q-w_j+1\le q'\le q+w_i-1\}} L_{ij}\cdot W_{ij}\cdot x_{jk'p'q'(r-h_j)} \ge l_i\cdot w_i\cdot x_{ikpqr}$$
$$\text{where}\begin{cases} L_{ij} = \min(p+l_i, p'+l_j) - \max(p,p') & i=1,\dots,m, k=1,\dots,n \\ W_{ij} = \min(q+w_i, q'+w_j) - \max(q,q') & p\in X_i, q\in Y_i, r\in Z_i\backslash\{0\}\end{cases} \tag{12.32}$$

$$(p+l_i)\cdot x_{ikpqr} \le L_k^{'} \qquad i=1,\dots,m, k=1,\dots,n \quad p\in X_i, q\in Y_i, r\in Z_i \tag{12.33}$$

$$L_{k-1}^{'} - \delta_{ik} \le p\cdot x_{ikpqr} + (1-x_{ikpqr})\cdot \mathrm{M} \qquad i=1,\dots,m, k=2,\dots,n \quad p\in X_i, q\in Y_i, r\in Z_i \tag{12.34}$$

$$L_{k-1}^{'} \le L_k^{'} \quad k=2,\dots,n \tag{12.35}$$

$$0 \le L_k^{'} \le L \quad k=1,\dots,n \tag{12.36}$$

$$x_{ikpqr} \in \{0,1\} \quad i=1,\dots,m, k=1,\dots,n \quad p\in X_i, q\in Y_i, r\in Z_i \tag{12.37}$$

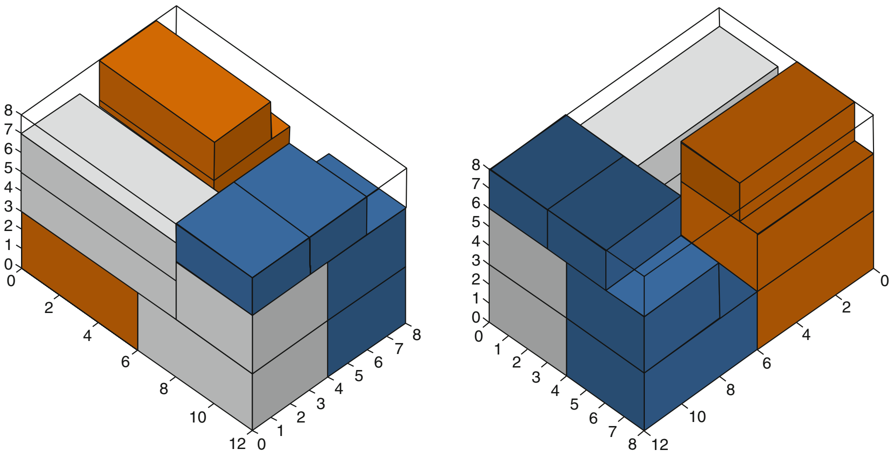


Fig. 12.8 Example of a solution obtained with formulation (12.29)–(12.37)

that is, if the container goes from destination k to destination $k-1$, then the boxes required by destination k are packed within the length limits $L'_{k-1} - \delta_{ik}$ and L'_k, and the length needed to pack all boxes required by destination k is within the length limits L'_{k-1} and L. We observe that the (adapted) vertical stability constraints (12.32) must be embedded into the (adapted) formulation (12.29)–(12.37) together with the multi-drop constraints (12.33)–(12.36), since the boxes of a certain destination must be placed either over the boxes of the same destination or over the boxes of a destination that will be visited later in the delivery route, that is, $\sum_{k'=1}^{k}$ on the left-hand side of constraints (12.32), in such a way that the handling of the boxes is not compromised (e.g., boxes required by customer k cannot be placed below boxes required by customers $k-1, k-2, ..., 1$). Figure 12.8 shows, for the same instance of Figs. 12.2 and 12.5, two perspectives of an optimal solution obtained with formulation (12.29)–(12.37). In this example, we considered that the 15 boxes are now divided by three customers/destinations, that is, $k = 1, ..., 3$, and the amounts are now $b_{i1} = (1, 3, 0, 0, 0)$, $b_{i2} = (0, 1, 2, 0, 2)$, and $b_{i3} = (1, 1, 1, 2, 1)$, for box types $i = 1, ..., 5$. The maximum reach of the worker δ_{ik} was arbitrarily set as l_i for each customer k, that is, the worker is allowed to surpass up to l_i units of length beyond the border to arrange the boxes of type i required by customer k. GAMS/ CPLEX took 5.18 s to solve this instance. The solution depicted packs 12 boxes in 82.81% of the container volume (the colors differ the boxes destinations). One box of type 1 (required by customer in "a darker color"), one box of type 3 (also required by customer in "a darker color"), and one box of type 5 (required by customer in "a lighter color") are left out of the loading. Note in this figure that all packed boxes are completely stable and that the loading pattern favors the multi-dropping of boxes.

12.4 Computational Results

In this section, we present some computational results achieved with some of the models presented in Sect. 12.2 and 12.3 with instances from the literature. All the models were implemented in the modeling language GAMS (version 23.0), and the solver CPLEX (version 11.0) was used to solve them. We set the parallel mode using up to 4 threads (of execution). All computational tests were performed in a PC Core i7 (3.40 GHz, 16.0 GB). Models (12.18)–(12.21), (12.18)–(12.21) with (12.25), and (12.18)–(12.21) with (12.25) and (12.28) were implemented to solve some of the instances presented in Lins et al. [25], while models (12.29)–(12.37) was implemented to solve some of the instances presented in Christensen and Rousoe [9]. In the experiments that follow, the computational time spent to solve each model was limited to 1 h (3,600 s), and the optimality gaps were computed as:

$$\text{Gap} = \frac{(\text{best bound obtained - best solution obtained})}{(\text{best bound obtained })}\ 100\%$$

Therefore, three possible cases, with respect to the quality of the solution obtained by GAMS/CPLEX, can occur: (i) optimal solution, with gap equals to zero; (ii) integer solution, with gap greater than zero and with CPLEX exceeding the time limit; (iii) no solution, without gap and with CPLEX exceeding the time limit. This last case is represented in the tables by the symbol "—". In order to facilitate the reading of the results, we refer to models (12.18)–(12.21) as model 1, models (12.18)–(12.21) with (12.25) as model 2, models (12.18)–(12.21) with (12.25) and (12.28) as model 3, and models (12.29)–(12.37) as model 4.

12.4.1 Results for the Instances of Lins et al. [25]

Lins et al. [25] present fourteen test instances consisting of a single box type per instance. The boxes do not have fixed orientation, that is, they are allowed to rotate and to be placed over any of their six faces. These instances can be seen as an application for the three-dimensional manufacture's pallet loading problem, where the boxes do not need to be arranged in horizontal layers on the pallet. Only the first seven instances were considered for these computational tests. The authors considered cubic containers with dimensions $(L,W,H) = (50,50,50)$ units. Models 1, 2, and 3 were properly adapted to consider multiple orientations of boxes. In the case of model 3, the weight P_i of a box of type i was arbitrarily set as its volume $(l_i \cdot w_i \cdot h_i)$, and the maximum admissible pressure σ_i that a box of type i can bear at any point of its top face was randomly generated by a uniform distribution in an interval taking into account which face of the box will be used as its supporting base: $[0, 3h_i]$ if the supporting base is $l_i \cdot w_i$, $[0, 3w_i]$ if the supporting base is $l_i \cdot h_i$, and $[0, 3l_i]$ if the supporting base is $w_i \cdot h_i$, $i = 1, ..., m$. Table 12.1 presents the results obtained with the instances and the models mentioned above. The first column shows the instance number, the second column shows the number of boxes that were packed in the solution of Lins et al. [25], and the third column shows the number of binary variables reported by CPLEX after preprocessing the models. Moreover, for each of the models, the number of constraints reported by CPLEX is also shown, the optimality gap (in %), the runtime (in seconds) spent to solve each instance, and the number of boxes packed in the solution.

Note that models 1 and 3 were able to find optimal solutions for the first three instances, while model 2 could only prove optimality of the third instance. Models 2 and 3 could not find integer solutions for the last two instances, while the remaining instances are sub-optimal integer solutions. Figure 12.9 shows the optimal loading patterns obtained for instance 3 with models 1, 2, and 3, respectively.

12.4.2 Results for the Instances of Christensen and Rousoe [9]

Christensen and Rousoe [9] present eight instances based on real-world data from a Danish company distributing construction products. The authors considered containers with dimensions $(L, W, H) = (720, 250, 280)$ units. Multiple orientations were allowed for some of the boxes. However, we assumed that rotations were not allowed, and we considered the largest box dimension to be placed along the container length, the smallest box dimension to be placed along the container width, and the intermediary box dimension to be placed along the container height. Besides that, we also limited the size of sets X, Y, and Z, by excluding the smaller dimensions along each *axis*, at a time, until no more that 20 positions were set. The value of the maximum reach of the worker δ_{ik} was arbitrarily set as l_i for each customer k, $k = 1, ..., n$, that is, the worker is allowed to surpass up to l_i units of length beyond the border to arrange the boxes of type i required by customer k. Table 12.2 presents the results obtained with these instances and model 4. The first column shows the instance number, the second and third columns show the number of customers/destinations and boxes available, respectively. Moreover, the number of boxes left out of the container in the solution of Christensen and Rousoe [9] is shown in the fourth column, and the remaining columns show the number of binary variables and constraints reported by CPLEX after preprocessing the model, the optimality gap (in %), the runtime (in seconds) spent to solve each instance, and the number of boxes left out of the container in the solution of model 4.

Note that model 4 was able to find optimal solutions for almost all instances, except the fourth and the last instances, which resulted in sub-optimal integer solutions. It should be observed that the approach proposed in Christensen and Rousoe [9] is not strictly comparable to ours, since their definition of multi-dropping is different from ours. In their approach, a box can occupy any empty space inside the container, if there is some access to the box at every drop-off point, regardless of the value of parameter δ_{ik}.

Table 12.1 Results obtained for the instances of Lins et al. [25]

	LLM02	Number	Model 1				Model 2				Model 3			
Problem	Number of boxes	of variables	Number of constraints	Gap (%)	Time (s)	Number of boxes	Number of constraints	Gap (%)	Time (s)	Number of boxes	Number of constraints	Gap (%)	Time (s)	Number of boxes
1	26	2,593	3,377	0.000	2.90	27	5,621	11.000	3,600.00	24	8,996	0.000	51.78	22
2	26	2,521	3,377	0.000	15.37	27	5,553	11.100	3,600.00	24	8,928	0.000	22.37	20
3	29	1,681	1,333	0.000	1.32	29	2,751	0.000	3,204.38	28	4,082	0.000	11.03	27
4	54	15,121	15,627	14.800	3,600.00	46	29,583	51.900	3,600.00	26	45,208	62.000	3,600.00	19
5	58	9,361	6,861	2.400	3,600.00	58	15,383	36.100	3,600.00	38	22,242	39.100	3,600.00	36
6	57	20,593	21,872	20.000	3,600.00	48	41,026	–	3,600.00	–	62,896	–	3,600.00	–
7	62	23,257	17,578	19.400	3,600.00	50	39,324	–	3,600.00	–	56,900	–	3,600.00	–

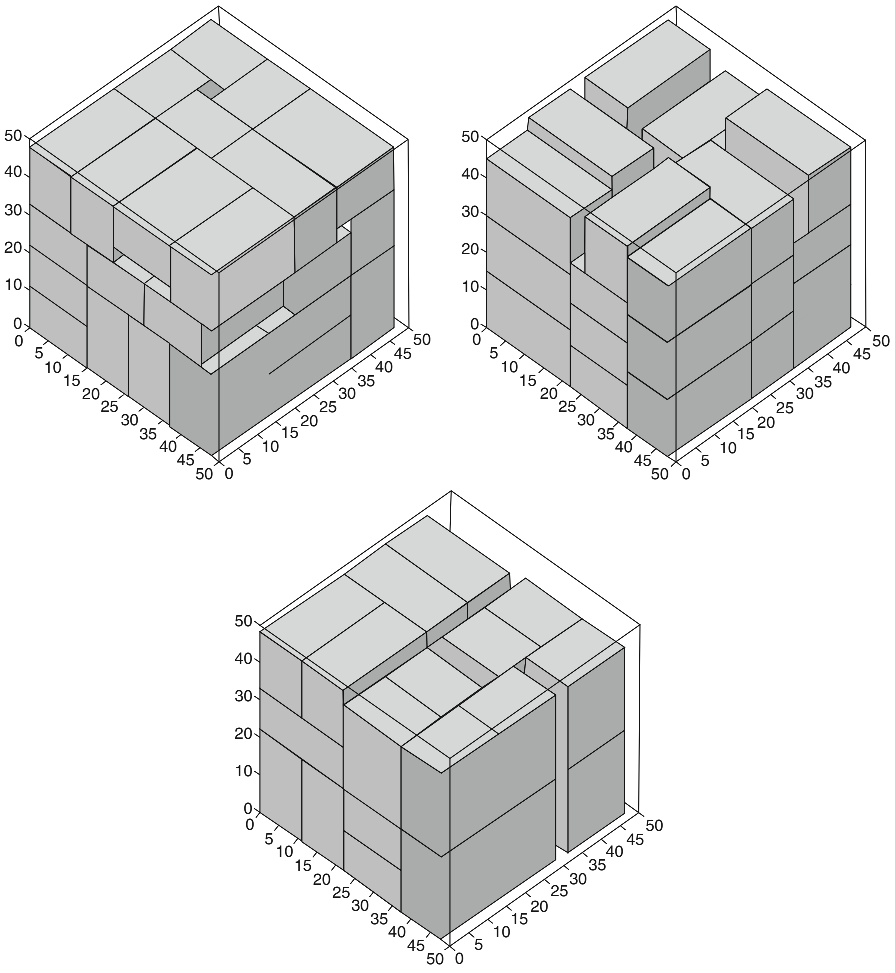


Fig. 12.9 Example of a solution obtained for one instance of Lins et al. [25]

12.5 Conclusion and Future Research

In this chapter, we presented mixed integer linear programming models for three-dimensional container loading problems. The objective is to find the best three-dimensional packing pattern for loading a set of boxes into a container so that the total volume (or value) of the boxes loaded is maximized and the boxes do not overlap. Apart from the non-overlapping of boxes, the vertical and horizontal stabilities of the cargo, the load bearing strength of the boxes, and the multi-dropping are also taken into account. In particular, the formulation are built on a grid-based position model. Computational tests using the proposed models were performed with instances from the literature using the GAMS/CPLEX software.

Table 12.2 Results obtained for the instances of Christensen and Rousoe [9]

			CR09	Model 4				
Problem	Number of customers	Number of boxes	Number of boxes left out	Number of variables	Number of constraints	Gap (%)	Time (s)	Number of boxes left out
1	2	3	0	407	1,356	0.000	0.05	0
2	1	3	0	182	448	0.000	0.03	0
3	2	9	0	8,046	19,468	0.000	12.61	0
4	3	22	0	27,886	76,209	3.260	3,600.00	4
5	4	8	0	8,705	25,360	0.000	16.41	0
6	2	8	0	10,353	31,578	0.000	2.30	0
7	1	11	0	3,998	10,257	0.000	6.98	0
8	6	19	0	15,357	48,216	16.661	3,600.00	11

The models proposed can be useful to motivate future research exploring decomposition methods, relaxation methods, and heuristics, among others, in order to solve more realistic container loading problems. Other interesting topics for future research are (i) to embed into the models other constraints that are common in the container loading literature, such as box orientation, box handling, box grouping, box separation, complete shipment of box groups, box priorities, complexity of the arrangement, container weight limit, and weight distribution within the container, among others, (ii) to build container loading formulation with practical constraints on a position-free model, such as Chen et al. [8] formulation, (iii) to extend the practical constraints considered in this chapter to address other variants of the container loading problems, such as the three-dimensional bin packing, multiple container loading and the strip packing, and (iv) to modify the models to deal with extended situations where nonstandard 3D-packing issues are present, such as in space engineering and cargo accommodation of space vehicles and modules [13, 14].

Acknowledgments This research was partially supported by FAPESP, CAPES, and CNPq.

References

1. Beasley, J.E.: An exact two-dimensional non-guillotine cutting tree search procedure. Oper. Res. **33**(1), 49–64 (1985)
2. Beasley, J.E.: A population heuristic for constrained two-dimensional non-guillotine cutting. Eur. J. Oper. Res. **156**(3), 601–627 (2004)
3. Bischoff, E.E.: Three-dimensional packing of items with limited load bearing strength. Eur. J. Oper. Res. **168**(3), 952–966 (2006)
4. Bischoff, E.E., Ratcliff, M.S.W.: Issues in the development of approaches to container loading. Omega **23**(4), 377–390 (1995)
5. Bortfeldt, A., Gehring, H.: A hybrid genetic algorithm for the container loading problem. Eur. J. Oper. Res. **131**(1), 143–161 (2001)

6. Bortfeldt, A., Gehring, H., Mack, D.: A parallel tabu search algorithm for solving the container loading problem. Parallel Comput. **29**(5), 641–662 (2003)
7. Bortfeldt, A., Wäscher, G.: Container loading problems - a state-of-the-art review. Magdeburg: Otto-von-Guericke Universität Magdeburg. Working Paper No. 7/2012 (2012)
8. Chen, C.S., Lee, S.M., Shen, Q.S.: An analytical model for the container loading problem. Eur. J. Oper. Res. **80**(1), 68–76 (1995)
9. Christensen, S.G., Rousoe, D.M.: Container loading with multi-drop constraints. Int. Trans. Oper. Res. **16**(6), 727–743 (2009)
10. Christofides, N., Whitlock, C.: An algorithm for two-dimensional cutting problems. Oper. Res. **25**(1), 30–44 (1977)
11. Dyckhoff, H.: A typology of cutting and packing problems. Eur. J. Oper. Res. **44**(2), 145–159 (1990)
12. Eley, M.: Solving container loading problems by block arrangement. Eur. J. Oper. Res. **141**(2), 393–409 (2002)
13. Fasano, G.: MIP-based heuristic for non-standard 3D-packing problems. 4OR **6**(3), 291–310 (2008)
14. Fasano, G.: A global optimization point of view to handle non-standard object packing problems. J. Global Optim. (2012). doi:10.1007/s10898-012-9865-8 (to appear)
15. Gehring, H., Bortfeldt, A.: A parallel genetic algorithm for solving the container loading problem. Int. Trans. Oper. Res. **9**(4), 497–511 (2002)
16. George, J.A., Robinson, D.F.: A heuristic for packing boxes into a container. Comput. Oper. Res. **7**(3), 147–156 (1980)
17. Gilmore, P.C., Gomory, R.E.: Multistage cutting stock problems of two and more dimensions. Oper. Res. **13**(1), 94–120 (1965)
18. Gonçalves, J.F., Resende, M.G.C.: A parallel multi-population biased random-key genetic algorithm for a container loading problem. Comput. Oper. Res. **39**(2), 179–190 (2012)
19. Hadjiconstantinou, E., Christofides, N.: An exact algorithm for general, orthogonal, two-dimensional knapsack problems. Eur. J. Oper. Res. **83**(1), 39–56 (1995)
20. Huang, W., He, K.: A caving degree approach for the single container loading problem. Eur. J. Oper. Res **196**(1), 93–101 (2009)
21. Jin, Z., Ohno, K., Du, J.: An efficient approach for the three-dimensional container packing problem with practical constraints. Asia-Pacific J. Oper. Res. **21**(3), 279–295 (2004)
22. Junqueira, L., Morabito, R., Yamashita, D.S.: Three-dimensional container loading models with cargo stability and load bearing constraints. Comput. Oper. Res. **39**(1), 74–85 (2012)
23. Junqueira, L., Morabito, R., Yamashita, D.S.: Mip-based approaches for the container loading problem with multi-drop constraints. Ann. Oper. Res. (2011). doi:10.1007/s10479-011-0942-z (to appear)
24. Lai, K.K., Xue, J., Xu, B.: Container packing in a multi-customer delivering operation. Comput. Ind. Eng. **35**(1–2), 323–326 (1998)
25. Lins, L., Lins, S., Morabito, R.: An n-tet graph approach for non-guillotine packing of n-dimensional boxes into an n-container. Eur. J. Oper. Res. **141**(2), 421–439 (2002)
26. Mack, D., Bortfeldt, A., Gehring, H.: A parallel hybrid local search algorithm for the container loading problem. Int. Trans. Oper. Res. **11**(5), 511–533 (2004)
27. Morabito, R., Arenales, M.: An And/Or-graph approach to the container loading problem. Int. Trans. Oper. Res. **1**(1), 59–73 (1994)
28. Moura, A., Oliveira, J.F.: A GRASP approach to the container-loading problem. IEEE Intell. Syst. **4**(20), 50–57 (2005)
29. Parreño, F., Alvarez-Valdes, R., Oliveira, J.F., Tamarit, J.M.: Neighborhood structures for the container loading problem: a VNS implementation. J. Heurist. **16**(1), 1–22 (2010)
30. Pisinger, D.: Heuristics for the container loading problem. Eur. J. Oper. Res. **141**(2), 382–392 (2002)
31. Ratcliff, M.S.W., Bischoff, E.E.: Allowing for weight considerations in container loading. OR Spektrum **20**(1), 65–71 (1998)

32. Silva, J.L.C., Soma, N.Y., Maculan, N.: A greedy search for the three-dimensional bin packing problem: the packing static stability case. Int. Trans. Oper. Res. **10**(2), 141–153 (2003)
33. Tsai, R.D., Malstrom, E.M., Kuo, W.: Three dimensional palletization of mixed box sizes. IIE Trans. **25**(4), 64–75 (1993)
34. Wang, Z., Li, K.W., Levy, J.K.: A heuristic for the container loading problem: a tertiary-tree-based dynamic space decomposition approach. Eur. J. Oper. Res. **191**(1), 86–99 (2008)
35. Wäscher, G., Haussner, H., Schumann, H.: An improved typology of cutting and packing problems. Eur. J. Oper. Res. **183**(3), 1109–1130 (2007)
36. Yeung, L.H.W., Tang, W.K.S.: A hybrid genetic approach for container loading in logistics industry. IEEE Trans. Ind. Electron. **52**(2), 617–627 (2005)

Chapter 13
Optimal Magnetic Cleanliness Modeling of Spacecraft

Klaus Mehlem

Abstract The magnetometers used by spacecraft for scientific research in the near-Earth and interplanetary space are highly sensitive. Since spacecraft contain in general some more or less magnetic parts which can impair scientific measurements, stringent magnetic cleanliness requirements have to be imposed on the spacecraft. In the domain of constant magnetics (magnetostatics), which is part of EMC (electromagnetic compatibility), modeling is a key issue for the verification of the magnetic cleanliness requirements. The paper describes the concept, improvements, and extensions of the multiple magnetic dipole modeling (MDM) method which had been introduced by the author in 1977 and which then has been used by numerous international scientific spacecraft projects during more than three decades. Specific issues, like the NLP method chosen and like the problem of the ambiguity of solutions, are presented in detail. Special techniques for the handling of model parameter constraints, for optimal MDM sizing, for avoidance of relative minima, and for multiple-point far-field compensation are presented as well. The extension of the MDM method to field gradient measurements is formulated and demonstrated by a significant example. Some challenging applications of MDM to spacecraft provide insight in practical modeling problems. Finally, a short description of the MDM software used is given.

Keywords Magnetic cleanliness • Multiple dipole model • Magnetic field and field gradient modeling • Magnetic testing • Magnetic compensation • Inversion problems

K. Mehlem (✉)
European Space Agency, Sonnenweg 22, D-56203 Hoehr-Grenzhausen, Germany
e-mail: klaus.mehlem@web.de

Acronyms

ASTOS	Astos Solutions GmbH, Germany
CNES	Centre National d'Etudes Spatiales, France
CSG	Centre Spatial Guyanais, French Guiana
CSP	Magnetic Cleanliness Specification Point
ECG	Electrocardiography
EEG	Electroencephalography
EMC	Electromagnetic Compatibility
ESA	European Space Agency, Paris
ESTEC	European Space Technology Centre, Netherlands
F2, F6, F7	RTG flight models
FGM	Fluxgate Magnetometer
FGMI	Inboard Magnetometer
FGMO	Outboard Magnetometer
GAMAG	MDM Software
GRB	Solar X-ray and Cosmic Gamma-Ray Burst Instrument
GSFC	Goddard Space Flight Center, USA
IABG	Industrieanlagen Betriebsgesellschaft, Germany
ISEE-B	International Sun-Earth Explorer
KSC	Kennedy Space Center, USA
MAG-1	Magnetometer
MCF	Mobile Coil Facility
MDM	Multiple Dipole Model
MEG	Magneto Encephalography
MFSA	Magnet-Field simulations-Anlage, IABG, Germany
NLP	Non-Linear Programming
RTG	Radioisotope Thermoelectric Power Generator
S/C	Spacecraft
SCS	Spacecraft Coordinate System
SNR	Signal-to-Noise Ratio
TCS	Test Coordinate System
TSS	Tethered Satellite System
TWT	Travelling Wave Tube
UCS	Unit Coordinate System
URAP	Unified Radio and Plasma Wave Instrument
VHM	Vector Helium Magnetometer

13.1 Interplanetary Magnetometry

Spacecraft generate in general some magnetic field disturbances which impair magnetometer and particle experiments, like those on the spacecraft Voyagers, GEOS, ISEE-B, Giotto, TSS, Ulysses, Cassini/Huygens, Cluster, Rosetta, and many others (Fig. 13.1).

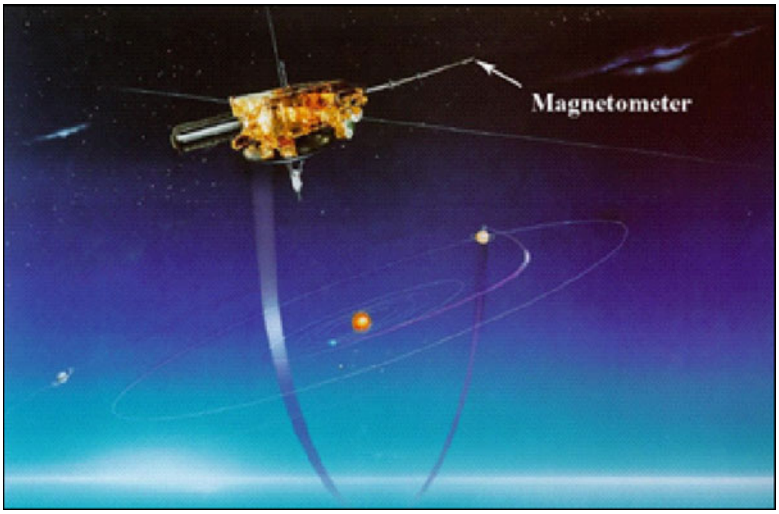


Fig. 13.1 Ulysses solar polar orbiter (image: ESA)

Even the fuel may be depleted earlier as planned when the attitude control system has to compensate the torque generated by the interaction of the spacecraft global dipole moment with a strong local magnetic field, like the one in low Earth orbits or close to Jupiter.

For interplanetary missions it is important to note that the field between Pluto and Mercury is in the range of 0.1–20 nT (Fig. 13.2). For these missions typical cleanliness specifications for the magnetic *cleanliness specification point* (CSP) are in the range of $0.1 < |\mathbf{b}^{\mathrm{sp}}| < 1.0$ nT.

13.2 Magnetic Cleanliness Verification

The verification of this specification (Ulysses: 0.1 nT, Cluster: 0.25 nT) has to be performed in tests on the ground. In order to shield the spacecraft from the Earth field (about 50,000 nT), different types and sizes of coil systems are in use which compensate the external fields by artificial counterfields which are generated by controlled currents injected into the coils (Figs. 13.6, 13.7, 13.8, and 13.9). Large coil facilities, like the one of IABG, achieve a resolution of 0.1 nT, an accuracy of 1 nT and at best a field uniformity of 0.5 nT within a sphere of $\Phi = 4$ m [3] (Fig. 13.7, yellow half-sphere).

Spacecraft with long magnetometer booms (for instance, the boom of Ulysses has a size of 6.45 m) exceed this volume. Even if the spacecraft volume was compatible with the size of the coil system, the verification of specifications lower than 0.5 nT by direct measurements would be difficult due to a number of disturbances.

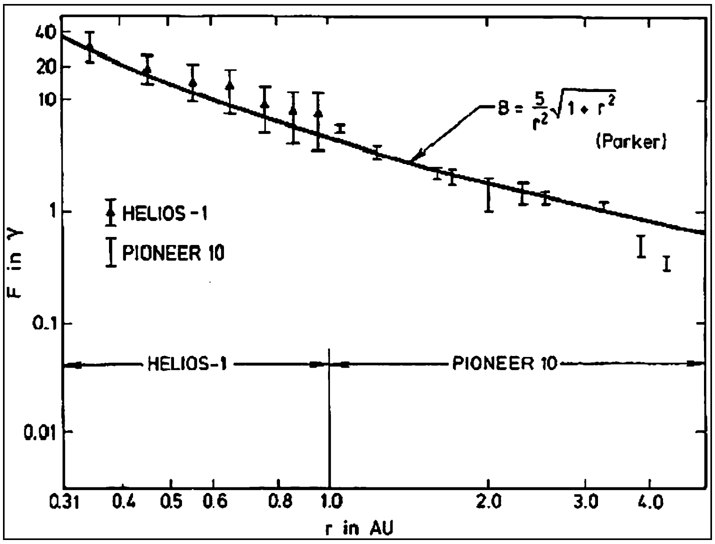


Fig. 13.2 Interplanetary magnetic field levels [1, 2]

This dilemma can be solved by the introduction of an intermediate step which consists in near-field measurements with high signal-to-noise ratio (SNR), taken close to the spacecraft, in the determination of an optimal system of dipoles by Least Squares, and in the calculation of the associated far-field at the CSP.

13.2.1 Concept

The concept of the above mentioned *magnetic cleanliness verification* is as follows (Fig. 13.3):

1. Measurement of the near-field at locations with high SNR.
2. Identification of the *multiple dipole model (MDM)* (position and moment vectors) which optimally fits the near-field measurements.
3. Calculation of the associated nonmeasurable far-field at the CSP.
4. Check of the magnetic cleanliness level w.r.t. the specification.
5. Should the far-field exceed the specification, a magnet can be determined to compensate the far-field.

Note that due to its physical analogy, the MDM method can easily be used for any multiple-point field alteration in the space outside the measurement distance (see Sect. 3.8).

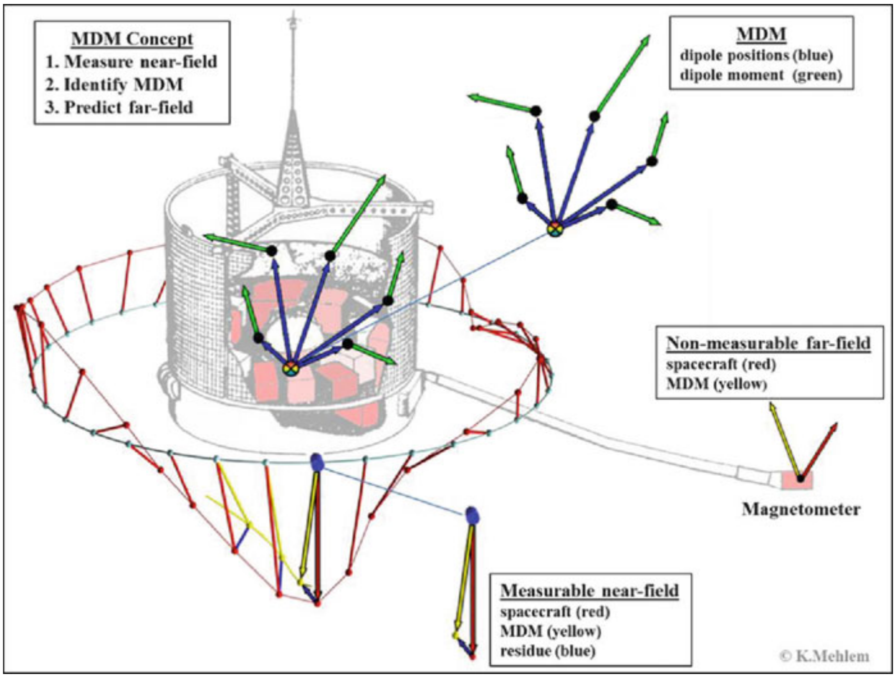


Fig. 13.3 Concept of magnetic cleanliness verification

13.2.2 Rotational Field Measurements

As mentioned earlier, magnetic field measurements are usually performed in coil facilities. The rotational mode is widely in use for test article with appropriate dimensions. The unit is placed on a turntable and rotated by steps of typically 10° around the vertical axis. At each rotational step magnetic field readings are taken by tri-axial facility probes, which are placed on the side of the turntable (Fig. 13.4). The measurements may be repeated with the object placed on its sides so as to generate multi-plane data coverage. But this is not always practicable.

The test distance has to be chosen such that the SNR of the field measurements is maximized while keeping the field smoothness compatible with the modeling capabilities of the method used (Fig. 13.5).

13.2.3 Magnetic Test Facilities

Figure 13.6 shows the 1.4 m mobile coil facility (MCF) of ESA–ESTEC, The Netherlands, which is used for small test articles like electronic boxes. It has been deployed at many locations, like in European industries and institutes, and also at launch sites like KSC, USA and CSG, French Guiana.

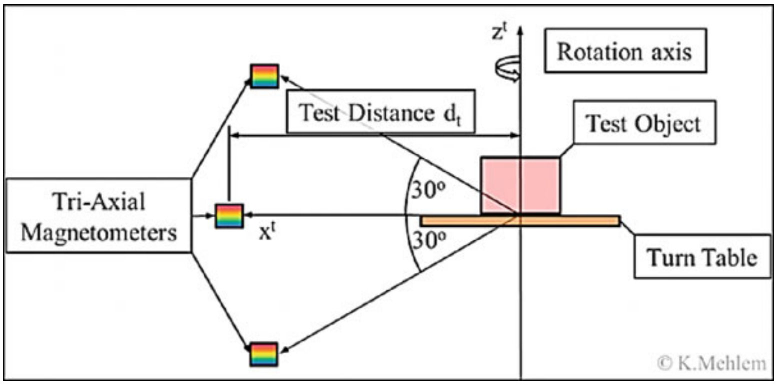


Fig. 13.4 Basic test-setup configuration for rotational measurements

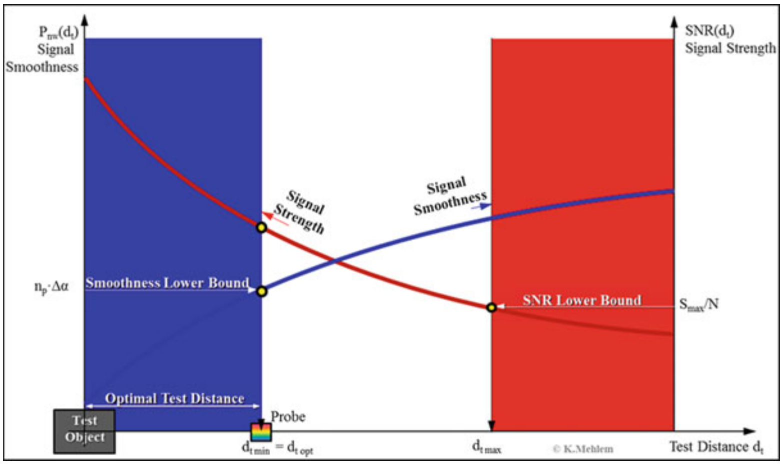


Fig. 13.5 Optimal test distance, compromise between signal smoothness and strength

Figure 13.7 shows the midsize 6.6 m coil facility of CNES, France, which has also a high accuracy; it is used for small to medium-sized test articles.

Figure 13.8 shows the large 13 m Braunbek coil facility of GSFC, USA, with the lunar rover in the center. In 1985 it has obtained the status of National Historic Landmark. In terms of accuracy, it is similar to the IABG facility.

Finally, Fig. 13.9 shows the large 15 m Helmholtz coil facility (MFSA) of IABG, Germany; it is used for all sizes of test articles, reaching from small units to complete spacecraft. The facility has one of the highest accuracies available and all European magnetometer carrying spacecraft have been tested there.

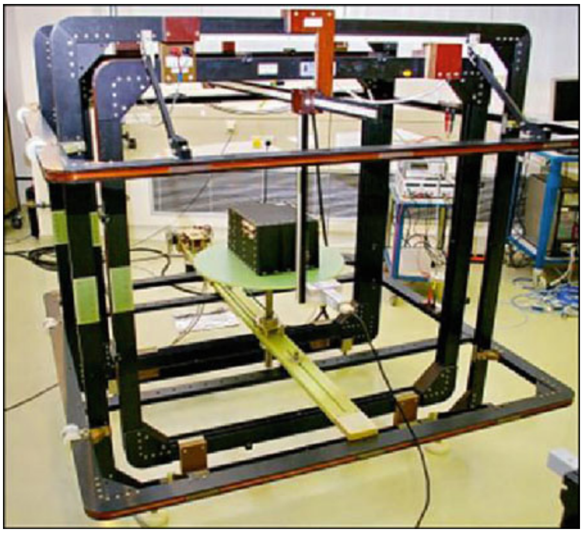


Fig. 13.6 1.4 m mobile coil facility (MCF) at ESTEC (image: ESA)

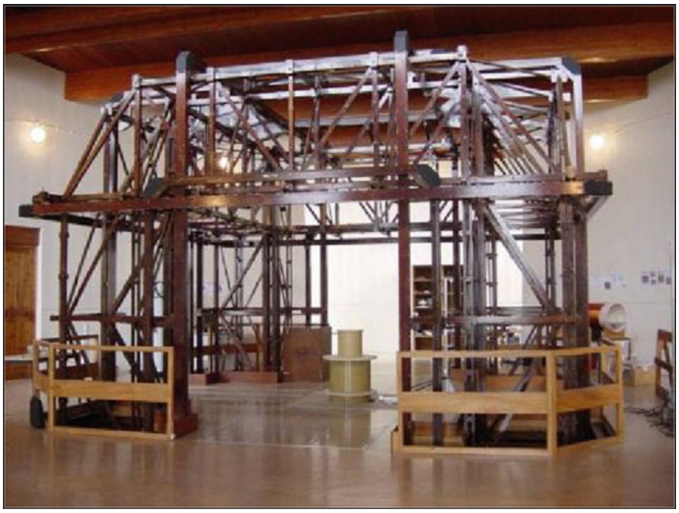


Fig. 13.7 6.6 m coil facility at CNES (image: CNES)

13.3 Multiple Dipole Modeling

13.3.1 Background

Benefiting from knowledge in parametric optimization of launcher and spacecraft trajectories, the author noticed in the early days of his activities in the European Space Agency that classical magnetic potential modeling by spherical harmonics

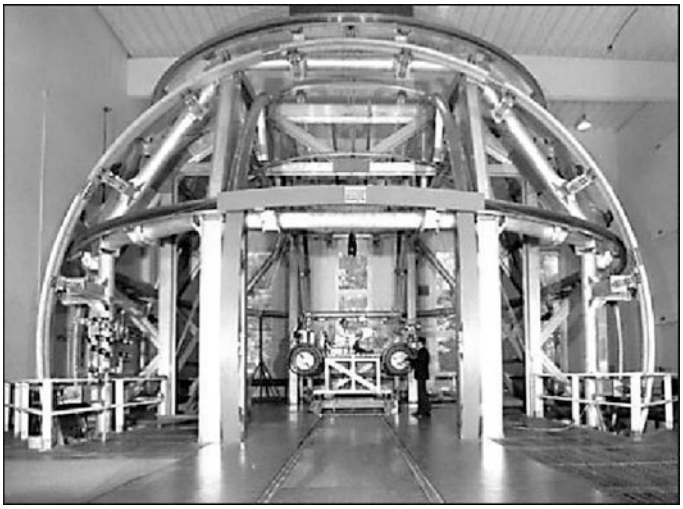


Fig. 13.8 13 m Braunbek coil facility at GSFC (image: GFSC)

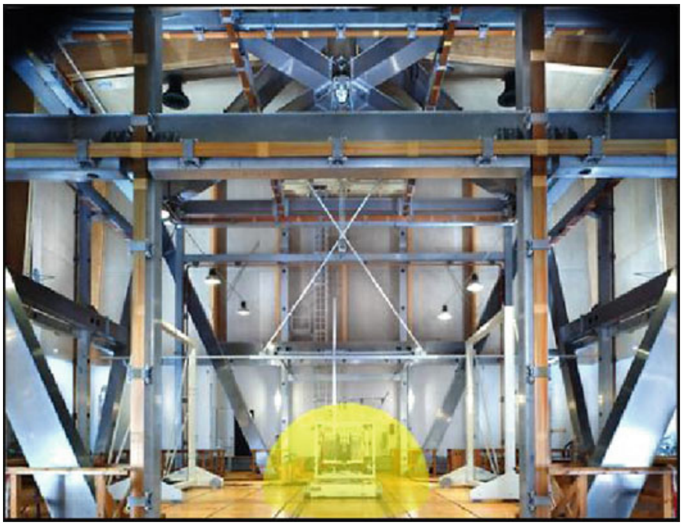


Fig. 13.9 15 m Helmholtz coil facility at IABG (image: IABG)

did not allow an easy physical interpretation of the model parameters. The multiple dipole modeling method, which is also applied in different disciplines like geology and medicine, seemed to be a good candidate for an alternative approach.

The method is based on the postulate that *any magnetostatic object can be represented by a set of dipoles* (MDM). The mathematical formulation and the related software had been developed by the author in 1976, and since then it has been continuously improved and applied to many ESA and international magnetometry missions. It is still in use as the standard modeling software in ESA. A recently developed significantly more powerful software will be presented in Sect. 4.5.

13.3.2 MDM Parameter Identification from Field Measurements

Definitions:

s	= scalar (regular lower case)
$\mathbf{v}$	= 3 × 1 vector (bold regular lower case)
$\mathbf{M}$	= 3 × 3 matrix (bold regular upper case)
$\boldsymbol{a}$	= n × 1 array (bold cursive lower case)
$\boldsymbol{A}$	= n × m array (bold cursive upper case)
n^s	= number of tri-axial sensors
n^d	= number of dipoles
$\mathbf{b}^c$	= calculated field vector
$\mathbf{b}^m$	= measured field vector
$\mathbf{r}^s$	= sensor position vector
$\mathbf{r}^d$	= dipole position vector
$\mathbf{m}^d$	= dipole moment vector
$\mathbf{m}^g$	$= \sum_{i=1}^{n^d} \mathbf{m}_i^d$ = global moment vector
$\boldsymbol{p}$	$= 3n^d \times 1$ array of optimizable dipole position vectors
$\boldsymbol{m}$	$= 3n^d \times 1$ array of optimizable dipole moment vectors
$\mathbf{M}$	$= 3n^d \times 2$ array $[\boldsymbol{p}\ \ \boldsymbol{m}]$ = MDM

A magnetic dipole moment vector $\mathbf{m}^d$, located at the position $\mathbf{r}^d$, is related to the field vector $\mathbf{b}$, located at the position $\mathbf{r}^s$, by the basic equation [4, 5]:

$$\mathrm{b} = \mathbf{D} \cdot \mathbf{m^d} \tag{13.1}$$

with the matrix

$$D = \frac{\mu_0}{4\pi} \cdot \left[\frac{3 \cdot \Delta r \Delta r^T - |\Delta r|^2 \cdot I}{|\Delta r|^5}\right] \tag{13.2}$$

and with the distance vector

$$\Delta r = r^s - r^d \tag{13.3}$$

Where μ_0 is the magnetic permeability in vacuum

If we use (cm) for the distance and (mA m^2) for the dipole moment then Eq. (13.1) becomes [4]

$$b\,[nT] = 10^5 \cdot D\,[cm^{-3}] \cdot m^d\,[mAm^2] \tag{13.4}$$

The matrix **D** represents the "Inverse Cubic Law" of the magnetic field. We note that the field is highly nonlinear w.r.t. the difference vector **Δr**, whereas the moment is linear. Therefore **m** can be calculated directly by inverting the matrix **D** (Eq. 13.1):

$$m^d = D^{-1} \cdot b \tag{13.5}$$

For multiple dipoles, the field vectors **b**$_i$, located at n^s sensor positions r_i^s (i = 1, …, n^s), are the vector sum of the n^d individual field vectors which are generated by n^d dipole moment vectors m_k^d located at positions r_k^d (k = 1, …, n^d):

$$b_i = \sum_{k=1}^{n^d} D_{ik} \cdot m_k^d \tag{13.6}$$

with the 3 × 3 matrix

$$D_{ik} = \frac{\mu_0}{4\pi} \cdot \left[\frac{3 \cdot \Delta r_{ik} \Delta r_{ik}{}^T - |\Delta r_{ik}|^2 \cdot I}{|\Delta r_{ik}|^5} \right] \tag{13.7}$$

and with

$$\Delta r_{ik} = r_i^s - r_k^d \tag{13.8}$$

We collect all vectors **b**$_i$, r_k^d, m_k^d, and all 3 × 3 matrices **D**$_{ik}$ in the following arrays:

$$\begin{aligned} p^T &= \left[r_1^d \ldots r_k^d \ldots r_{n^d}^d\right] \\ m^T &= \left[m_1^d \ldots m_k^d \ldots m_{n^d}^d\right] \\ b^T &= \left[b_1 \ldots b_i \ldots b_{n^s}\right] \\ D^T &= \left[D_{11} \ldots D_{ik} \ldots D_{n^s n^d}\right] \end{aligned} \tag{13.9}$$

Where ***p*** and ***m*** are 3n^d × 3 arrays, ***b*** is a 3n^s × 1 array, and ***D*** is a 3n^s × 3n^d array, or matrix.

So, we can condense Eq. (13.6) to:

$$b = D \cdot m \tag{13.10}$$

Similar to Eq. (13.5), we can write for $n^s = n^d$:

$$m = D^{-1} \cdot b \tag{13.11}$$

For $n^s < n^d$, however, Eq. (13.11) can only be solved by use of the left pseudo-inverse $\boldsymbol{D}^+$ [6]:

$$m = D^+ \cdot b = \left[D^T \cdot D\right]^1 \cdot D^T \cdot b \tag{13.12}$$

If we take real field measurements $b^{m,}$ the optimal moments become (Eq. 13.12):

$$m^{opt} = D^+ \cdot b^m \tag{13.13}$$

It can be shown that the solution $\mathbf{m}^{d\ opt}$ is of the type “least squares” [6].

From Eqs. (13.7) and (13.8) it is evident that the matrix $\boldsymbol{D}$ contains the dipole positions r_k^d. Hence, the optimal dipole moments are a function of r_k^d:

$$m^{opt} = f\left(r_k^d\right) \tag{13.14}$$

As the dipole positions r_k^d appear in the matrix $\boldsymbol{D}$ in a highly nonlinear form, they have to be calculated by use of nonlinear programming (NLP) methods, as explained hereafter.

13.3.3 NLP Approach

By the combination of Eqs. (13.10) and (13.12) the so-called calculated field is:

$$b^c = D \cdot \left[D^T \cdot D\right]^{-1} \cdot D^T \cdot b^m \tag{13.15}$$

According to Eq. (13.9), the array $\boldsymbol{p}$ represents the dipole positions. The field error ε is defined as the difference between the measured and the calculated field vector:

$$\varepsilon(p) = b^m - b^c(p) \tag{13.16}$$

Eq. (13.16), we can write

$$\varepsilon = \left\{I - D \cdot \left[D^T \cdot D\right]^{-1}\right\} \cdot b^m \tag{13.17}$$

The quadratic cost function to be minimized in the sense of a least square fit is defined by

$$c(p) = \varepsilon(p)^T \cdot \varepsilon(p) = \sum\nolimits_{i=1}^{n_s} \sum\nolimits_{j=1}^{3} \varepsilon_{ij}^2\ (p) \tag{13.18}$$

In the following lines we derive the Gauss–Newton algorithm. The development of c to the second order is

$$c = c_0 + \frac{\partial c}{\partial p}\Delta p + \frac{\partial^2 c}{\partial p^2}\Delta p^2 + 0 \tag{13.19}$$

The derivative of c w.r.t. ***p*** is:

$$\frac{\partial c}{\partial p} = \frac{\partial^2 c}{\partial p^2}\Delta p + \frac{\partial \Delta p}{\partial p}\frac{\partial c}{\partial p} + 0 \tag{13.20}$$

When setting $\frac{\partial c}{\partial p} = 0$ we obtain the step $\mathbf{\Delta p}$ leading to the minimum:

$$\Delta p = -\left[\frac{\partial^2 c}{\partial p^2}\right]^{-1} \cdot \frac{\partial c}{\partial p} \tag{13.21}$$

with

$$\frac{\partial c}{\partial p} = 2 \cdot \frac{\partial \varepsilon^T}{\partial p}\varepsilon \tag{13.22}$$

and with

$$\frac{\partial^2 c}{\partial p^2} = \frac{\partial}{\partial p}\left(\frac{\partial \varepsilon^T}{\partial p}\varepsilon\right) + \frac{\partial}{\partial p}\left(\varepsilon^T \frac{\partial \varepsilon}{\partial p}\right) \tag{13.23}$$

where

$$\frac{\partial}{\partial p}\left(\frac{\partial \varepsilon^T}{\partial p}\varepsilon\right) = \frac{\partial^2 \varepsilon^T}{\partial p^2}\varepsilon + \frac{\partial \varepsilon}{\partial p}\cdot\frac{\partial \varepsilon^T}{\partial p} = 0 + \frac{\partial \varepsilon^T}{\partial p}\cdot\frac{\partial \varepsilon}{\partial p} \tag{13.24}$$

and where

$$\frac{\partial}{\partial p}\left(\varepsilon^T \frac{\partial \varepsilon}{\partial p}\right) = \frac{\partial \varepsilon^T}{\partial p}\cdot\frac{\partial \varepsilon}{\partial p} + \frac{\partial^2 \varepsilon}{\partial p^2}\varepsilon^T = \frac{\partial \varepsilon^T}{\partial p}\cdot\frac{\partial \varepsilon}{\partial p} + 0 \tag{13.25}$$

Due to $\varepsilon \rightarrow 0$ in the neighborhood of the minimum the term $\frac{\partial^2 \varepsilon^T}{\partial p^2}\varepsilon$ in Eqs. (13.24) and (13.25) can be neglected.

So, finally we have

$$\frac{\partial^2 c}{\partial p^2} = 2 \cdot \frac{\partial \varepsilon^T}{\partial p} \cdot \frac{\partial \varepsilon}{\partial p} \tag{13.26}$$

The Jacobian is a $3n^s \times 3n^d$ matrix:

$$J = \frac{\partial \varepsilon}{\partial p} \tag{13.27}$$

By inserting Eq. (13.26) into Eqs. (13.21) and (13.25), and both into Eq. (13.20), we obtain the Gauss–Newton algorithm [7]:

$$\Delta p = -\left[J^T \cdot J\right]^{-1} \cdot J^T \cdot \varepsilon \tag{13.28}$$

where the term $\left[J^T \cdot J\right]$ represents the approximated Hessian matrix.

Since the cost function c is *not* a quadratic function of $\boldsymbol{p}$ (see Eq. 13.7), $\Delta \boldsymbol{p}$ is not leading to the minimum in one step.

Let us write Eq. (13.27) in form of a series of iterations (i = counter of NLP iterations):

$$p^{i+1} = p^i - \lambda^i \cdot \left[J^i\right]^+ \cdot \varepsilon^i \tag{13.29}$$

Note that $\left[J^i\right]^+ = \left[J^T \cdot J\right]^{-1} \cdot J^{iT}$ is the left pseudo inverse of $\boldsymbol{J}^i$ [6].

$\left[J^i\right]^+ \cdot \varepsilon^i$ can be interpreted as a search direction $\mathbf{d}^i$ in the parameter space p:

$$d^i = \left[J^i\right]^+ \cdot \varepsilon^i \tag{13.30}$$

So Eq. (13.29) becomes:

$$p^{i+1} = p^i - \lambda^i \cdot d^i \tag{13.31}$$

where λ^i represents a parameter shift or progress factor used in the line search.

We start with an initial guess $\boldsymbol{p}^0$ (Fig. 13.10, middle part left) and calculate the search direction $\boldsymbol{d}^0$ Eqs. (13.7), (13.15), and (13.30).

In a so-called line search, $\boldsymbol{p}^{i+1}$ is calculated by

$$p^{i+1} = p^i - \lambda^{i\,opt} \cdot d^i \tag{13.32}$$

$\lambda^{i\,opt}$ is determined in the following way (j = counter of line-search iterations) (Fig. 13.10, lower part): Starting from a sufficiently low value of $\lambda^{j=0}$, such that $c^{j=0} < c^{j=0}$, λ is increased by a suitable factor $u > 1$:

$$\lambda^{j+1} = u \cdot \lambda^j \tag{13.33}$$

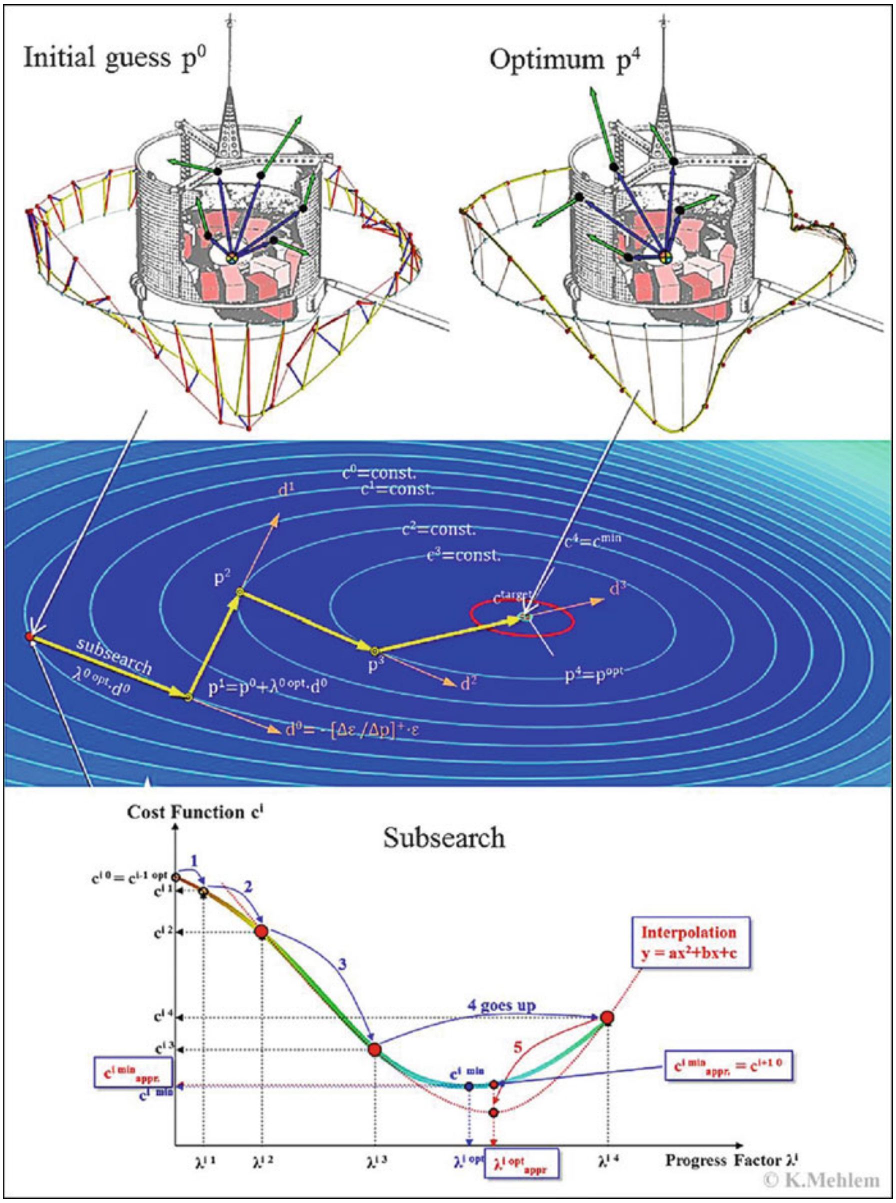


Fig. 13.10 Initial and final data fit (*top*), NLP iterations (*mid*), and line search (*bottom*)

Then a step is carried out

$$p^{j+1} = p^j - \lambda^{j+1} \cdot d^0 \tag{13.34}$$

and c($\boldsymbol{p}^{j+1}$) is calculated. This is repeated until c increases ($c^{j+1} > cj$).

Finally, a good approximation of $c^{j+1} = c^{\min}(\lambda^{i\ opt}, d^j)$ is obtained by use of a cubic interpolation between the last three points λ^{j-1}, λ^{j} and λ^{j+1}.

$$c^{i+1} = c^{\min}(d^j) = c(\lambda^{i\ opt}, d^i) \leq c(\lambda)\ \forall\ \lambda \tag{13.35}$$

At this point the parameter $\boldsymbol{p}^{j+1}$ becomes the new starting parameter $\boldsymbol{p}^{i+1}$ for the primary NLP process. The main iteration (Eq. 13.30) is stopped when the cost function c is below the facility noise τ:

$$c^{i+1} \leq \tau \tag{13.36}$$

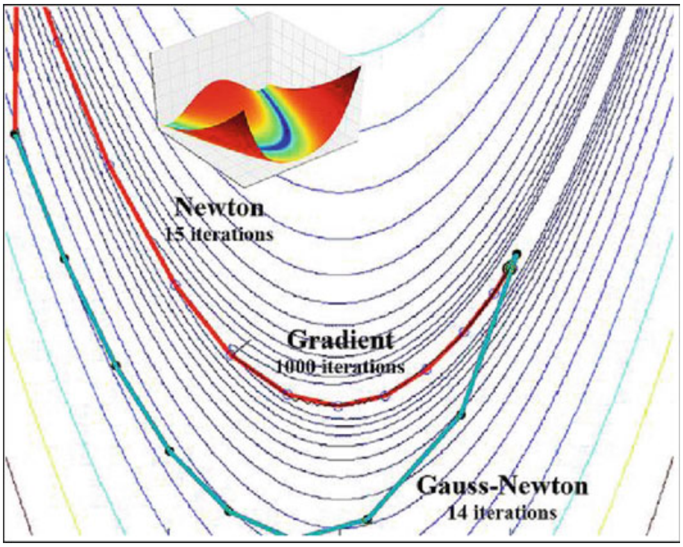
Here we have found the *optimal position parameter*$p^{opt} = p^{i+1}$. The corresponding optimal MDM is described by the $3n^d \times 2$ array $\boldsymbol{M}$:

$$M^{opt} = \left[p^{opt}\, m^{opt} = \left[G^T \cdot G\right]^{-1} \cdot G^T \cdot g^m\right] \tag{13.37}$$

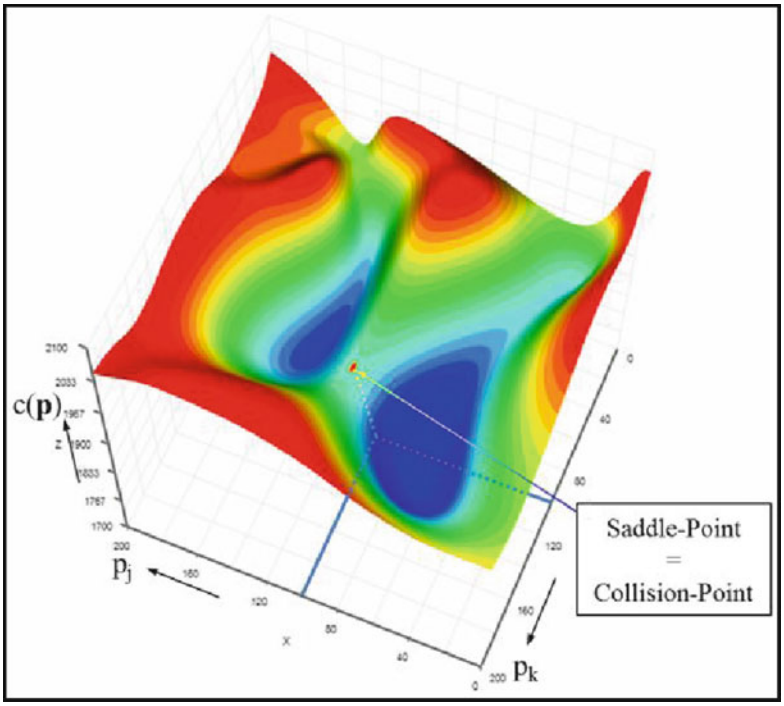
The optimal dipole positions $r_{d_k}^{opt}$ can now be extracted from the array $\mathbf{p}^{opt}$ according to Eq. (13.9). The optimal dipole moments m_k^{opt} have already been calculated implicitly by Eq. (13.13).

The main characteristics of the Gauss–Newton algorithm (Eq. 13.27) are:

1. It requires only *first derivatives* in form of the Jacobian matrix $\boldsymbol{J}$.
2. It *never diverges*, due to the semipositivity by construction of the approximated Hessian matrix $[\boldsymbol{J}^T\boldsymbol{J}]$.
3. Its *convergence speed is superlinear*. $\rightarrow$
   Newton (red, 2nd der.,15 its.)
   Gauss–Newton (turq., 1st der., 14 its.)
   Gradient (saw pattern, 1stder., >1,000 its., barely visible on red line) [8]



4. For c $\rightarrow 0$ the *convergence speed* (number of iterations) tends to be *similar to the Newton method* [9].
5. A significant *increase of convergence speed and robustness* is obtained by solving for the dipole moments in a sub-procedure.
6. The NLP solver converges well *with up to 40 dipoles*.
7. Solving the linear problem for the optimal moments alone works well for 100 and more dipoles.
8. The convergence *stops if the positions of two dipoles are identical*.
   The Jacobian $\boldsymbol{J}$ is not of full rank, so the Hessian $[\boldsymbol{J}^T\boldsymbol{J}]$ cannot be inverted. $\rightarrow$



9. Gauss–Newton is only formulated for free parameter spaces.
   Constraints have to be handled by use of another approach (Sect. 3.5).

13.3.4 Optimal MDM Sizing

Till here we assumed that the number of dipoles $\mathbf{n}^{d\ \min}$, necessary to reach the minimum $c^{\min} < \tau$, was known. Unfortunately, it is a basic unknown in modeling problems. For its determination we use the following procedure: After starting with $n^d = 1$ a solution is obtained which satisfies the condition $\frac{\partial c}{\partial p} << 1\ \forall\ c$. By increasing n^d by 1 a new solution is obtained with $c^{i+1.} < c^i$. Finally, the minimum necessary number of dipoles is found when the condition $c < \tau\ \forall \frac{\partial c}{\partial p}$ is met

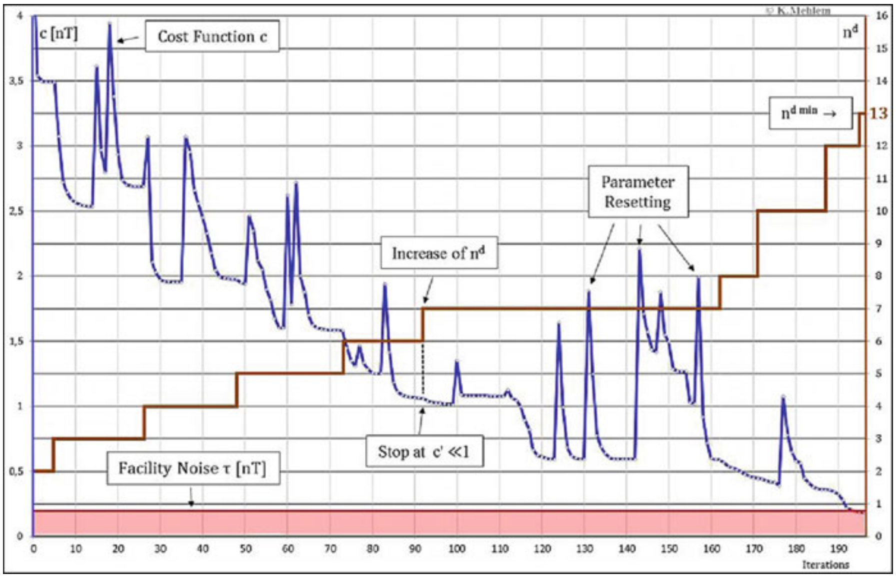


Fig. 13.11 Determination of the minimum necessary number of dipoles n$^{\mathrm{d\ min}}$ for $c < \tau\ \forall\ \frac{\partial c}{\partial p}$

(Fig. 13.11, blue line entering the red zone). Note that *this algorithm always converges*, although it does not necessarily lead to $\mathbf{n}^{\mathrm{d\ min}}$.

Since the far-field shows a significant scattering for different well-fitting MDM solutions ($c^{\min} < \tau$) (Fig. 13.18), we continue in a second phase to increase the number of dipoles beyond $\mathbf{n}^{\mathrm{d\ min}}$. The empirical function $Si\left(n_i^d\right)$ which represents a measure of the far-field scattering, is evaluated over a window of five solutions by the following equation: (sp* stands for specification point):

$$Si = \frac{n_i^d}{5} \cdot \sqrt{\sum_{i=1}^{5}\left(\left|b_i^{sp*}\right| - \frac{1}{5}\cdot\sum_{j=i-2}^{j=i+2}\left|b_j^{sp}\right|\right)^2} \tag{13.38}$$

The optimal MDM size $\mathbf{n}^{\mathrm{d\ opt}}$ is found when $Si\left(n_i^d\right)$ is minimum. At higher numbers of dipoles $n_i^d > n^{d\,opt}$ the function S increases again due to problems of mis-modeling by over-parameterization (Figs. 13.12 and 13.14).

13.3.5 Parameter Constraints

There are two kinds of constraints imposed on the dipole positions: *boundary constraints* for each dipole and *proximity constraints* for dipole pairs (point. 7 above). These constraints are enforced by use of the following strategy:

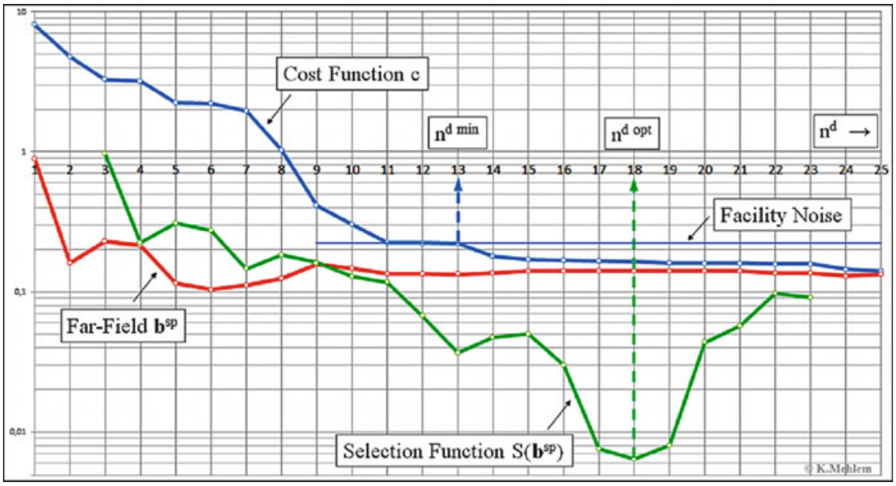


Fig. 13.12 Determination of $n^{d\ opt}$ by searching the minimum of the function S

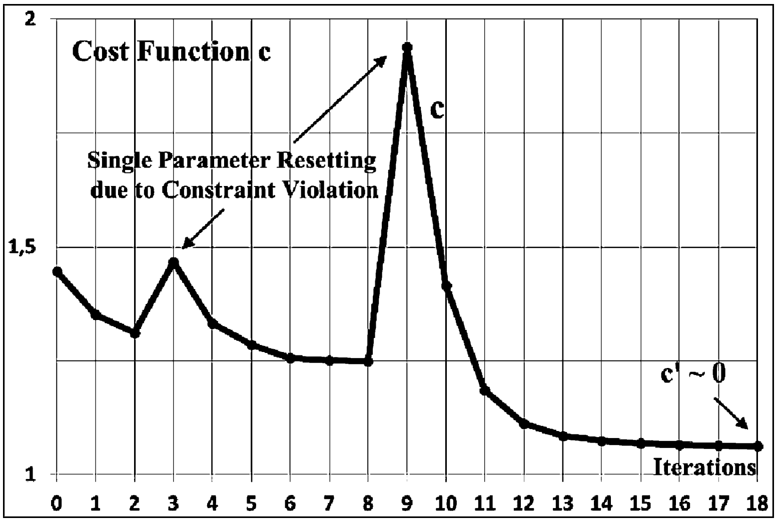


Fig. 13.13 Parameter constraint violations

When an element of a dipole position vector violates a boundary or a proximity constraint, it is reset to a constraint-compatible value. In first instance, the cost function increases, but it decreases then quite rapidly below the previous level (see peaks in Figs. 13.11 and 13.13). This strategy works very well due to a minor parameter disturbance created at a time.

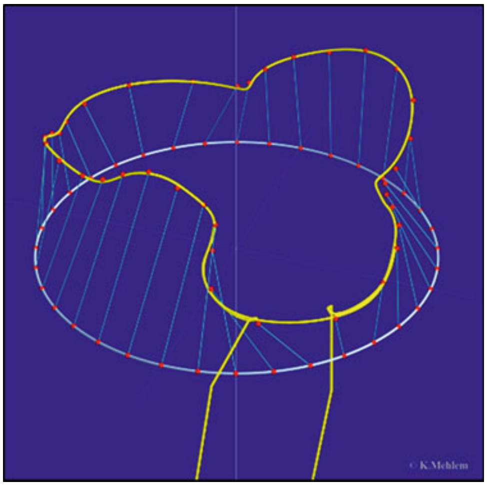


Fig. 13.14 Pathological MDM field in non-observed data gap

A drastic example of mis-modeling due to the lack of constraints is shown in Fig. 13.14, where a dipole "escaped" through a data gap without affecting the cost function.

13.3.6 *Ambiguity of MDM Solutions*

A unique MDM solution, where the dipoles represent the internal sources (Fig. 13.15, red units inside the S/C), exists only if the number of field data is very high and evenly distributed on an enclosing surface. In practice however, only a relatively low number of data with more or less uneven distributions are available (Fig. 13.16).

Consequently, an unknown number of equivalent (*ambiguous*) MDM solutions can be found which all satisfy the condition $c \leq \tau$, but where the associated far-field shows a large scattering due to MDM field differences within the data gaps, which are not reflected in the least square function c (Eq. 13.18).

Figure 13.17 shows two ambiguous MDM solutions $c \leq \tau$ in the space of two suitable parameter sub-sets $|\boldsymbol{p}_1|$ and $|\boldsymbol{p}_2|$ ($n_d = 13$).

Twenty ambiguous MDM solutions ($c \leq \tau$) are shown in Fig. 13.18. The dolomite-like pattern is a projection of consecutive cuts through the topology between two minima. Note also the scattering of the associated far-field which does not appear to be correlated with the cost function.

The question is how to cope with this general problem of ambiguous solutions which is caused by data sparsity. In order to arrive to the best possible far-field estimate we use a statistical approach (as an example we use a 13-dipole modeling

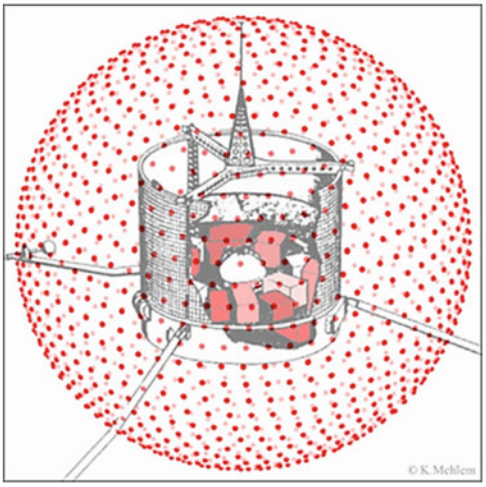


Fig. 13.15 Desirable data coverage

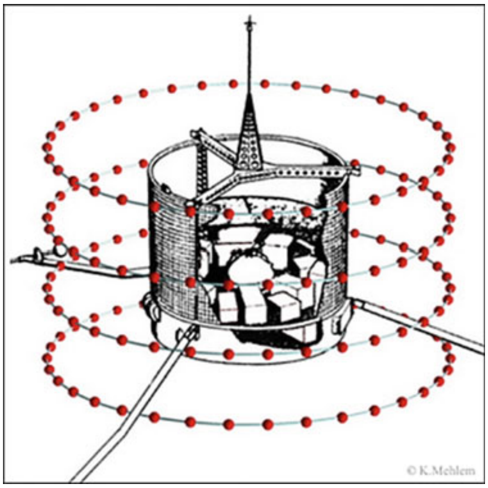


Fig. 13.16 Real data coverage

case): first, we generate n MDM solutions ($i = 1, \ldots$, n), satisfying all the data fit condition $c \leq \tau$. Then we determine the average and the standard deviation of the associated far-fields $|\mathbf{b}_i|$ and $|\mathbf{b}_i + \Delta\mathbf{b}_{i.3\sigma}|$ and $|\mathbf{b}_i\text{-}\Delta\mathbf{b}_{i.3\sigma}|$, respectively. Due to the severe condition of optimality for each MDM the averages of the far-field $|\mathbf{b}_i|_{av}$ and the averages of the standard deviations $|\mathbf{b}_i|_{av} \pm \Delta\mathbf{b}_{i.3\sigma}|_{av}$ stabilize already at $i \geq 18$ around the values $|\mathbf{b}|_{av}$ and $|\mathbf{b}|_{av} \pm |\Delta\mathbf{b}_{3\sigma}|_{av}$, respectively (see Fig. 13.19).

For further far-field calculations, for instance, for compensation tasks, we select the best MDM from the lot of MDM$_i$'s, (in this case $i = 11$) such that

$$M^{opt} = M_i\left\{(|b_i| - |b|_{av}) = \min \forall\, i \leq 20\right\} \tag{13.39}$$

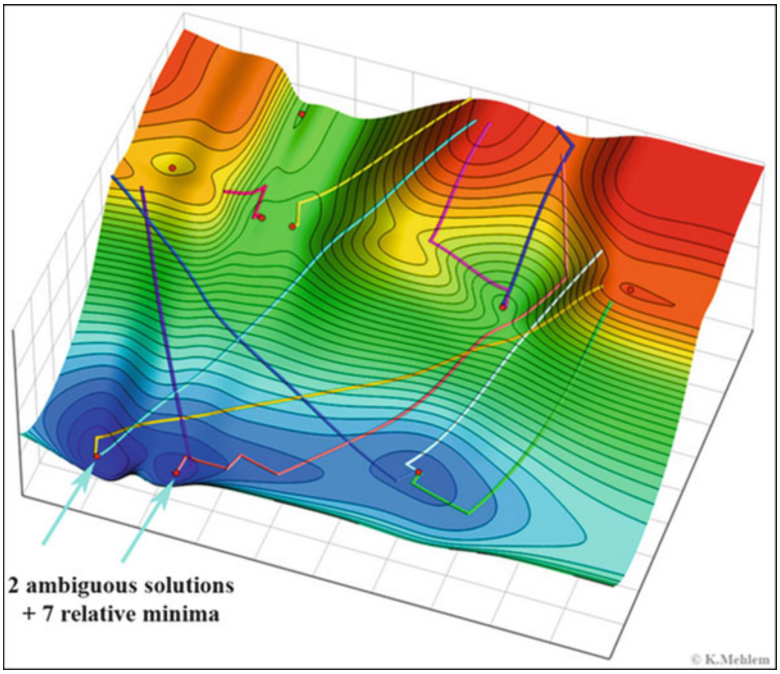


Fig. 13.17 Two ambiguous MDM solutions and a number of relative minima

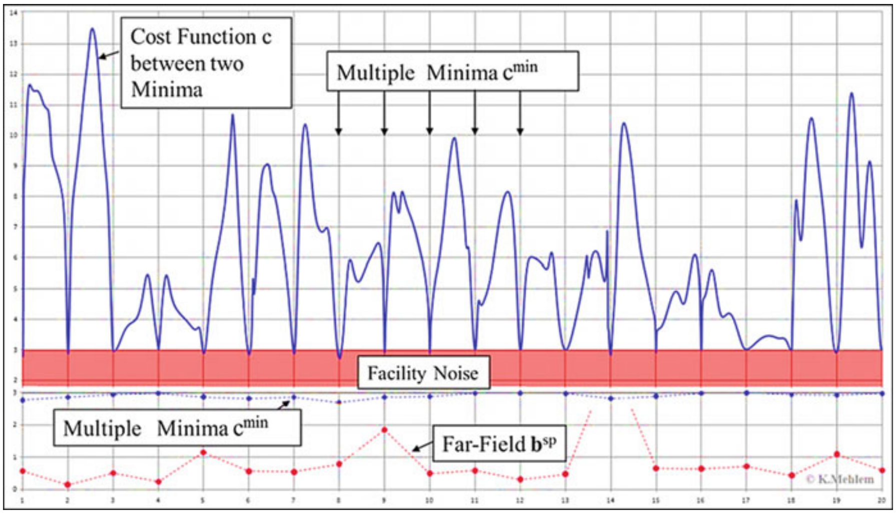


Fig. 13.18 Twenty ambiguous MDM solutions ($c \leq \tau$)

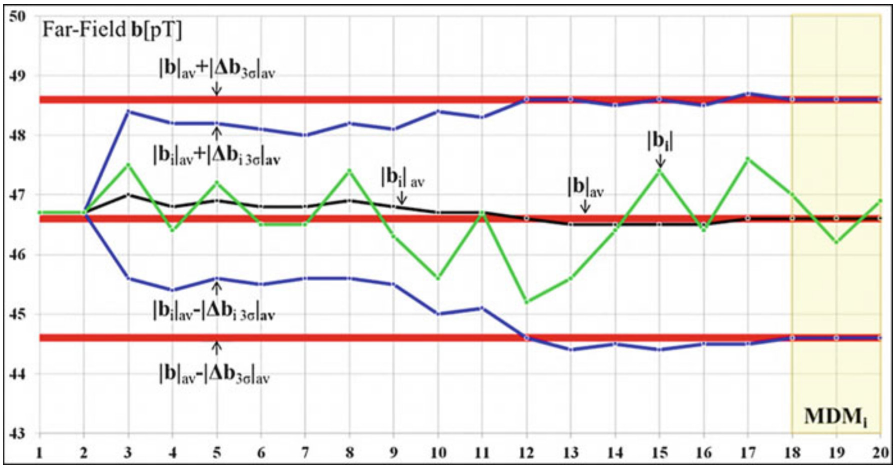


Fig. 13.19 Far-field averages $|\mathrm{b_i}|_{\mathrm{av}}$ (*gray*) and $|\mathrm{b_i} \pm \Delta \mathrm{b_{i\ 3\sigma}}|_{\mathrm{av}}$ (*blue*) vs. number of trials $n = 20$

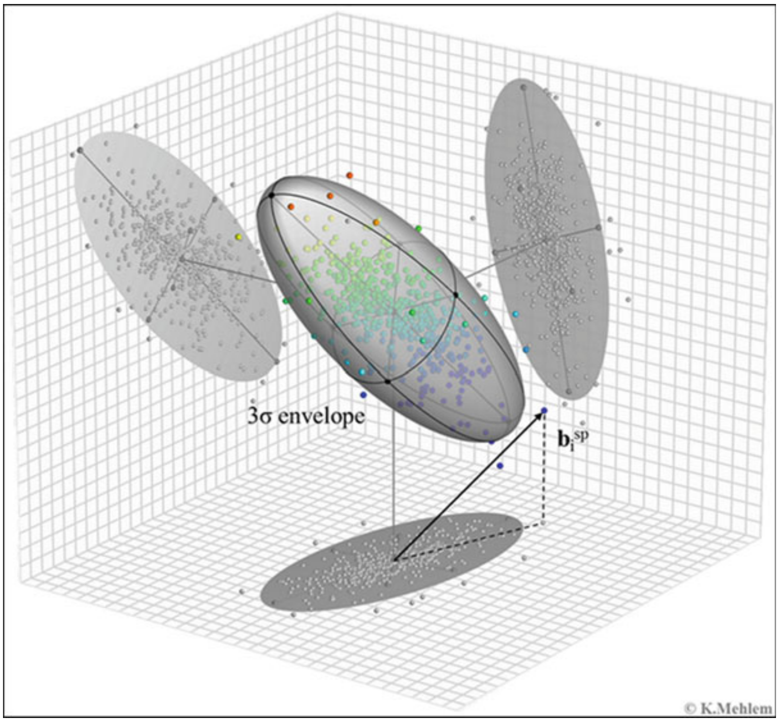


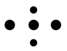
Fig. 13.20 Distribution of 200 far-field vectors associated to 200 ambiguous MDM solutions (3σ ellipsoid, aligned in average first Gauss direction)

In this way we have obtained not only an accurate estimate of the far-field $|\mathbf{b}|_{av}$ but also a measure for the test setup characteristics through the value of $|\mathbf{b} \pm \Delta\mathbf{b}_{3\sigma}|_{av}$, which represents the semimajor axis of the 3σ ellipsoid of Fig. 13.20.

Figure 13.20 shows as an example of $n = 200$ the average and the 3σ ellipsoid of the far-field **b**. The black arrow represents one of the field vectors involved. It turns out that the semimajor axis of the ellipsoid is a function of the density and of the symmetry of the data coverage (see Figs. 13.15 and 13.16). The density is of course the total number of points n^{tot}. The number of symmetrical points can be defined as

$$n^{sym} = n^{ts} \cdot \frac{n^{nb}}{n^{nb\,ref}} \tag{13.40}$$

where

- n^{nb} is the number of neighbors which have the same distance from the central point and which are separated by the same angle
- $n^{nb\ ref} = 4$ corresponds to the figure: 
- $n^{ts}$is the number of points which have at least $n^{nb} = 2$ equidistant symmetrical neighbors: ■■■

Let us define coverage index C as:

$$C = \frac{n^{sym}}{n^{tot}} = \frac{n^{ts} \cdot n^{nb}}{n^{nb\,ref} \cdot n^{tot}} \tag{13.41}$$

For the 13-dipole example which we used above for the statistics, we find the following relation between the semimajor axis $a = |\Delta\mathbf{b}^{spmax}{}_{3\sigma}|/|\Delta\mathbf{b}^{spav}|$ of the 3σ ellipsoid and the coverage index C (Fig. 13.21):

$$a = \frac{1}{10 \cdot \sqrt{C}} = 10^{-1}\sqrt{\frac{n^{nb\,ref.}}{n^{nb}} \cdot \frac{n^{tot}}{n^{ts}}} \tag{13.42}$$

Equation (13.42) represents a useful estimation of the modeling errors to be expected when a test setup is chosen in form of n^{nb} and n^{ts}. In order to keep the errors to a minimum it is therefore of utmost importance *to maximize the coverage index C* by using the maximum possible number of probes and by distributing them homogeneously on a sphere centered on the test article.

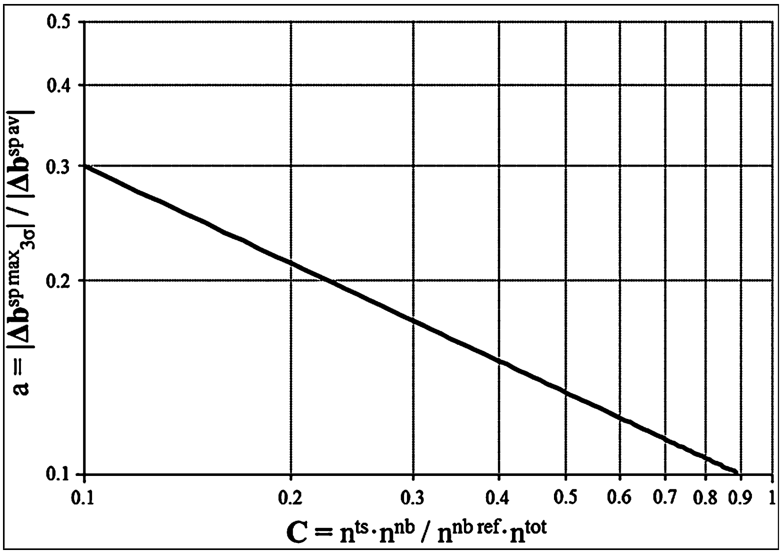


Fig. 13.21 Semimajor axis a vs. coverage index C

13.3.7 Synthetic Spacecraft MDM

Definitions:

x^t, y^t, z^t	= coordinate system of the test facility (TCS)
x^u, y^u, z^u	= coordinate system of the unit (UCS)
x^s, y^s, z^s	= coordinate system of the spacecraft (SCS)
t^u	= vector from the unit's reference point to test facility's origin
u^s	= vector from the spacecraft's origin to the unit's reference point
p^t	= dipole position vector described in SCS
p^t	= dipole position vector described in the TCS
m^s	= dipole moment vector described in the SCS
m^t	= dipole moment vector described in the facility's coordinate system
R^{us}	= rotation matrix leading from TCS to SCS
R^{tu}	= rotation matrix leading from the TCS to UCS

The field measurements and thus the MDM of a unit are related to the TCS. In order to build up a synthetic spacecraft model, the position and moment vectors of each dipole of a unit have to be transformed from the TCS to the SCS.

This is done in two steps. First, the transformation from TCS to UCS is obtained by a translation and by a rotation (Fig. 13.22):

$$p^u = R^{tu} \cdot p^t + t^u \tag{13.43}$$

$$m^u = R^{tu} \cdot m^t \tag{13.44}$$

Then the transformation from UCS to SCS is obtained by:

$$p^s = R^{us} \cdot p^u + u^s \tag{13.45}$$

$$m^s = R^{us} \cdot m^u \tag{13.46}$$

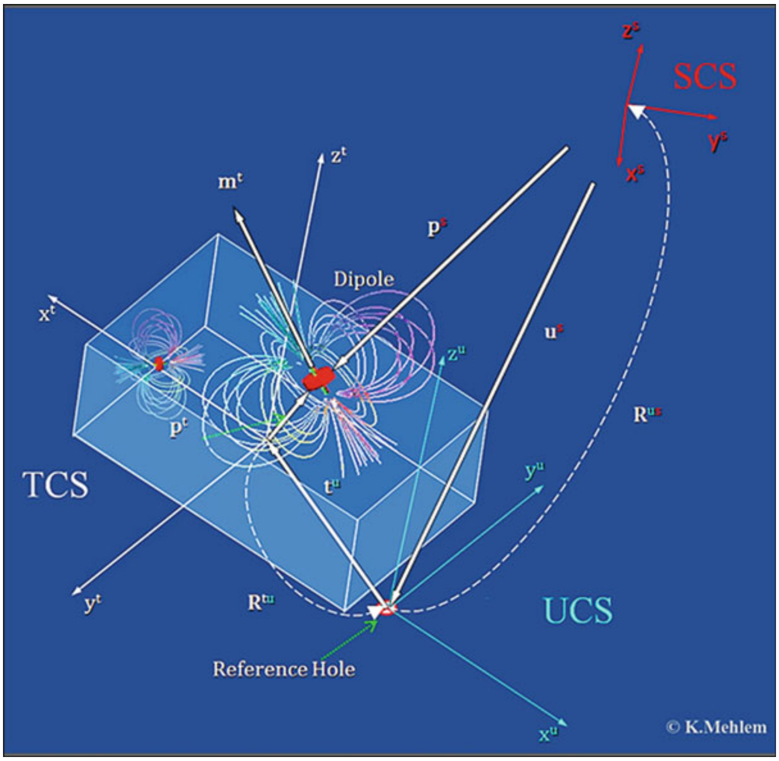


Fig. 13.22 Coordinate transformations from test to spacecraft coordinate system

By inserting Eq. (13.43) into Eq. (13.45) we obtain for the positions of a unit:

$$p^s = u^s + R^{us} \cdot t^u + R^{us} \cdot R^{tu} \cdot p^t \tag{13.47}$$

By inserting Eq. (13.44) into Eq. (13.46) we obtain for the moments of a unit:

$$m^s = R^{us} \cdot R^{tu} \cdot m^t \tag{13.48}$$

If we suppose that the ith unit contains $\mathrm{n}_\mathrm{i}^\mathrm{d}$ dipoles, we have the ith unit MDM in SCS (Eqs. 13.45 and 13.46):

$$M_i^s = \begin{bmatrix} p_{li}^s & m_{li}^s \\ \vdots & \vdots \\ p_{ki}^s & m_{ki}^s \\ \vdots & \vdots \\ p_{n_i i}^{s_d} & m_{n_i i}^{s_d} \end{bmatrix} \tag{13.49}$$

The total synthetic spacecraft model is then

$$M^s = \begin{bmatrix} M_l^s \\ \vdots \\ M_i^s \\ \vdots \\ M_n^{s_u} \end{bmatrix} \tag{13.50}$$

Note however, that the accuracy of the synthetic model depends on possible inducted moments of units when they are integrated in the spacecraft and exposed to external fields. Therefore a very challenging problem arises in magnetic cleanliness when soft-magnetic materials are used. It should definitely be avoided wherever possible.

13.3.8 Far-Field Compensation

The ultimate goal of magnetic cleanliness is to insure that the cleanliness specification for a given CSP is met. This can also *be the case* for several points. Should the far-fields generated by the optimal MDM of the spacecraft $M^{s/c\ opt}$ (Eq. 13.39) exceed the specifications, one can derive a system of compensation magnets M^m *which, depending on the number of magnets used,* reduce *or even zero* the far-field vectors at multiple points.

Definition:

n^d = number of dipoles of the spacecraft
n^{sp} = number of CSPs
n^m = number of magnets
r^{sp} = $3n^{sp} \times 1$ array with n^{sp} position vectors r_j^{sp} of the CSP
$\boldsymbol{p}^{s/c}$ = $3n^d \times 1$ array with n^d dipole position vectors $r_j^{s/c}$ of the spacecraft
$\boldsymbol{m}^{s/c}$ = $3n^d \times 1$ array with n^d dipole moment vectors $m_j^{s/c}$ of the spacecraft
$\boldsymbol{p}^m$ = $3n^m \times 1$ array with n^m position vectors r_k^m of compensation magnets
$\boldsymbol{m}^m$ = $3n^m \times 1$ array with n^m moment vectors m_k^m of compensation magnets

The optimal moments of a set of compensation magnets, which reduce the far-field vectors, are obtained by Eq. (13.13):

$$m^{m\ opt} = -D^{m+} D^{s/c} \cdot m^{s/c} \tag{13.51}$$

The $\boldsymbol{D}^m$ contains the 3×3 matrices D_{ik}^m, (i = 1,. . ., n^{sp}, k = 1,. . ., n^m):

$$D_{ik}^m = \frac{\mu_0}{4\pi} \cdot \frac{3 \cdot \left[\left(r_i^{sp} - r_k^m\right) \cdot \left(r_i^{sp} - r_k^m\right)^T - \left|\left(r_i^{sp} - r_k^m\right)\right|^2 \cdot I\right]}{\left|\left(r_i^{sp} - r_k^m\right)\right|^5} \tag{13.52}$$

and $\boldsymbol{D}^{s/c}$ contains the 3×3 matrices $D_{ij}^{s/c}$(i = 1,. . ., n^{sp}, j = 1,. . ., n^d):

$$D_{ij}^{s/c} = \frac{\mu_0}{4\pi} \cdot \frac{3 \cdot \left[\left(r_i^{sp} - r_j^{s/c}\right) \cdot \left(r_i^{sp} - r_j^{s/c}\right)^T - \left|\left(r_i^{sp} - r_j^{s/c}\right)\right|^2 \cdot I\right]}{\left|r_i^{sp} - r_j^{s/c}\right|^5} \tag{13.53}$$

When considering several far-field points and several magnets, Eq. (13.51) leads to three types of solutions (see also Figs. 13.23 and 13.24):

1. $n^m = n^{sp} \rightarrow b^{sp} = 0 \;\; \forall r^m$: exact solution $\boldsymbol{b}^{sp} = 0$ for all magnet positions r_k^m chosen
2. $n^m < n^{sp} \rightarrow b^{sp\,T} \cdot b^{sp} = \min \forall r^m \neq r^{m\ opt}$: least square solution for all non-optimal positions r_k^m chosen
3. $n^m < n^{sp} \rightarrow b^{sp} = 0 \; \forall \, r^m = r^{m\ opt}$: exact solutions $\boldsymbol{b}^{sp} = 0$ for all optimized magnet positions $r_k^m = r_k^{m\ opt}$

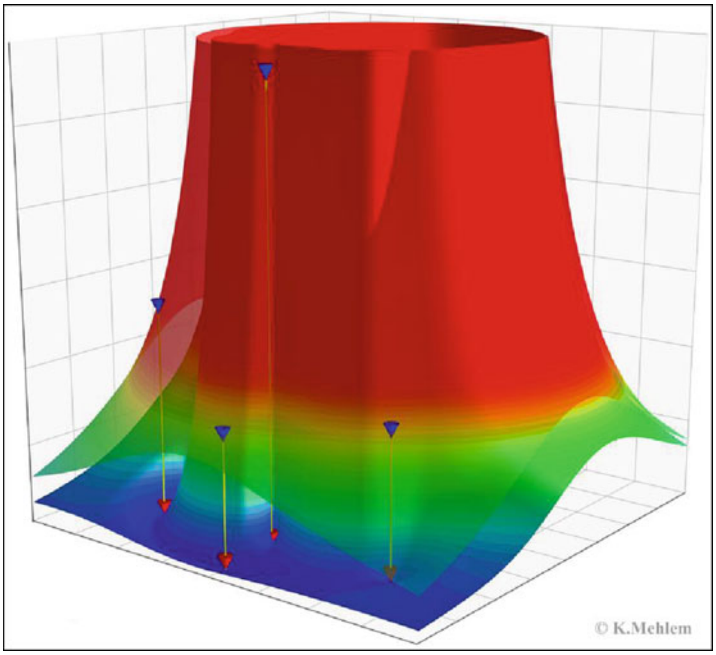


Fig. 13.23 Example of an exact 4-point field compensation by use of four magnets (type 1)

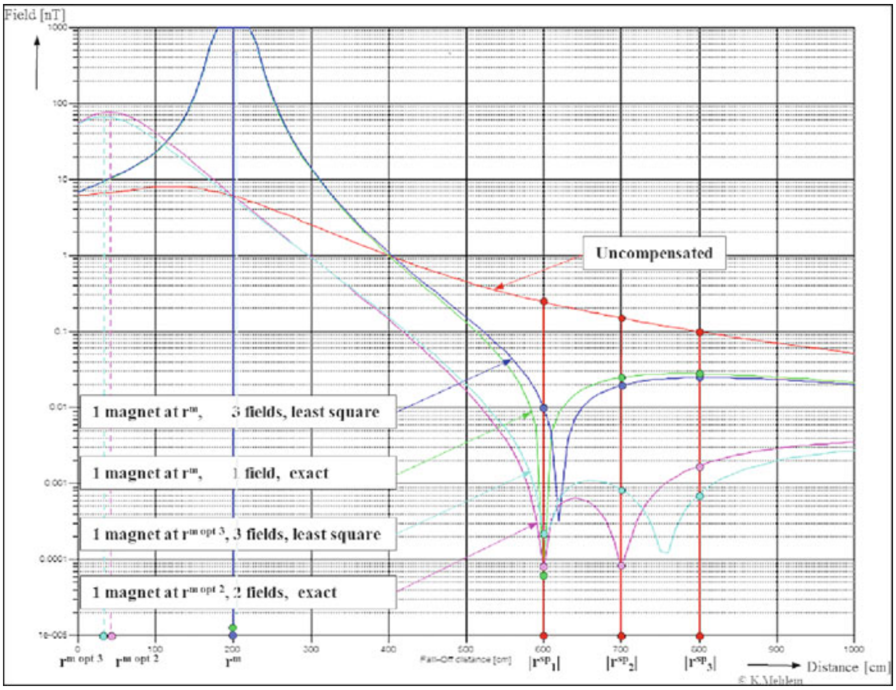


Fig. 13.24 Example of four possibilities when using one magnet for one to three far-field points

13.3.9 MDM Identification from Field Gradients

13.3.9.1 Formulation

The field gradient has the important property to ignore all external perturbing fields which are constant over the length of a gradiometer sensor element. This fact can under certain precautions be used when the magnetic tests have to be done without the help of a coil facility. For this reason we have formulated the MDM method also for field gradients. The full magnetic field gradient tensor is [10]

$$\Gamma = \begin{bmatrix} \frac{\partial b_x}{\partial x} & \frac{\partial b_x}{\partial y} & \frac{\partial b_x}{\partial z} \\ \frac{\partial b_y}{\partial x} & \frac{\partial b_y}{\partial y} & \frac{\partial b_y}{\partial z} \\ \frac{\partial b_z}{\partial x} & \frac{\partial b_z}{\partial y} & \frac{\partial b_z}{\partial z} \end{bmatrix} \tag{13.54}$$

The magnetic potential, in absence of an electrical current, has the following properties:

$$\nabla \cdot b = 0 \quad \rightarrow \quad \frac{\partial b_x}{\partial x} + \frac{\partial b_y}{\partial y} + \frac{\partial b_z}{\partial z} = 0 \tag{13.55}$$

$$\nabla \times b = 0 \rightarrow \frac{\partial b_z}{\partial y} = \frac{\partial b_y}{\partial z}; \rightarrow \frac{\partial b_x}{\partial x} = \frac{\partial b_z}{\partial x}; \rightarrow \frac{\partial b_x}{\partial y} = \frac{\partial b_y}{\partial x} \tag{13.56}$$

According to Eqs. (13.55) and (13.56) only five elements of the tensor Γ are independent:

$$\Gamma = \begin{bmatrix} \frac{\partial b_x}{\partial x} & \frac{\partial b_y}{\partial x} & \frac{\partial b_z}{\partial x} \\ \frac{\partial b_y}{\partial x} & \frac{\partial b_y}{\partial y} & \frac{\partial b_z}{\partial y} \\ \frac{\partial b_z}{\partial x} & \frac{\partial b_z}{\partial y} & -\frac{\partial b_x}{\partial x} - \frac{\partial b_y}{\partial y} \end{bmatrix} \tag{13.57}$$

Hence, the field gradient can be written as a 5 $\times$ 1 array **g** (not a vector):

$$g^T = \begin{bmatrix} \frac{\partial b_x}{\partial x} & \frac{\partial b_y}{\partial x} & \frac{\partial b_z}{\partial x} & \frac{\partial b_y}{\partial y} & \frac{\partial b_y}{\partial z} \end{bmatrix} \tag{13.58}$$

From here on we use the indices 1, 2, 3 instead of x, y, z. In analogy to Eq. (13.1) we can write for g_i

$$g_i = \sum_{k=1}^{n^d} \left[\frac{\partial D_{ik}}{\partial r_i^s} \right] \cdot m_k^d = \sum_{k=1}^{n^d} G_{ik} \cdot m_k^d \tag{13.59}$$

with the 5 $\times$ 3 matrix

$$G_{ik} = \begin{bmatrix} \frac{\partial D_{11\,ik}}{\partial r^s_{1\,i}} & \frac{\partial D_{11\,ik}}{\partial r^s_{2\,i}} & \frac{\partial D_{11\,ik}}{\partial r^s_{3\,i}} \\ \frac{\partial D_{21\,ik}}{\partial r^s_{1\,i}} & \frac{\partial D_{21\,ik}}{\partial r^s_{2\,i}} & \frac{\partial D_{21\,ik}}{\partial r^s_{3\,i}} \\ \frac{\partial D_{31\,ik}}{\partial r^s_{1\,i}} & \frac{\partial D_{31\,ik}}{\partial r^s_{2\,i}} & \frac{\partial D_{31\,ik}}{\partial r^s_{3\,i}} \\ \frac{\partial D_{22\,ik}}{\partial r^s_{1\,i}} & \frac{\partial D_{22\,ik}}{\partial r^s_{2\,i}} & \frac{\partial D_{22\,ik}}{\partial r^s_{3\,i}} \\ \frac{\partial D_{23\,ik}}{\partial r^s_{1\,i}} & \frac{\partial D_{23\,ik}}{\partial r^s_{2\,i}} & \frac{\partial D_{23\,ik}}{\partial r^s_{3\,i}} \end{bmatrix}_{\mathrm{i=1,\ldots,n^s,k=1,\ldots,n^d}} \tag{13.60}$$

By collecting all vectors r^d_k, m^d_k, and all 5 $\times$ 1 arrays g_i and all 5 $\times$ 3 matrices G_{ik} in the following arrays

$$\begin{aligned} p^T &= \left[r^d_1 \ldots r^d_k \ldots r^d_{n^d}\right] \\ m^T &= \left[m^d_1 \ldots m^d_k \ldots m^d_{n^d}\right] \\ g^T &= [b_1 \ldots b_i \ldots b_{n^s}] \\ G^T &= [G_{11} \ldots G_{ik} \ldots G_{n^s n^d}] \end{aligned} \tag{13.61}$$

We can write Eq. (13.59) simply as:

$$g = G \cdot m \tag{13.62}$$

In analogy to Eq. (13.13) the optimal moments are given by:

$$m^{opt}(p) = G(p)^+ \cdot g^m \tag{13.63}$$

The associated “calculated” or MDM field gradient is then:

$$g^c = G.m^{opt} = G^+ \cdot g^m \tag{13.64}$$

In analogy to Eq. (13.15) the field gradient residues are:

$$\varepsilon = \left(I - G \cdot \left[G^T \cdot G\right]^{-1} \cdot G^T\right) \cdot g^m \tag{13.65}$$

If Eq. (13.31) is fulfilled we have found the optimal MDM:

$$M^{opt} = \left[p^{i+1} \quad m^{d\ opt}\left(p^{i=1}\right) = \left[G^T \cdot G\right]^{-1} \cdot G^T \cdot g^m\right] \tag{13.66}$$

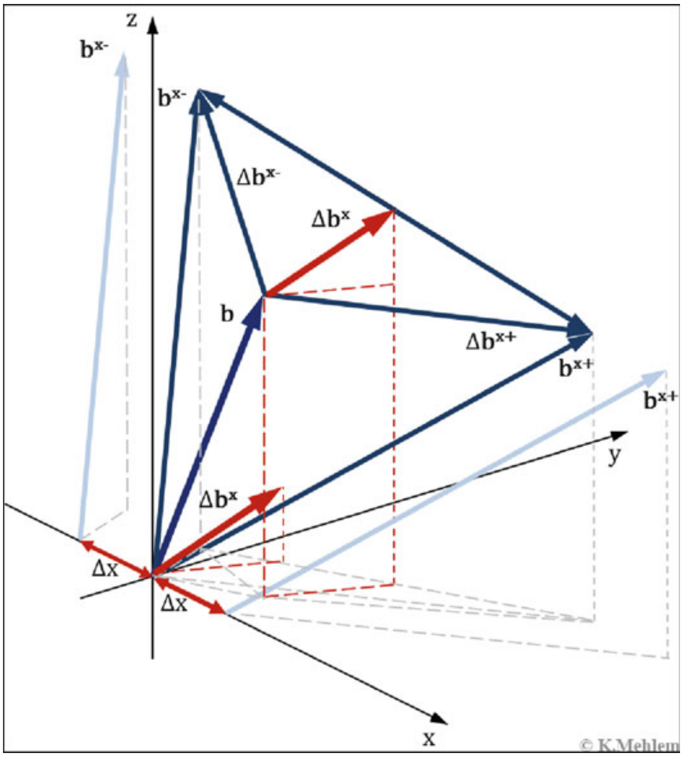


Fig. 13.25 Double-sided perturbation in x-direction of the probe position vector r^s

All the MDM issues described above for field measurements, like number of dipoles, etc., apply of course in the same way for field gradient measurements.

13.3.9.2 Field Gradient Tensor as Vector

Whereas we are used to the field as a vector, it is rather difficult to visualize the field gradient tensor. The following approach is a possible way to define the tensor as a vector. The double-sided small variation of the probe position in x-direction entails the field vector Δb^x (Fig. 13.25):

$$\Delta b^x = \frac{1}{2}(b^{x-} + b^{x+}) \tag{13.67}$$

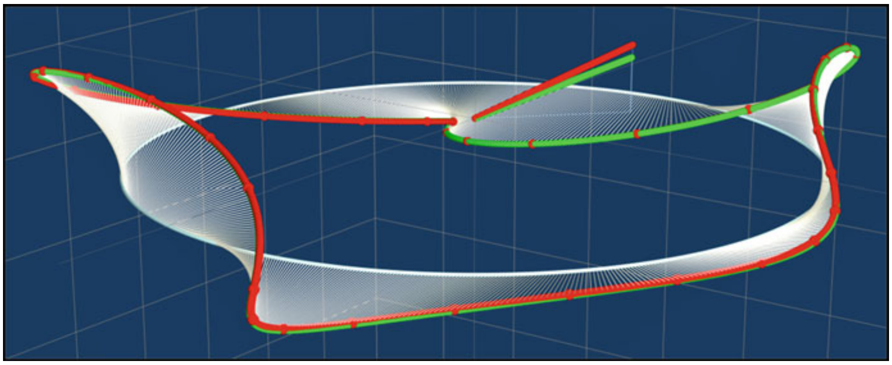


Fig. 13.26 Pseudo field gradient vectors $\boldsymbol{\gamma}_i$ (*white lines* with colored *top-lines*); global moment vectors, (*green = reference, red = recovered* from gradient data)

with

$$\begin{bmatrix} b^{x-} = b & (r^{s-} = r^s - \Delta r \cdot e_x) \\ b^{x+1} = b & (r^{s+} = r^s + \Delta r \cdot e_x) \end{bmatrix} \tag{13.68}$$

When repeating the variation in the y- and z-direction we obtain the field vectors Δb^y and Δb^z, respectively. With a variation Δr $\ll$ 1 we obtain the pseudo field gradient vector γ by the following vector sum:

$$\gamma = \frac{\Delta b^x}{\Delta r} + \frac{\Delta b^y}{\Delta r} + \frac{\Delta b^z}{\Delta r} \tag{13.69}$$

Figure 13.26 shows rotational pseudo field gradient vectors $\boldsymbol{\gamma}_j$ (white lines connected by colored topcurves). Each vector represents the local field change due to a single variation of the probe position in +x-, +y-, and +z-direction.

13.3.9.3 Example of Field Gradient Modeling by MDM

In the absence of real rotational field gradient data we had to simulate them. The following points explain what has been done (see also Figs. 13.25 and 13.28):

1. A reference 3-dipolemodel was chosen with a global moment of $|\boldsymbol{m}^{\text{ref}}| = 104.4$ mA m^2 (green) $\rightarrow$
2. A dipole at a distance of 2 m was added to the model in order to simulate at the center of the turntable a severe 100, nT perturbation like from a steel structure. Also a second dipole with a random moment located at an extreme distance was added in order to simulate worst case daily Earth field variations of $\pm$100 nT. Thus, the rotational *field gradient* measurements were simulated by use of a 5-dipole model.

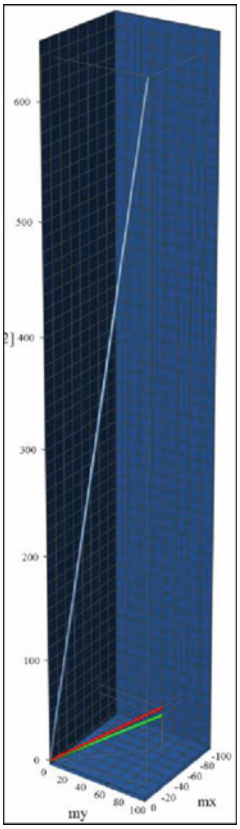


Fig. 13.27 Global moment vectors, (*green* = *reference*, *red* = *recovered* from gradient data, *blue* = *recovered* from field data)

3. From the field gradient data a 4-dipole-model with a global moment of $|\boldsymbol{m}_4|$ = 107.8 mA m^2 was obtained. Despite the strong perturbations the residual moment was only 8.0 mA m^2 (7.7 %) (red) →
4. In contrast, from the perturbed field data a 10-dipole model was obtained which had a huge moment of $|\boldsymbol{m}_{10}|$ = 639,0 mA m^2 (blue) → (Fig. 13.27)

Figure 13.28 shows the perturbations of the rotational *field* data (blue) and their propagation (red) into the *field gradient* data.

We have varied the distance (point.2 above) between 2 and 10 m (see Fig. 13.29). The error for both the global moment and for the far-field starts with a relatively low value of about 10 % at 2 m, and it decreases quite rapidly, vanishing completely after 10 m.

The present example demonstrates that MDM is perfectly suited also for *field gradient* modeling tasks which may become very useful, in particular when tests have to be made at places where no coil systems are available. Still, it has to be

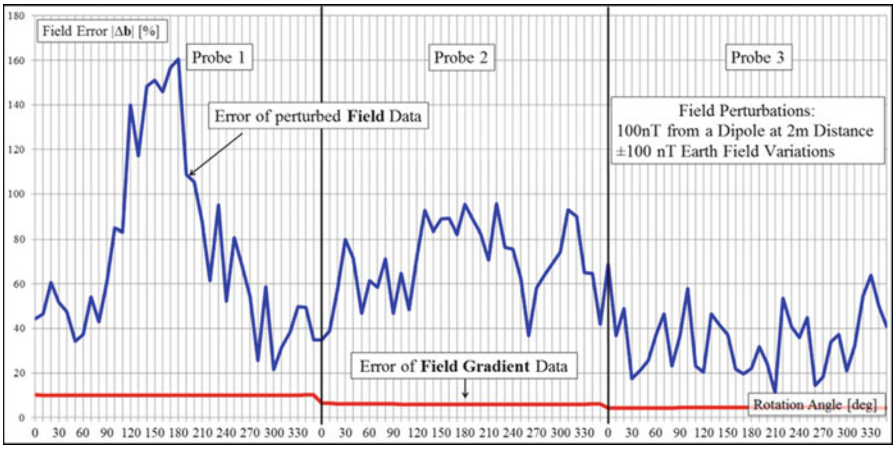


Fig. 13.28 *Field* (*blue*) and *field gradient* (*red*) data affected by the same perturbations

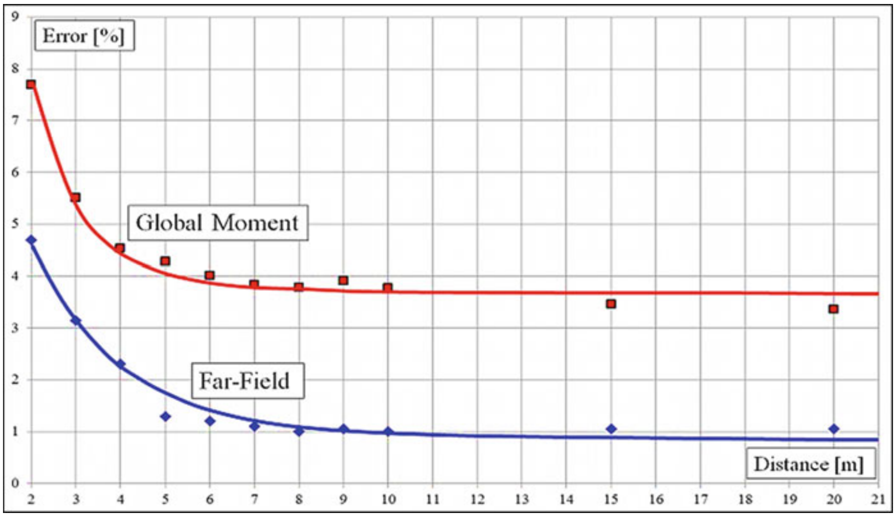


Fig. 13.29 Error of the global moment and the far-field vs. distance of perturbing dipole

reminded that field gradient data have a lower SNR than field data, and that the use outside a coil system is restricted to test articles which do not contain any significant amount of softmagnetic material in which the external magnetic field could induce disturbing moments. The modeling of induction is very difficult.

13.4 Applications

13.4.1 Giotto

The famous European spacecraft Giotto, which passed close to the comet Halley in 1986, had two magnetometers mounted on one leg of a so-called antenna tri-pot (Fig. 13.30). A pair of strongly magnetic *Travelling Wave Tubes* (TWTs) located on the upper platform and at a distance of only 1,47 m from the upper magnetometer, generated a perturbing field of 39.3 nT which would have severely affected the magnetometer readings during flight (Table 13.1). A typical pair of TWTs (courtesy of Thales) is shown as insert in Fig. 13.31. This pair had therefore to be compensated by use of a magnet. In a first step three candidate TWT combinations (composed of two flight models and one spare) were mapped in the CNES magnetic test facility and precise MDMs were derived (see Sects. 3.2, 3.3 and Table 13.1). In a second step the fields at MAG-1 for all three possible TWT combinations were calculated (Eq. 13.1). In a third step the associated optimal compensation magnet moment vectors were determined (Eq. 13.5).

The Combination Nr. 1 (Table 13.1) was finally chosen as flight hardware. A 6 cm long magnet of 1,139 mA m^2 was fabricated and then installed on a bracket

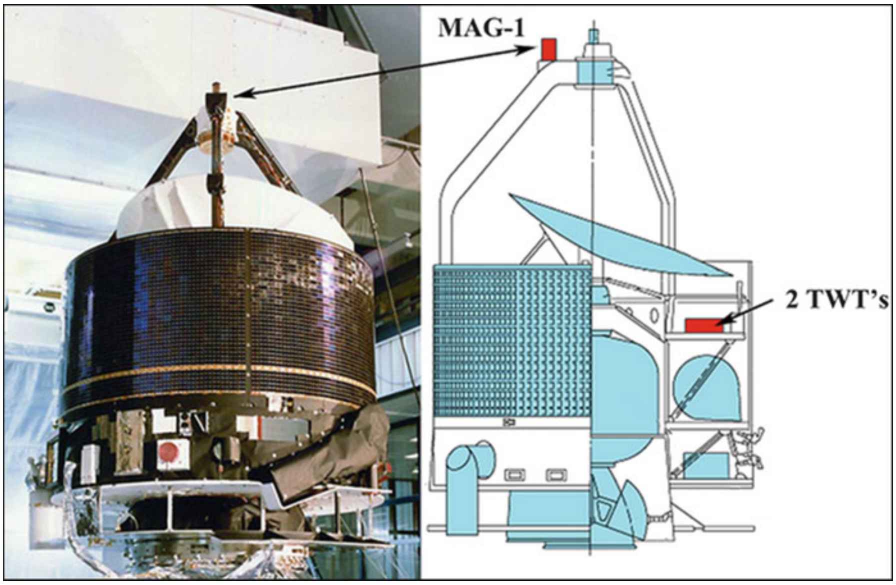
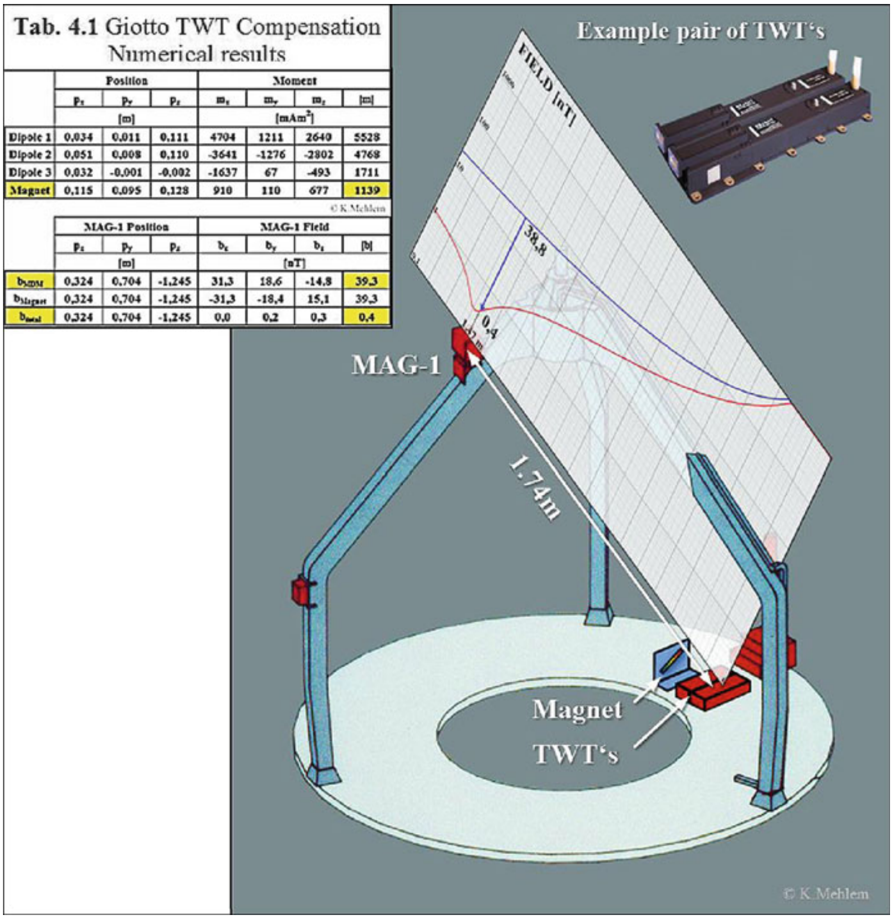


Fig. 13.30 Giotto with the magnetometer (MAG-1) on the antenna tri-pod, and 2 TWTs (images: ESA)

Table 13.1 Giotto TWT compensation numerical results

	Position			Moment			
	P_x (m)	P_y (m)	P_z (m)	m_x (mA m^2)	m_y (mA m^2)	m_z (mA m^2)	$\lvert m\rvert$ (mA m^2)
Dipole 1	0.034	0.011	0.111	4,704	1,211	2,640	5,528
Dipole 2	0.051	−0.008	0.110	−3,641	−1,276	−2,802	4,768
Dipole 3	0.032	−0.001	−0.002	−1,637	67	−493	1,711
Magnet	0.115	0.095	0.128	910	110	677	1,139

	MAG-1 position			MAG-1 field			
	P_x (m)	P_y (m)	P_z (m)	b_x (nT)	b_y (nT)	b_z (nT)	$\lvert b\rvert$ (nT)
b_{MDM}	0.324	0.704	−1.245	31.3	18.6	−14.8	39.3
b_{Magnet}	0.324	0.704	−1.245	−31.3	−18.4	15.1	39.3
b_{Total}	0.324	0.704	−1.245	0.0	0.2	0.3	0.4



Tab. 4.1 Giotto TWT Compensation
Numerical results

	Position			Moment			
	p_x	p_y	p_z	m_x	m_y	m_z	$\lvert m\rvert$
	[m]			[mAm^2]			
Dipole 1	0,034	0,011	0,111	4704	1211	2640	5528
Dipole 2	0,051	0,008	0,110	-3641	-1276	-2802	4768
Dipole 3	0,032	-0,001	-0,002	-1637	67	-493	1711
Magnet	0,115	0,095	0,128	910	110	677	1139

© K.Mehlem

	MAG-1 Position			MAG-1 Field			
	p_x	p_y	p_z	b_x	b_y	b_z	$\lvert b\rvert$
	[m]			[nT]			
b_{MDM}	0,324	0,704	-1,245	31,3	18,6	-14,8	39,3
b_{Magnet}	0,324	0,704	-1,245	-31,3	-18,4	15,1	39,3
b_{total}	0,324	0,704	-1,245	0,0	0,2	0,3	0,4

Fig. 13.31 Giotto upper platform, uncompensated (*blue*) and compensated (*red*) fall-off fields of the TWTs (images: ESA, Thales + K. Mehlem)

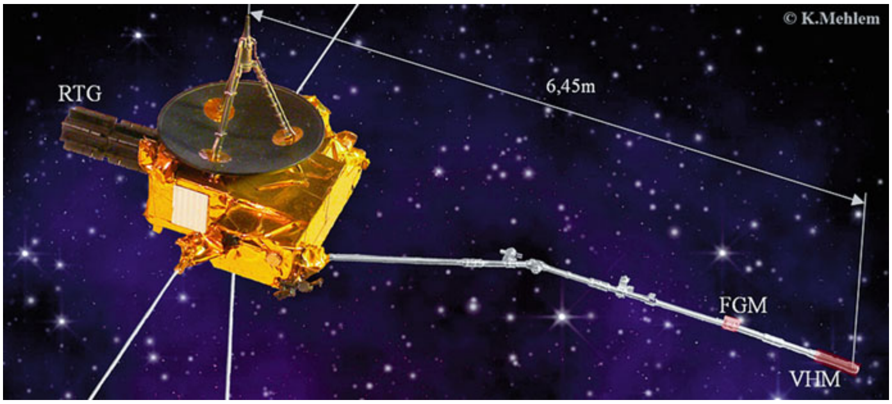


Fig. 13.32 Ulysses spacecraft with the fluxgate magnetometer (FGM), the vector helium magnetometer (VHM) and radioisotope thermoelectric power generator (RTG)

next to Comb.1 (Fig. 13.31). The composite was then mapped and modeled again, confirming the predictions very accurately. Since the distance between the TWTs and MAG-1 was only 1.47 m, we had the rare opportunity to check the compensation effect directly by a measurement at the equivalent MAG-1 location. The field readings in all three axes were quasi zero, so that the facility engineer first thought that he had forgotten to switch on the facility magnetometer. This test was an excellent validation of the MDM approach.

13.4.2 Ulysses

After 19 years of spectacular discoveries the Ulysses spacecraft travels still around the sun on a polar orbit. It carries a fluxgate magnetometer (FGM) and a vector helium magnetometer (VHM) at a distance of 6.45 m from the spacecraft center (Fig. 13.32). The cleanliness specification of 0.1 nT at the VHM location, was the strongest one ever imposed.

The modeling task was made more difficult than usual because the spacecraft could not be rotated with its boom extended and because the radioisotope thermoelectric power generator (RTG) of NASA could for evident reasons not be present during the magnetic test at IABG, Germany.

The RTG was tested separately at EG&G in Miamisburg, Ohio, USA [11], prior to the system test at IABG. Since the RTG was for safety reasons not allowed to be moved to a coil facility, the tests had to be performed in the unshielded magnetic environment of the EG&G plant. Even after careful calibration of the probes the field measurements were plagued by field offsets which did not allow a satisfactory modeling.

As mentioned before the property of the magnetic potential in absence of an electric current is $\nabla \cdot b = 0$ (Eq. 13.55). This means that the integral of the tangential field on a closed line around the test article must be zero. The deviation from zero corresponds to an external field which then can be subtracted from the tangential measurement data. Thanks to the gimbaled RTG fixture on the turn table, it was possible to map the RTG in two orthogonal planes. The tangential elements of the measured field vectors were then used to derive an accurate MDM, and the associated field vector at the Ulysses VHM location was calculated.

Thereupon, a compensation magnet was determined, manufactured on-site, and fixed with a bracket on one fin of the RTG. Further mapping and modeling with the magnet installed confirmed the compensation effect accurately.

A pair of highly magnetic TWTs, located under the antenna dish, had been compensated prior to integration (similar to Fig. 13.31).

Since the spacecraft with extended boom (Fig. 13.32) was too large for rotational measurements, it had to be tested in two separate modes.

First, the spacecraft was tested in the rotational mode with the boom in the stowed configuration. Figure 13.33 shows the spacecraft, together with measured near-field vectors (red dots, 10° separation) and the MDM near-field (white vectors connected by a yellow curve, 1° separation).

Some significant field warping, which was quite a challenge for the modeling, is visible on the right side of the figure. It was caused by two magnetic experiments (URAP and GRB) which were fixed on the boom close to the boom hinge.

Since the spacecraft was magnetically quite clean (no RTG, TWTs compensated), an acceptable SNR could only be obtained close to the spacecraft body. This meant that the URAP experiment unit passed very closely (40 cm) by the facility probes during rotation.

The URAP MDM had to be identified separately by use of a so-called linear fall-off scan of the extended boom (Fig. 13.34, insert). It was then transformed into the stowed boom configuration and subtracted from the spacecraft MDM (boom stowed). Then it was added to the spacecraft MDM in extended configuration. Finally, the MDM of the RTG was also added. The total S/C MDM contained 35 dipoles [12]. The fact that the TWTs were compensated w.r.t. the VHM location can be seen in Fig. 13.33: The field in the direction of the extended boom (y) is quite lower than the field in the opposite direction (−y).

Figures 13.34 and 13.35 show the spacecraft field with the RTG and the TWTs uncompensated and compensated (918 and 76 pT at the FGM location and 566 pT and 45 pT at the VHM location, respectively).

The total S/C MDM predicted a field at the VHM location of 45 pT. The following appreciation by mission scientists [13] reflects the success of the cleanliness effort: “Using magnetic mapping and modeling and appropriate compensation, the background field of the spacecraft at these locations was determined, prior to launch, to be approximately 30 pT and 50 pT, respectively; this makes the Ulysses spacecraft probably the magnetically cleanest interplanetary probe ever flown… Both magnetic mapping and modeling indicate the unparalleled cleanliness of the spacecraft, confirmed in flight.”

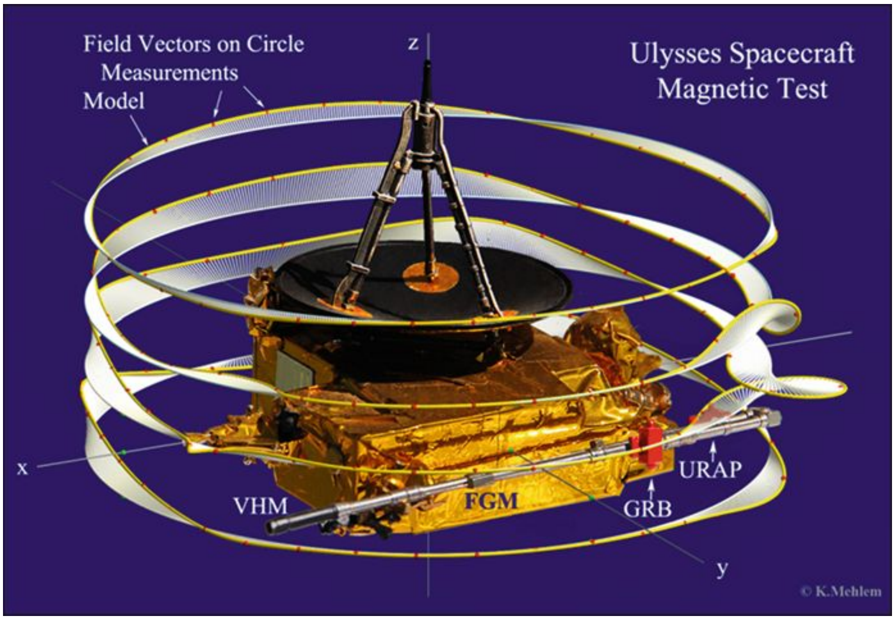


Fig. 13.33 Ulysses rotational system test with complex field patterns generated by URAP

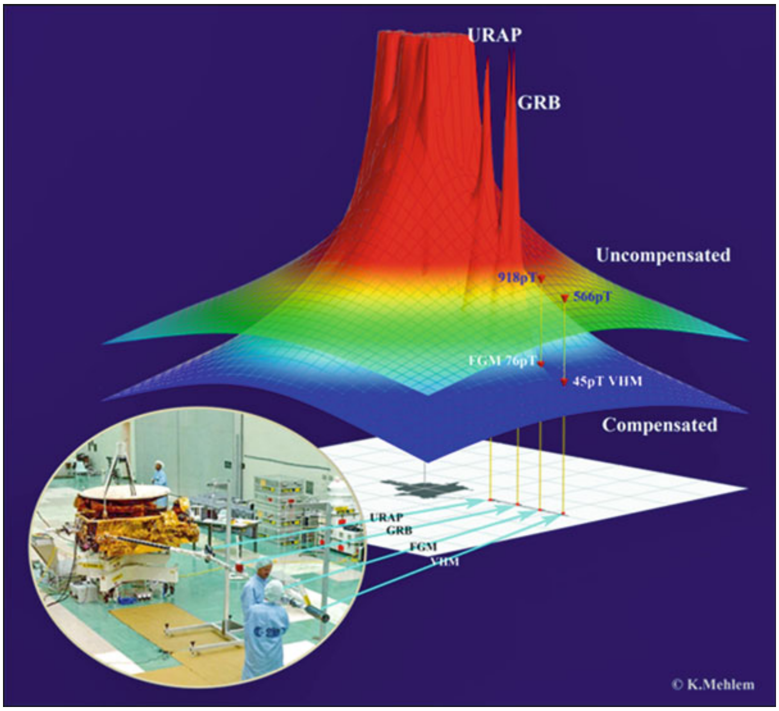


Fig. 13.34 Spacecraft fall-off field |**b**| (RTG and TWTs uncompensated and compensated) overlay on the x,y-plane containing the VHM location (insert: ESA)

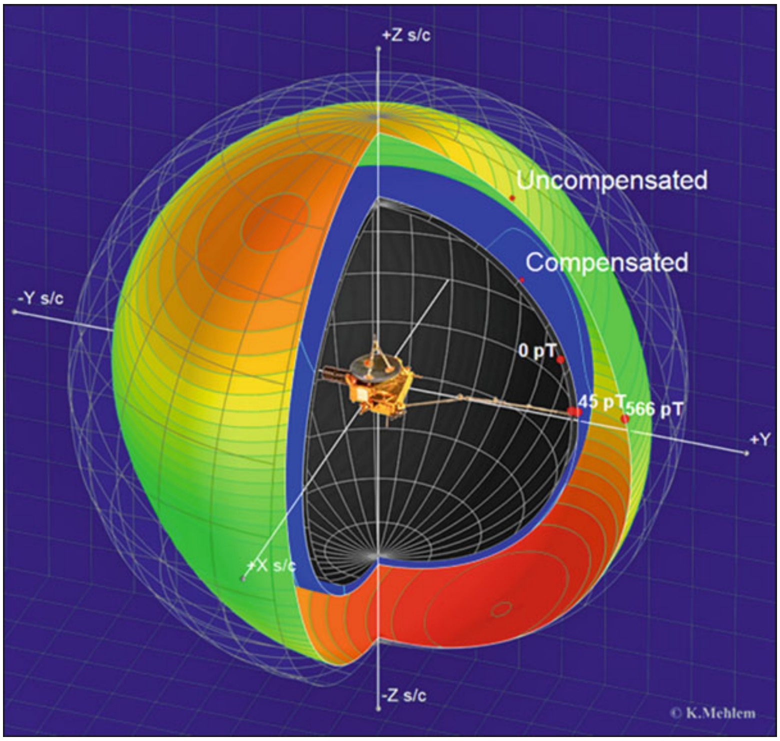


Fig. 13.35 Ulysses far-field |**b**| (RTG and TWTs uncompensated and compensated) overlay on the black zero-level sphere with $r = r^{VHM}$

13.4.3 Cluster

The four Cluster spacecraft explore the magnetic field around the Earth in a formation flight. The magnetic cleanliness specification for the location of the outboard fluxgate magnetometer (FGMO) was set to 0.25 nT. In order to achieve this value, the four spacecraft went through the most intense magnetic cleanliness program ESA has ever carried out (Fig. 13.36).

Each of the four spacecraft had a number of magnetically critical units and subassemblies (like thruster-valves, etc.). In a preparatory phase every unit was mapped and modeled by using a small ESA coil facility (Fig. 13.6) and the ESA MAGNET software which had been developed by the author and the University of Braunschweig, Germany. A synthetic spacecraft MDM was successively built up. Figure 13.37 depicts critical units on board of a Cluster spacecraft.

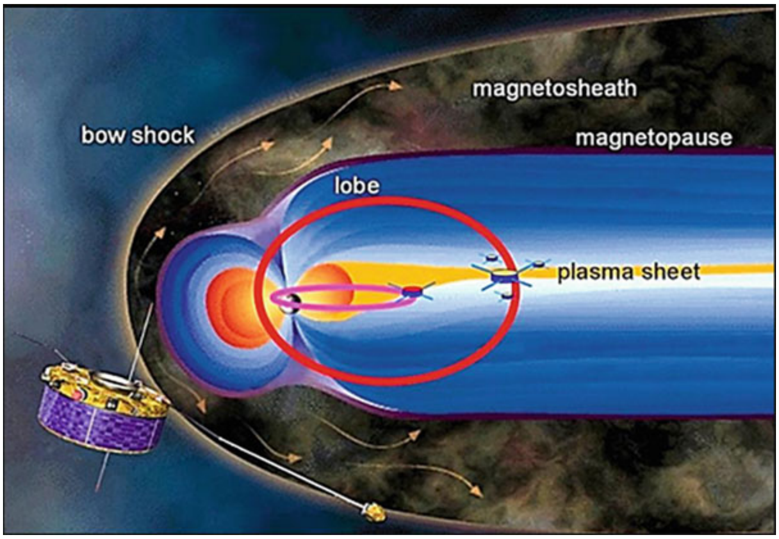


Fig. 13.36 The Cluster mission (four spacecraft on red orbit) (ESA)

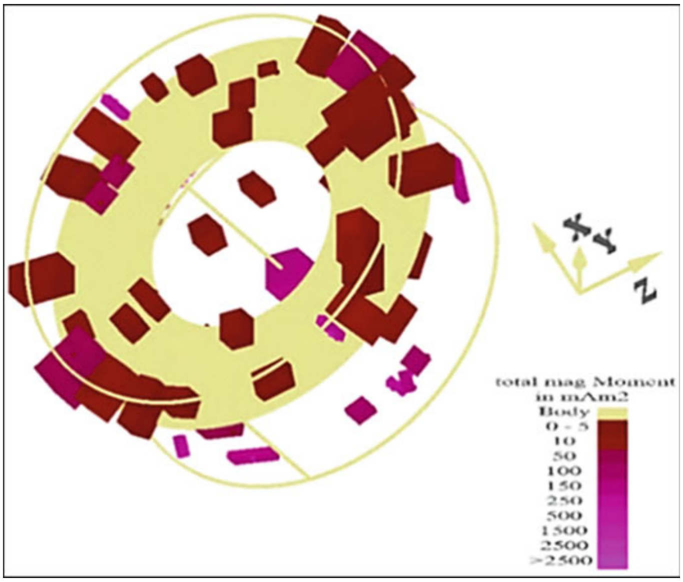


Fig. 13.37 Magnetic units (about 120) (H. Kügler, IABG)

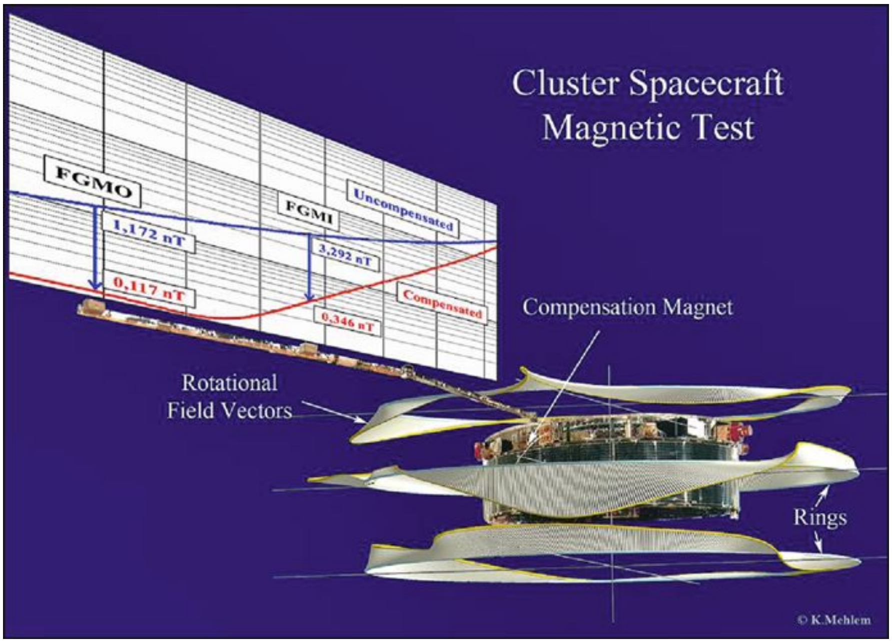
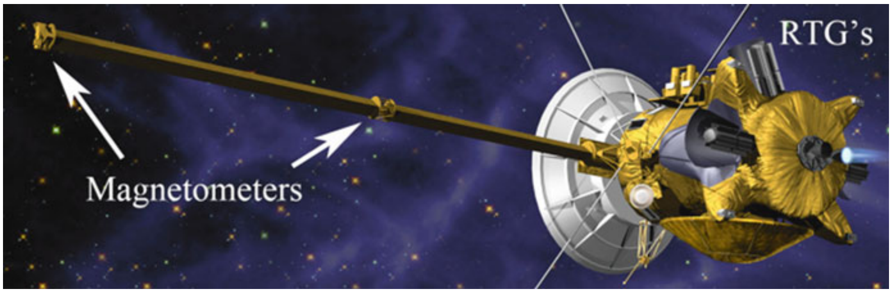


Fig. 13.38 Fall-off fields on a line through FGMI and FGMO and compensated rotational fields on three rings, compensation achieved by one magnet

Each complete spacecraft was tested in the large Helmholtz coil facility of IABG, Germany, in at least four magnetic states. The optimal MDMs for all rotational measurements and the associated fields at the CSP had to be calculated live during the test. Figure 13.38 gives an impression of the modeling challenge due to the presence of many eccentric magnetic sources. A minimum of 12 dipoles were required per MDM [14].

As the magnetic cleanliness specification of 0.25 nT was exceeded, a magnet was determined for the compensation of both the FGMI and the FGMO field. The corresponding magnet had to be split into two parts because only two orthogonal surfaces were available for the installation of the magnets. A so-called x,y-magnet with the components $[m_x, m_y, 0]$ had to be fixed on a small horizontal surface of the spacecraft at a position $\mathbf{p}_1$. A so-called z-magnet with the components $[0, 0, m_z]$ had to be fixed on a small vertical bracket installed on the spacecraft at a position $\mathbf{p}_2$. The moment vectors of both magnets have been optimized by using Eq. (13.51). Since we had three moment components to generate in total six compensating far-field vectors, only a least square solution could be achieved (see Sect. 3.8, point 2). Nevertheless, a reduction of the field by an order of magnitude at both locations was achieved (Fig. 13.38). It had then to be verified by a new measurement-modeling cycle. Each of the four spacecraft needed compensation. After the tragic launch failure of Ariane V and the loss of all four Cluster spacecraft, the recovery project

Cluster II was started. The same cleanliness program was successfully carried out again. In total about 50 MDMs were derived and all spacecraft were successfully compensated for FGMI and FGMO, each by use of one magnet only.



13.4.4 Cassini

Orbiting Saturn Cassini carries two magnetometers (one VHM, left, and one FGM, mid) on a 11 m long deployable boom. The target cleanliness level was set to 200 pT at the VHM location. The spacecraft is powered by three RTGs, similar to the one on Ulysses (Sect. 4.2).

A series of complex mapping exercises were performed in the industrial and radioactive environment of the EG&G facility in Miamisburg, Ohio, USA [11]. All RTGs (F2, F6 and F7) had been modeled by the author during the tests. When building the synthetic RTG model it turned out that in the worst case the RTGs would generate up to 115 pT at the VHM location, which is more than half of the whole spacecraft allocation (200 pT).

In an effort to avoid the problematic use of compensation magnets (radiation, stability, etc.) for each RTG, the author investigated the possibility of self-compensation when each RTG was allowed to be installed at any of the three locations, and when the rotation angles could be chosen in steps of 30° (Fig. 13.39). Under these conditions a formidable self-compensating configuration was found which dropped the field at the VHM location from 49 pT to 4 pT (Table 13.2) [5]. Figure 13.40 shows the strong self-compensation of the three RTG field vectors down to a rest vector of only 4.1 pT. In Fig. 13.41 the associated fall-off fields (nominal in blue and compensated in red) along a line through both magnetometers are shown.

These optimal location and clocking values of Table 13.2 were finally adopted by the project, and the RTGs were attached to the spacecraft accordingly. In-flight measurements have confirmed the success of this compensation exercise which represents a classical example of self-compensation techniques.

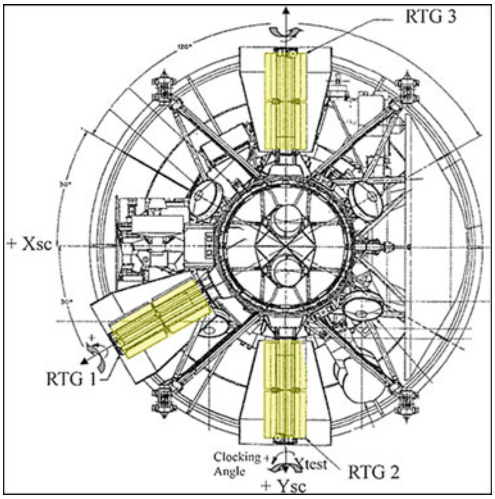


Fig. 13.39 RTG locations on Cassini

Table 13.2 Optimal clocking angles

RTG position	1	2	3	
Clocking angle	ϕ_1	ϕ_2	ϕ_3	$\lvert b\rvert_{VHM}$
	Degrees			pT
RTG configuration	F2	F6	F7	
Nominal	150	0	0	72
Minimum	−30	60	−90	17
RTG configuration	F2	F7	F6	
Nominal	150	0	0	73
Minimum	−30	60	−90	16
RTG configuration	F6	F2	F7	
Nominal	150	0	0	49
Minimum	−120	90	−60	4
RTG configuration	F6	F7	F2	
Nominal	150	0	0	116
Minimum	−30	−150	60	25
RTG configuration	F7	F2	F6	
Nominal	150	0	0	45
Minimum	−120	90	−30	8
RTG configuration	F7	F6	F2	
Nominal	150	0	0	111
Minimum	−30	−150	60	26

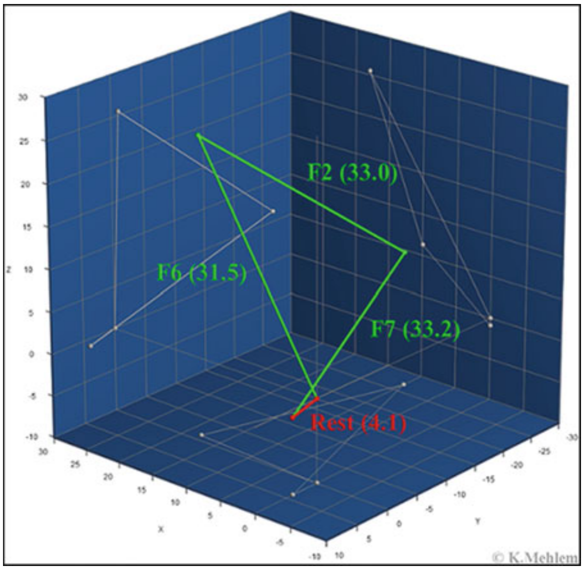


Fig. 13.40 Self-compensating far-fields

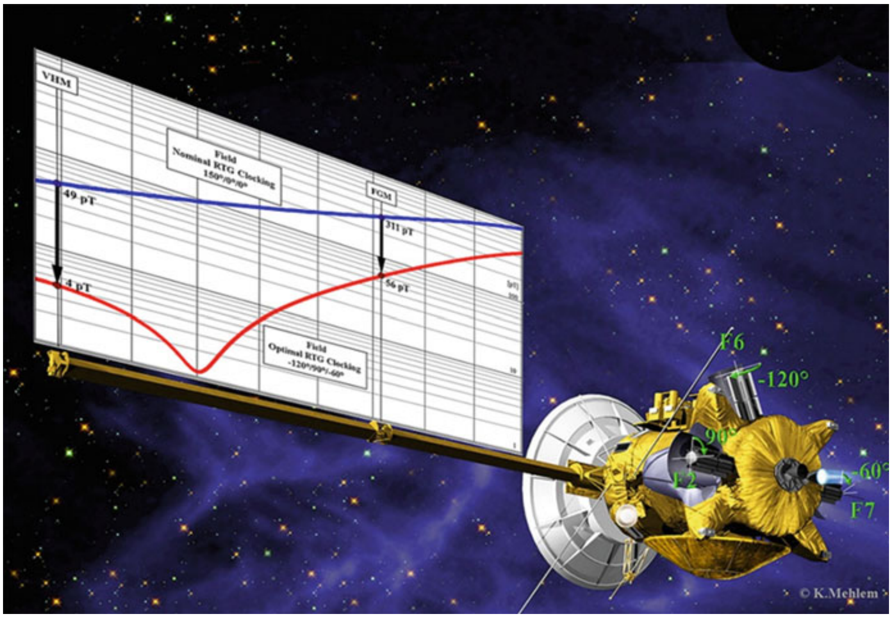


Fig. 13.41 Self-compensation of RTG fields achieved by optimal location and clocking parameters (green) (image: JPL, K. Mehlem)

13.4.5 Software

All results of the paper have been calculated with the MDM software GAMAG which integrates three decades of magnetic modeling experience. It is presently the most sophisticated while automatic MDM software is available (www.astos.de).

The software works for any rotational, translational, or static test setup, and more importantly, it requires only external input data: field or field gradient data, sensor positions, body constraints, CSPs and positions for compensation magnets.

It solves the problems related to constraint handling, optimal model sizing and reduction of model ambiguities internally. In fact, thanks to the use of a complex strategy of MDM sizing, and parameter resetting (and of course under the condition of consistent input data), the NLP process finds *always* an acceptable MDM. The internal steering parameters have been optimized such that 70 very different modeling cases could be solved in a single batch run.

The software has a user-friendly graphical user interface and it delivers a complete report on the optimization process, on the MDM properties, on the uncompensated far-fields, on the optimal compensation magnets, and on the compensated far-fields. The results are also available as 2D graphs and interactive 3D graphics similar to the ones shown in Fig. 13.40.

The modeling run for the case of Fig. 13.33, which included 144 field data and 9 dipoles, took 69 s, including the generation of 20 independent 9DM's for statistics. This compares to some hours of intensive hands-on work if similar results had to be produced by use of precursor MDM software.

13.5 Conclusion

A specific application of optimization in space technology, the magnetic modeling method MDM as used for magnetic cleanliness tasks, has been described in detail.

Special attention has been focused on the solution of optimal model sizing and on solution ambiguities, which are common problems in the domain of inversion tasks. In particular, an optimized strategy of combined MDM sizing and parameter re-settings guarantees that the NLP process never ends up in a relative minimum.

Multiple-point compensation techniques for fields exceeding the cleanliness specifications, have been detailed.

The extension of the MDM approach to field gradient measurements has also been described, together with an impressive example.

Finally, the solution of a number of different challenging modeling tasks, related to four spacecraft, has been described in some detail.

The GAMAG software which integrates the experience of magnetic modeling gained in a period of more than three decades has been briefly characterized as well. In particular, in the presence of a number of ill-conditioned modeling problems, it was a challenge in itself to develop a fully automatic software tool.

The author hopes that the presented MDM method will stimulate similar applications like reverse identification of test setup configurations by use of magnetic calibration sources (coils). Outside the field of EMC there are some potential areas of applications like gravitational modeling of irregular celestial bodies by point masses, trajectory optimization, attitude control etc. Other interesting areas are aerodynamic and propulsion modeling, industrial processes, and even medical MDM applications like for MEG, EEG, and ECG.

References

1. Musmann, G., Neubauer, F.M., Lammers, E.: Observations by the Helios-1 spacecraft. J. Geophys. **42**, 591–598 (1977)
2. Musmann, G.: Design guide for magnetic cleanliness control, internal paper GIOTTO (1982)
3. Kuegler, H.: Performance improvement of the magnetic field simulation facility MFSA. In: Proceedings of the 5th International Symposium on Environmental Testing for Space Programmes, Noordwijk (ESA SP-558, June 2004). Compiled by: K. Fletcher, pp. 407–414, Bibliographic Code: 2004ESASP.558.407K, p. 409, § 3 (2004)
4. Mehlem, K.: Multiple magnetic dipole modeling and field prediction of satellites. IEEE Trans. Magn. **14**(5), 1064–1071 (1978). ISSN: 0018-9464
5. Mehlem, K., Narvaez, P.: Magnetostatic cleanliness of the radioisotope thermoelectric power generators (RTGs) of Cassini. IEEE Int. Symp. Electromagn. Compat. **2**, 899–904 (1999). ISBN: 0-7803-5057-X
6. Campbell, S.L., Meyer Jr., C.D.: Generalized Inverses of Linear Transformations. Dover, New York (1991)
7. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (1999). ISBN 0387987932
8. Rosenbrock, H.H.: An automatic method for finding the greatest or least value of a function. Comput. J. **3**, 175–184 (1960). doi:10.1093/comjnl/3.3.175. doi:dx.doi.org, ISSN: 0010-4620
9. Zisserman, A.: Robotics Research Group in the Department of Engineering Science, University of Oxford, Andrew Zisserman's Home Page, Lecture Courses, B1 Optimization (Michaelmas Term 2011), Lecture 2 "Newton and Newton like methods, and the amoeba algorithm", www.robots.ox.ac.uk/~az/lectures/b1/lect2.pdf
10. Duffin, W.J.: Electricity and Magnetism, 4th edn. McGraw-Hill, London (1990)
11. Mehlem, K.: Cassini RTG F5, F2, F7, F6 magnetic analysis report, ESA/ESTEC-JPL internal working paper (1996/1997)
12. Mehlem, K.: Ulysses RTG F3 magnetic compensation, ESA/ESTEC internal working paper 1537 (March 1989)
13. Balogh, A., Beek, T.J., Forsyth, R.J., Hedgecock, P.C., Marquedant, R.J., Smith, E.J., Southwood, D.J., Tsurutani, B.T.: The magnetic field investigation on the Ulysses mission: instrumentation and preliminary scientific results. Astron. Astrophys. Suppl. Ser. **92**, 221–236 (1992)
14. Mehlem, K.: Magnetic properties of the Cluster I and II spacecraft, 11 ESA/ESTEC working papers (2000)

Chapter 14
Integrated Design-Trajectory Optimization for Hybrid Rocket Motors

Dario Pastrone and Lorenzo Casalino

Abstract Hybrid rocket motors present a competitive edge among space applications, as they are flexible, safe, reliable, and low cost. The motor design and operation are contingent on the type of designated mission; therefore, it is useful to couple design and trajectory optimization. This approach is especially needed for hybrid rockets in which a unique combustion process links thrust and specific impulse. In this chapter, a multidisciplinary optimization code is described for the use of designing hybrid rocket motors. There are few parameters which define the design of the hybrid engine, whereas the trajectory optimization is characterized by continuous controls. Due to the difference between these unknowns, a mixed optimization procedure has been developed. To maximize mission performance, direct optimization of engine design variables is coupled with the trajectory indirect optimization.Some interesting applications of the aforementioned procedure are presented.

Keywords Hybrid rocket motors • Multidisciplinary design optimization

14.1 Introduction

For the purposes of space transportation and space exploration, chemical rockets are still the most frequently used propulsion systems. Both liquid rocket engines (LREs) and solid rocket motors (SRMs) had a dynamic period of progress from the 1940s to the 1970s. Consequently, they can now be considered mature technologies, while hybrid rocket motors (HRMs) have not experienced the same intense development. In addition to some common chemical rocket challenges, one of the main HRM obstacles is the low regression rate (i.e., the rate at which the

D. Pastrone (✉) • L. Casalino
Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy
e-mail: dario.pastrone@polito.it; lorenzo.casalino@polito.it

Table 14.1 Advantages of hybrid propulsion systems

Compared to	Liquid	Solid
Performance	Higher fuel density	Higher specific impulse
	Additives in grain	Throttling/cut-off/restart capability
Safety	Reduced fire hazard	Reduced explosion hazard
	Reduced hard start	Reduced inadvertent ignition
System	One liquid line	Only-fuel grain
	Lower propellant management needs	Lower environmental impact
Cost	Reduced development costs	
	Reduced recurring costs	

solid fuel burns) of solid grains, which limits the thrust level and determines complex geometric constraints. Moreover, HRMs cannot guarantee the same performance of cryogenic LREs. Nevertheless, hybrid rockets have recently come to the forefront and have powered the first commercial vehicle for human suborbital flight [15]. In fact, they can provide a safe and affordable option for many applications (accessing upper atmosphere and space for scientific missions, educational projects, space tourism, etc.). A large number of works concerning the experimental and numerical analysis of HRMs show the interest in utilizing this type of chemical rocket. Many projects under development are currently conducted by research centers, national and international space agencies, and universities [6, 16, 22, 28, 29, 31].

Similar to other propulsion systems, HRMs present advantages and technical challenges. In HRMs, the oxidizer and fuel are stored apart in two different physical phases. In this case, the most common configuration is considered, using a liquid or gaseous oxidizer and a solid fuel. There are many interesting consequences of separating the two propellants and using two different phases. Containing only fuel, the solid grain is less prone to have defects and can be produced safer and cheaper compared to SRM grains, where both fuel and oxidizer are present. Other beneficial effects are pores or small cracks in the grain cannot cause deflagration to detonation transition as in SRMs; chemical explosion hazards are reduced and inadvertent ignition avoided.

Problems related to liquid propellant feed system and storage, such as components reliability, leaks, fire hazards and propellant management in microgravity or adverse acceleration conditions, are cut to half. Moreover, the solid phase presents higher density and also allows for easy inclusion of solid additives that can improve performance, such as aluminum. On the other hand, one of the propellant is kept fluid so that its flow can be controlled. It enables throttling, cutoff, and restart capability. Last but not least, propellants for HRMs are more environmental friendly than the most common oxidizer for SRMs, i.e., ammonium perchlorate (AP), and storable propellants for LREs, such as nitrous tetroxide (NTO) and monomethylhydrazine (MMH). Table 14.1 summarizes pros and cons of HRMs with respect to LREs and SRMs.

Challenges arise from the particular combustion process of HRMs, which is sketched in Fig. 14.1. The oxidizer flowing over the solid grain surface generates a

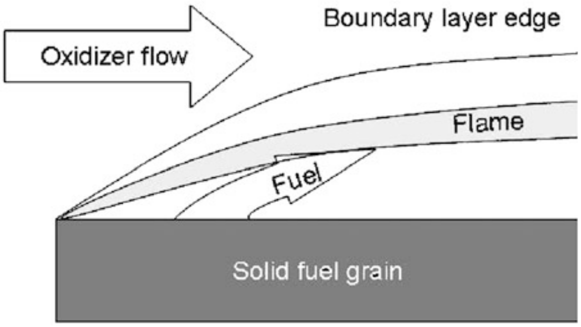


Fig. 14.1 Main features of the hybrid propellant combustion process

boundary layer. A diffusion flame is formed in a slightly fuel-rich region inside this layer. The fuel coming from the solid grain is gasified by the heat coming from the flame. A first issue that is related with the combustion process is propellant mixing. Due to the diffusion flame characteristics, part of the fuel in the region below the flame may not mix with the oxidizer above the flame and exit the nozzle before releasing all the chemical energy. Therefore, HRMs usually have a combustion efficiency which is lower when compared to LREs and SRMs. The introduction of a mixer, downstream of the solid grain and just before the nozzle, mitigates this problem. Of course, the system weight and dimensions are worsened, and a trade-off is required. Another important issue is the grain low regression rate. It results to be one order of magnitude smaller than the one typical of solid composite propellants. The main limitation is related to the heat transfer from the flow to the solid grain surface. In fact, unless the mass flux is very high or the pressure very low, chemical reaction kinetics in the flame is very fast when compared to the fuel gasification process, which in turn depends on the heat transfer from the flame. Approaches to this issue are discussed in Sect. 14.2. The heat transfer is mostly convective except for very large combustion chambers or when combustion gases contain a large number of solid particles. Regression rate experimental data are in agreement with pressure independent correlations based on the oxidizer mass flux $G_O = \dot{m}_O / A_p$ (i.e., the ratio of oxidizer mass flow $\dot{m}_O$ to port area A_p). Pressure dependence can be significant only in two cases: when the mass flux is very high so that chemical kinetics and heat transfer have comparable timescales or when mass flux is so slow that radiation and convection heat transfer are comparable. Due to the dependence on G_O, the regression rate is influenced not only by the oxidizer mass flow but also by the grain geometry, which changes during motor operation. As the fuel mass flow rate depends both on the regression rate and on the surface area where the gaseous fuel originates from, the mixture ratio changes even if the oxidizer mass flow is kept constant (mixture-ratio shifting). In SRMs, the grain composition determines the specific impulse, and the thrust law is determined by grain geometry history and nozzle geometry. Mixture ratio and chamber pressure can be separately controlled in classic bi-propellant LREs. On the opposite, the combustion process that occurs in HRMs conditions the design of the solid grain and engine operations. The fuel mass flow is ruled by grain and nozzle geometries and by the oxidizer mass flow: mixture ratio and chamber pressure are correlated. For this reason, the optimization of this propulsion system is particularly challenging and requires adequate methods.

The optimization of the propulsion system design and its operations is also strictly related to the type of mission considered. In many applications thrust has a major influence, since it affects both the propulsion system design and the trajectory performance; a compromise must be sought, as greater thrust levels reduce the gravitational losses but increase structural mass. For this reason, the design optimization should include trajectory analysis and its optimization. This is especially true when HRMs are considered, due to the aforementioned peculiar combustion process. Therefore, coupled optimization of engine and trajectory is necessary in most applications of hybrid rockets. In the present work, a fast and reliable multidisciplinary optimization code is described. The optimization procedure is a nested direct-indirect technique which aims at finding the engine design parameters and the trajectory that maximize the mission performance index. The number of engine design parameters is usually low, and their optimization is easily carried out by a direct method. On the other hand, the trajectory optimization is characterized by continuous controls (e.g., thrust direction), which requires discretization by means of a large number of parameters or the use of indirect methods. The nature of this problem suggests the use of a mixed optimization procedure. An efficient parametric optimization code, developed at the Politecnico di Torino [7], is used to find the optimal values for design parameters in conjunction with an indirect procedure [12] to optimize the trajectory.

Principles that determine the preliminary choices concerning propellant combinations, grain geometry, injectors, and feed system configurations are discussed in Sect. 14.2. Analysis of motor design and operation is presented in Sect. 14.3, providing a detailed description of the design variables and their influence on performance. The optimization procedure is described in Sect. 14.4. Examples of use of this procedure for interesting applications are presented in Sect. 14.5, and concluding remarks are provided in Sect. 14.6.

14.2 Preliminary System Options

Some decisions heavily condition the propulsion system and are chosen before the optimization procedure. Among others, the most salient factors are:

- Propellant combination
- Grain type
- Injector type
- Feed system/oxidizer control

There are few appropriate oxidizers which comply with the most important selection criteria, while a large choice of fuels exists. The most effective oxidizers envisaged for HRMs are liquid oxygen (LOX), hydrogen peroxide (HP), and nitrous oxide (N2O).

Oxygen is cryogenic and stored in liquid phase. It can be both injected as a liquid or gasified prior to combustion to improve combustion stability. Nevertheless, the means of vaporizing LOX introduce complexity, weight, and safety issues. Oxygen determines maximum values of characteristic velocity c^* at a low mixture ratio, thus making the low regression rate a more important issue. On the other hand, both HP (in H2O solutions) and N2O are slightly toxic and may decompose in the propellant tank. A catalytic bed is normally used for HP. N2O presents a much slower decomposition rate with respect to HP and is directly injected without a catalyzer. In this case, the system is lighter, but uncontrolled decomposition may determine overpressure. Nevertheless, N2O is considered to be fairly safe and is widely used for different applications.

The burning surface required to produce a given fuel mass flow is inversely proportional to the regression rate. Due to the low regression rate, the length of the grain may become difficult to manage if a cylindrical grain with a single circular port is used. Multiport grains mitigate this problem; however, they present some drawbacks: the presence of slivers (unburned mass fraction can be about 5–10%), complex design and fabrication, possible compromised grain structural integrity toward the end of burning, and uneven behavior of ports. The regression rate issue can also be mitigated using propellant combinations which have a maximal value of characteristic velocity c^* at high values of mixture ratio. In this case, the regression rate is of minor concern, as the fuel grain must give a minor contribution to the overall mass flow. Unfortunately, this happens when the oxidizer also contains atoms with non-oxidizing properties (e.g., nitrogen) and compromises performance.

Improvement of the regression rate continues to be the most straightforward option. Paraffin-based fuels naturally present high regression rates [20] since they form a melt layer at the combustion surface, which may become unstable [21]. The regression rate is enhanced by a new mass-transfer mechanism, i.e., entrainment of liquid droplets from the melt layer. The same effect may be obtained by freezing propellants which are liquid or gaseous at standard conditions (solid cryogenic hybrids). Heat transfer to the surface can be increased by introducing additives in the fuel (metal or hydride particles [17, 25], oxidizer particles) or by modifying the flow field with unconventional injectors (e.g., vortex hybrid rocket [23]) or grain geometries (e.g., cascaded multistage impinging-jet [26]). All these methods present drawbacks which may affect cost, safety, and the environment.

As far as the feed system is concerned, both pressurized and turbopump systems may be adopted. When requiring a modest level of velocity gain ΔV, the best choice is the simplest and more reliable blowdown type. In this case, the propellant flow is not controlled. Performance may be enhanced with a shot of repressurization, but an auxiliary gas tank is needed. Better performance may be obtained with a regulated feed system. Usually, the optimal solution requires a partially regulated operation in which a constant tank pressure phase is followed by a blowdown phase. When ΔV is increased, high values of tank pressure are necessary to keep the thrust magnitude and regression rate sufficiently large. Therefore, the masses of the oxidizer tank and auxiliary tank become large. In this case, the use of a turbo pump feed system can improve the rocket performance, reducing propellant tank mass and eliminating pressurizing gas and its tank. Since a HRM has only one liquid propellant, it is difficult to have high-energy fluid to feed a turbine to move the pump. Decomposing monopropellants (e.g., HP), heating liquid oxidizers, and using thrust chamber tap-off or auxiliary liquid propellants are solutions that introduce complexity, reduce reliability, and increase costs. For these reasons, an electrical drive system powered by batteries has been proposed [10].

14.3 Motor Design and Operation

After the preliminary design choices discussed in Sect. 14.2 have been fixed, the performance of a HRM depends on grain geometry, nozzle geometry, and oxidizer mass flow rate. The initial values of oxidizer flow rate, grain burning area, and port area determine the initial regression rate, thrust, mixture ratio, and specific impulse. The motor performance changes during operation; the oxidizer flow rate depends on the propellant feed system, whereas the regression rate and grain geometry determine how port and burning areas change with time. The guidelines to evaluate the motor performance are presented here.

14.3.1 Combustion and Performance

After selecting a propellant combination, the relevant properties of the combustion gases can be evaluated at a given mixture ratio and chamber pressure. Suitable codes [18] can be used to evaluate the characteristic velocity c^* and the specific heat ratio γ as a function of the mixture ratio α. Performance and gas properties are computed, assuming a frozen expansion (i.e., during nozzle expansion the combustion products maintain the composition they have in combustion chamber). To compute the thrust coefficient C_F, a one-dimensional isentropic expansion with a constant value of γ is taken into consideration. Real effects are accounted for by introducing a correction factor η_F [32] to modify the vacuum thrust coefficient, thus obtaining

$$C_F = \eta_F \left\{ \sqrt{\frac{2\gamma^2}{\gamma - 1} \left(\frac{2}{\gamma + 1} \right)^{\frac{\gamma+1}{\gamma-1}} \left[1 - \left(\frac{p_e}{p_c} \right)^{\frac{\gamma-1}{\gamma}} \right]} + \varepsilon \frac{p_e}{p_c} \right\} - \varepsilon \frac{p_{atm}}{p_c} \tag{14.1}$$

where ε is the nozzle-expansion area ratio, and p_c, p_e, and p_{atm} are chamber, exit, and ambient pressure, respectively. The pressure in the combustion chamber p_c

is determined via the ballistic model (see Sect. 14.3.2). C_F can then be evaluated, provided the ambient pressure and expansion (ε or p_e/p_c) are known.

The above-mentioned hypotheses are conservative assumptions that are introduced to account for the low combustion efficiency of HRMs. In actual expansion, some reassociation of the combustion products may occur, consequently improving performance. Moreover, the maximum performance for shifting equilibrium is attained at higher values of mixture ratio, with a reduction of the structural mass fraction.

In this work, third-degree polynomial fitting of c^* and γ are evaluated at a reference combustion pressure $p_c = 10$ bar. The resulting expressions are embedded in the code to compute the proper values as the mixture ratio changes during motor operations, also including a proper c^*-efficiency η^* [32]. It should be noted that the actual pressure in the combustion chamber can span over a wide range during motor operations. Nevertheless, it has been verified that the error due to the above-mentioned fits is usually very small.

14.3.2 Ballistic Model

After selecting the propellants, it is possible to find a proper regression rate correlation. Classical head-end injector is here used, and the regression rate $\dot{y}$ is expressed as a function of the oxidizer mass flux $G_O = \dot{m}_O/A_p$, according to the frequently used correlation

$$\dot{y} = aG_O^n = a(\dot{m}_O/A_p)^n. \tag{14.2}$$

Values of a and n can be found in literature [14, 21, 30, 34], Care must be taken as these semi-empirical correlations are usually valid for a given mass flux range and can be motor-scale related.

The fuel mass flow is

$$\dot{m}_F = \rho_F \dot{y} A_b = \rho_F \dot{y} L_b P, \tag{14.3}$$

where L_b is the grain length, ρ_F is the fuel density, and P is the burning perimeter. Under the assumption of incompressible turbulent flow, the oxidizer flow rate is evaluated as

$$\dot{m}_O = \sqrt{(p_f - p_1)/Z}, \tag{14.4}$$

where p_f is the pressure level due to the feeding system (tank pressure with pressurizing gas feed system or discharge pressure with turbopump), p_1 is the head-end pressure, and Z is the hydraulic resistance in the oxidizer flow path from the tank or pump discharge to the combustion chamber. The head-end pressure p_1 can be related to the chamber/nozzle stagnation pressure p_c by means of approximate relations, such as the one proposed by Barrere et al. [3] for side-burning grains

$$p_1 = \left[1 + 0.2\left(\frac{A_{th}}{A_p}\right)^2\right] p_c, \tag{14.5}$$

where A_{th} is the throat area. It is assumed that Z is constant during motor operation. The mixture ratio

$$\alpha = \frac{\dot{m}_O}{\dot{m}_F} \tag{14.6}$$

and the corresponding value of c^* are then determined. An isentropic expansion is assumed in the nozzle, and the chamber/nozzle stagnation pressure p_c is evaluated as

$$p_c = \frac{(\dot{m}_O + \dot{m}_F)c^*}{A_{th}}. \tag{14.7}$$

14.3.3 Grain Geometry

The initial values of port area and burning area determine the initial mass flow rate; during combustion, these areas change according to the burn distance y. The evaluation of the rocket performance requires the knowledge of the instantaneous grain geometry during combustion.

In this work, common circular grain sections are adopted. The single-port geometry with axial symmetry and a circular port is the simplest case. The port radius describes the cylindrical single-port geometry, whereby the initial value is R_{ci} and the circular port area is

$$A_p = \pi(R_{ci} + y)^2 \tag{14.8}$$

with burning perimeter

$$P = 2\pi(R_{ci} + y). \tag{14.9}$$

Because of low regression rate values, single-port grains require large values of length to obtain prescribed thrust level; multiple-port grains reduce grain length.

Two different multiport grain geometries are considered in the following discussion. The first one [4] has triangular ports, defined by the number of ports N, the web thickness w, and the grain outer radius R_g. Making reference to

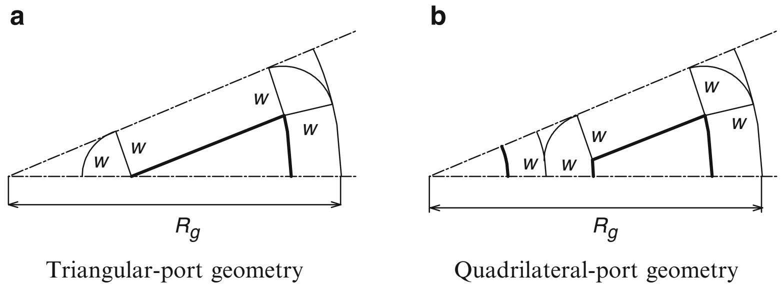


Fig. 14.2 Multiport geometries

Fig. 14.2a, initial port area and burning perimeter can be expressed [13] as functions of R_g and w, assuming minimum sliver. For a given burning distance y ($0 \leq y \leq w$), the burning perimeter and the port area can be calculated.

The second geometry assumes a wagon-wheel grain with a central circular port with initial radius R_{ci} surrounded by a row of N quadrangular ports, as shown in Fig. 14.2b. The value of the initial area for each quadrilateral port is assumed to be equal to the area of the central circular port to consequently obtain the same initial regression rate [33]. The geometry is again defined by R_g and w when imposing minimum sliver. It is also assumed that the regression rate remains the same in quadrilateral and circular ports for the duration of the burn, despite differences among port area and burning surface history. This assumption has a limited impact on performance, as the circular port only deals with a limited fraction of the overall flow. One can then solve the grain initial geometry [13], determining initial port area and burning perimeter as functions of R_g and w. During the operation, the values of port area and burning perimeter can then be derived as functions of y. Both $N = 6$ and $N = 8$ grains are considered in the following.

14.3.4 Feed System and Oxidizer Flow Rate

The pressure p_f provided by the feed system must be known to evaluate the motor behavior. If a pressurized feed system is adopted, p_f is equal to the tank pressure p_t. With the simplest blowdown system, only the mass of pressurizing gas m_g stored in the oxidizer tank is required to determine p_t. The initial ullage volume $(V_g)_i$ (i.e.,the initial gas volume in the propellant tank) can be more conveniently utilized as a design parameter.

When considering a partially regulated feed system a phase with constant tank pressure is introduced, followed by a blowdown phase. Pressure is maintained constant by means of a pressurizing gas (e.g., helium) flowing from an auxiliary tank. In addition to $(V_g)_i$, the initial pressure in the gas tank and the length of the constant-pressure phase must be known to determine the tank pressure history. For the valve's control stability, $(V_g)_i$ is typically related to overall oxidizer mass

$(m_O)_f$, which is generally unknown a priori; they can instead only be obtained at burnout after integration of Eq. (14.4). A tentative value, which becomes an additional unknown of the trajectory, is necessary to start the integration. The tentative value for $(m_O)_f$ is corrected by the trajectory indirect optimization procedure to match the value obtained at burnout after the integration of Eq. (14.4). During the regulated phase $p_t = (p_t)_i$, whereas p_t is calculated assuming an isentropic expansion of the pressurizing gas in the tank during the blowdown phase

$$p_t = (p_t)_i \left[\frac{(V_g)_{BD}}{V_g} \right]^{\gamma_g} \tag{14.10}$$

where $V_g = (V_g)_i + m_O/\rho_O$ and $(V_g)_{BD}$ is the gas volume in the oxidizer tank at the beginning of the blowdown phase. When the simpler blowdown pressurization is chosen, $(V_g)_{BD} = (V_g)_i$. Otherwise, $(V_g)_{BD} = (V_g)_i + (m_O)_{BD}/\rho_O$ for the regulated feed system, where $(m_O)_{BD}$ is the exhausted oxidizer mass at the start of blowdown operation.

Self-pressurization, related to the high vapor tension at ambient temperature, is an interesting feature which makes N2O a suitable propellant. This fact and the safety characteristics of N2O explain the wide use of nitrous oxide as an oxidizer in HRMs. When a self-pressurizing propellant such as N2O is used, the mass flux between the two phases becomes a fundamental control mechanism of the tank pressure change. Suitable models [8] can be developed to describe the pressure evolution in the tank during the burn; in particular, a two-phase model, where saturated vapor and superheated liquid are considered and the liquid/vapor mass-transfer evaluation is based on the liquid spinodal line, is adopted here. It provides the propellant tank pressure, which coincides with p_f, as a function of the exhausted oxidizer mass.

When a turbopump is used to feed the oxidizer, an electric system is used to drive the pump, and the needed energy comes from batteries. The masses of these main components (pump, electrical drive system, and batteries) have to be evaluated. Existing literature provides typical values of power density (power to mass ratio) for liquid propellant turbopump systems [27] and expected values for the electric drive system [1]. As far as batteries are concerned, power and energy densities are correlated [24]. The mass of the batteries depends on the most stringent requirement between the maximum electrical power required and the electrical energy needed to drive the pump during the burning time. The latter requirement usually prevails with long burning times.

14.3.5 Engine Operation

The choice of a limited number of design variables, (e.g., the initial thrust) determines the initial geometry and the engine operation. The initial oxidizer and fuel flow rates can be evaluated from the initial values of thrust (F_i) and mixture ratio (α_i)

$$(\dot{m}_p)_i = \frac{F_i}{c_i^*(C_F)_i} = (1+\alpha_i)(\dot{m}_F)_i = \frac{1+\alpha_i}{\alpha_i}(\dot{m}_O)_i = \frac{F_i}{c_i^*(C_F)_i}, \tag{14.11}$$

where c^* and C_F are computed as outlined in Sect. 14.3.1. Equation (14.4) is then used to evaluate Z.

The initial value of the throat area $(A_{\mathrm{th}})_{\mathrm{i}}$ can then be derived:

$$(A_{th})_i = \frac{(\dot{m}_p)_i}{(p_c)_i c_i^*}. \tag{14.12}$$

Throat area may be considered constant for a preliminary design analysis (note that throat erosion occurs and may plague performance especially when burning times are long and combustion gases are oxidizer-rich). The initial port area $(A_p)_i = (A_{th})_i/J$ is then evaluated given the initial throat-to-port area ratio J. Mass conservation also provides the initial burning area

$$(A_b)_i = \frac{(A_p)_i^n}{a\rho_F}\frac{(\dot{m}_F)_i}{(\dot{m}_O)_i^n} \tag{14.13}$$

and the grain geometry can thus be determined immediately in the case of a single circular port. An iterative procedure is instead required when complex geometries are adopted. A tentative value is assumed for R_{g}, and the grain geometry is solved in order to provide the required port area; the web thickness w and the initial perimeter P_{i}, which, in turn, provides the grain length $L_b = (A_b)_i/P_i$, are thus found. Once the initial geometry is determined, Eq. (14.2) is integrated up to burnout and the web thickness is computed. The trajectory indirect optimization procedure corrects the tentative value for R_{g} to match the necessary condition $y_f = w$ at burnout.

Numerical integration of Eqs. (14.2) and (14.4) gives the fuel grain geometry and the exhausted oxidizer mass, respectively. The latter provides the tank pressure during blowdown phases. A numeric procedure is required to determine the operating conditions of the HRM; at each instant t, after evaluating feeding system pressure p_{f} and motor geometry, the regression rate, the propellant flow rates (and their ratio α), and p_{c} and p_1 are computed by numerically solving the equations shown in Sect. 14.3.2 and using the curve fit for c^* as a function of α. Then, in order to integrate the trajectory equations, the thrust level $F = p_{\mathrm{c}}A_{\mathrm{th}}C_F$ is determined by evaluating C_F at the actual altitude via Eq. (14.1). At burnout the overall propellant mass (exhausted plus sliver) is finally evaluated, and an estimation of the structural masses is obtained.

14.4 Optimization

The optimization of a propulsion system must aim at the determination of the design variables that guarantee the fulfillment of the mission requirements while maximizing (or minimizing) a given performance index. Mass is typically one of the most important issues in space systems, and optimization problems are often in the form of maximization of mission performance for given propulsion system mass or, vice versa, propulsion system mass minimization for given mission performance.

The specific details of the mission determine the complexity of the optimization problem. In the simplest cases, e.g., satellite auxiliary propulsion, the spacecraft mass may be considered constant, and mission performance can be expressed by the ideal velocity gain ΔV . In this case, the engine optimization can be carried out separately.

In most cases, it is instead difficult to determine the correlation of mission performance and design variables. Thrust may influence the flown trajectory and the corresponding velocity losses, thus changing the required ΔV . This is in particular true for HRMs, where thrust, engine mixture ratio, and performance change during operations in a complex way, dictated by grain geometry and oxidizer flow rate. Also, the optimization problem itself may be more complex. For instance, if a HRM is used as a launcher upper stage, the overall launcher performance must be considered, and the HRM analysis cannot be made independently of the whole ascent trajectory optimization.

A nested direct/indirect optimization procedure is used to carry out the simultaneous optimization of design and trajectory. After the preliminary design choices have been done, the design of the HRM concerns a limited set of variables, which define the grain and nozzle geometry and the feed system. Also, it may not be possible to define the design by means of explicit relations (for instance, the nozzle mass flow rate can only be determined numerically with an iterative procedure); for these reasons, its optimization is dealt with by a direct optimization methods [7].

Given the engine design, the trajectory of the rocket is instead optimized by means of a fast and accurate indirect optimization procedure [12]. Indirect optimization methods effectively treat problems with continuous controls (thrust direction), with limited computational requirements. For each set of engine design variables, the procedure finds the trajectory which optimizes the performance index. The direct method is used to optimize the design parameters.

According to the model presented in Sect. 14.3, the motor design is determined by a limited number of parameters. The engine optimization typically concerns initial thrust level F_i, initial mixture ratio α_i, nozzle-expansion ratio ε, initial value of chamber pressure $(p_\mathrm{c})_\mathrm{i}$, initial value of feeding pressure $(p_\mathrm{f})_\mathrm{i}$, and initial port to initial throat area ratio J.

Not all these parameters are free, but some of them are constrained by safety or operation limits. As an example the coupling of the hybrid motor and the oxidizer feed system is to be avoided. Therefore, the initial chamber pressure is typically assigned, for instance, by imposing $(p_\mathrm{c})_\mathrm{i} = 0.4\,(p_\mathrm{f})_\mathrm{i}$, which is sufficient to guarantee $p_\mathrm{f}/p_\mathrm{c} > 1.5$ during operation. Also, the design variables may be implicitly determined by several constraints related to the trajectory (e.g., maximum values of acceleration or minimum values of regression rate). These cases are dealt with by dropping the relevant variables from the engine optimization, and considering them as additional variables of the trajectory optimization, during which they are determined in order to fulfill the relevant constraints.

For these reasons just three–five design parameters are to be optimized in the present model. Given a set of design optimization parameters, operating conditions during the engine burn are determined as shown in Sect. 14.3.5, and the trajectory equations can be integrated. Tentative values are first assigned to the design variables; the indirect optimization method optimizes the trajectory given the engine design and determines the performance index (few seconds are required on a standard PC). The design parameters are then varied by small quantities to evaluate numerically the derivatives of the performance index with respect to the design parameters, according to a forward-finite-difference scheme [2]. An iterative procedure based on Newton's method [2] is then applied to correct the tentative values in order to nullify the partial derivatives of the performance index.

As far as trajectory optimization is concerned, the reader can refer to [5] for a discussion of the theory of optimal control and to the chapter of the present book "Indirect Methods for the Optimization of Spacecraft Trajectories," by G. Colasurdo and L. Casalino, where it is applied in a form suitable to deal with the specificities of trajectory optimization. The optimal problem is transformed into a boundary value problem, which is solved with an iterative procedure.

14.5 Results

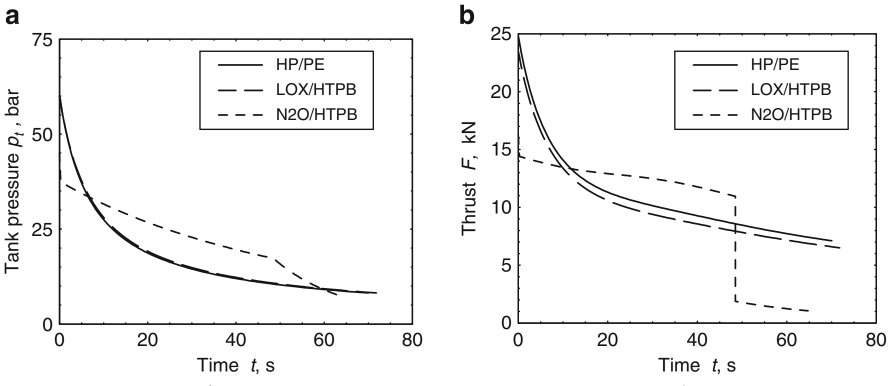
Different applications of a HRM are here considered. A sounding rocket and a booster accelerator for a hypersonic test bed are first analyzed; low cost is the most important objective to be pursued in these cases, and simple single-port geometries and blowdown feed system are assumed. The use of a HRM as a launcher upper stage is then investigated; attention is focused on performance and multiport grains and more complex feed systems are considered.

14.5.1 Sounding Rockets for Microgravity Platform

Hybrid rockets represent a low-cost option to transport payloads to high altitudes, where microgravity conditions can be experienced in free flight, due to the absence of atmosphere. Rocket payload (100 kg) and initial mass (500 kg, comprising payload, fixed masses, propulsion system, and propellant) are here given, and the time spent above 100-km ($t_{\mu g}$) is the performance index to be maximized. The optimizations aims at finding the optimal mass split between propellant and propulsion system (i.e., tanks, combustion chamber, nozzle), the optimal grain geometry, and the corresponding optimal trajectory [8]. Single-port grains are considered, and different propellant options are compared in Table 14.2. A simple blowdown feed system is adopted. The design variables are the initial values of thrust and mixture ratio, and the nozzle-expansion area ratio. The initial tank pressure and

Table 14.2 Optimal design and performance of sounding rocket for different propellant combinations

Propellants	F_i kN	α_i	ε	$(p_\mathrm{t})_\mathrm{i}$ bar	(V_g) m^3	m_p kg	R_t m	D m	L m	$t_{\mu g}$ s
HP/PE	24.8	8.57	4.98	60.1	0.094	339	0.048	0.42	5.01	299
LOX/HTPB	23.5	3.20	4.53	59.9	0.097	328	0.047	0.46	5.52	219
N2O/HTPB	16.2	10.92	4.14	45.0	–	340	0.046	0.40	4.75	177



Tank pressure Thrust

Fig. 14.3 Sounding rockets operation

ullage volume are additional design variables for HP and LOX; $(p_\mathrm{t})_\mathrm{i}$ is instead fixed (liquid–vapor equilibrium at ambient temperature) for the self-pressurizing N2O, with ullage volume fixed at 3% of the tank volume.

Results show the superior performance of the HP/PE combination. The use of LOX is penalized by the low mixture ratio, which requires a large grain and therefore increases the propulsion system mass to the detriment of propellant. On the other hand, N2O is penalized by the low performance in terms of specific impulse. However, N2O presents some interesting features, such as the reduced rocket length L and diameter D, as the large mixture ratio reduces the grain length.

It is interesting to note that the optimal initial thrust is quite different, depending on the propellant combination. There is a complex influence of the design choices on the thrust history, which in turn determines the trajectory. The importance of the optimization becomes clear if the thrust magnitude is chosen incorrectly. For instance, if the initial thrust level of the HP/PE combinations is adopted for the N2O/HTPB combination and vice versa, the microgravity times show a remarkable decrease to 150 s for N2O/HTPB (– 15%) and to 267 s for HP/PE (– 10%).

The use of a self-pressurizing oxidizer such as N2O, determines a different evolution of the tank pressure, as shown in Fig. 14.3a. The evaporation of liquid maintains the tank pressure at high values and avoids the relevant pressure decrease of the blowdown cases. As a consequence, the thrust level is more regular for the

Table 14.3 Optimal design and performance of hypersonic accelerator

Config.	Payload kg	α_i	F_i kN	$(p_t)_i$ bar	$(V_g)_i$ m^3	ε	D m	L m	M_f
1	100	7.43	26.64	58.20	0.265	5.89	0.431	8.62	7.51
1	200	7.39	29.57	70.86	0.191	5.88	0.407	8.15	5.34
1+1	100	8.62	34.73	86.27	0.082	4.67	0.468	4.68	7.50
1+1	200	8.96	37.10	93.77	0.079	5.11	0.454	4.54	4.80
2+1	100	9.10	18.10	64.41	0.076	6.71	0.408	4.08	7.68
2+1	200	8.49	18.02	75.64	0.058	6.35	0.390	3.90	5.32

N2O/HTPB case, as shown in Fig. 14.3b. This holds, of course, until the liquid phase is present. The final thrust drop and change of pressure derivative occur when the liquid phase in the tank vanishes, and gaseous oxidizer is fed into the combustion chamber. One should observe that, in this case, the pressurizing gas is not simply an inert mass but also acts as a propellant.

14.5.2 Booster for Hypersonic Testbed

There is a growing interest in hypersonic propulsion; testing of a hypersonic device requires its acceleration to Mach number above 5 at altitudes ranging from 25 to 50 km; a hybrid rocket is again a viable option for this kind of mission.

The performance of the HP/PE propellant combination for a single-port grain is presented in Table 14.3. The initial mass is fixed at 1,000 kg; the Mach number M_f at a 30-km altitude (horizontal flight) is maximized for payload of either 100 or 200 kg. The rocket spends most time at low altitudes, and an accurate modeling of the aerodynamic drag is necessary to obtain significant results [9]. A single stage design (1) is compared to two-stage designs, which use the same motor in both stages (to minimize development costs): in the first design, a single hybrid motor is used in each stage (1+1); in the second one, two motors are used in parallel in the first stage while one motor is maintained in the second stage (2+1). The optimization variables are the same as in the previous case, except for the initial volume of pressurizing gas which is determined in order to assure a sufficient regression rate at burnout.

There is a complex interaction between the trajectory and the design variables; small thrust and tank pressure should be preferred to reduce the propulsion system mass. In certain cases (single-stage, 100-kg payload and 2+1 configuration, both payloads), thrust would become too small and unfeasible solutions with negative coast arcs would be obtained. To avoid this situation, the initial thrust is dropped from the design variables and is instead determined by the trajectory optimization procedure in order to find a feasible solution.

Results show that a HRM can effectively accelerate a payload to the hypersonic corridor. Counter-intuitive results are obtained; it is observed that a two-stage design which employs the same engine in both stages does not provide any benefit, except when the 1+2 configuration is adopted for the lowest payload.

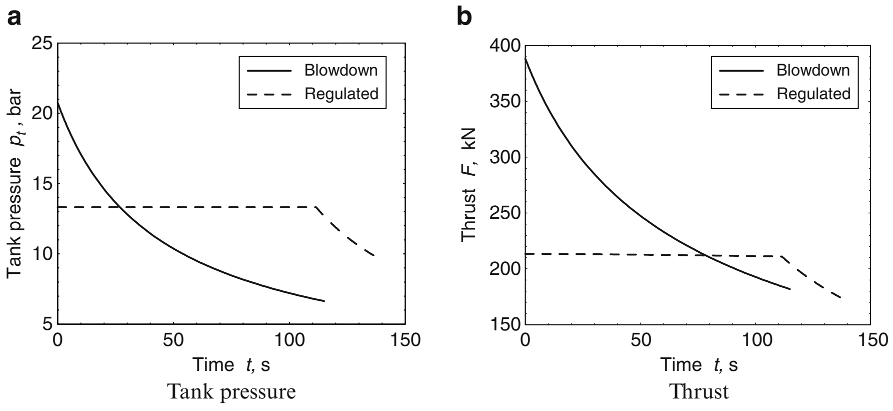


Fig. 14.4 Tank pressure and thrust history during launcher upper stage operation

14.5.3 Launcher Upper Stage

Small and low-cost launchers are commonly based on SRMs. SRMs are a good choice for lower stages, but are not suitable for upper stages. The low specific impulse, compared to LREs, penalizes the final part of ascent, and the launcher performance is plagued. Moreover, the lack of throttling and cut-off/restart capability determines a large scattering in the orbit insertion parameters. This issue may be solved by introducing an additional small LRE, but complexity is augmented and structural efficiency worsened. A HRM upper stage may synergistically perform the task of this small LRE plus the task of the SMR upper stage. Performance which is similar to that of storable liquid propellants is obtained at a lower cost. In addition, safety steps needed by chemicals, such as NTO and MMH, are eliminated. The replacement of third and fourth stages of a launcher based on the characteristics of Vega [19] with a single hybrid motor is here investigated. The goal is to maximize the payload delivered into a 700-km polar orbit with a launch from Kourou. The payload of the reference launcher (three solid-propellant stages and a liquid-propellant fourth stage) is about 1,400 kg, consistent with the performance of Vega. The launcher with the HRM third stage uses the same lower stages of the original case and the rocket initial mass is kept constant. Trajectory is optimized from lift-off to orbit insertion, retaining the same constraints of the reference launch (e.g., heat flux limits after fairing disposal) [11].

HP/PE is the selected propellant combination, and different options for the feed system are compared. Large thrust levels are required, and a multiple-port configuration is mandatory to obtain feasible designs. The use of different feed systems and port geometry is compared. Figure 14.4 shows tank pressure and thrust histories for two feed system options: blowdown and partially regulated. Mixture ratio is shown in Fig. 14.5. Quadrilateral ports and $N = 8$ are assumed. When the blowdown system is selected, thrust has a strong time dependence. Large values of the initial

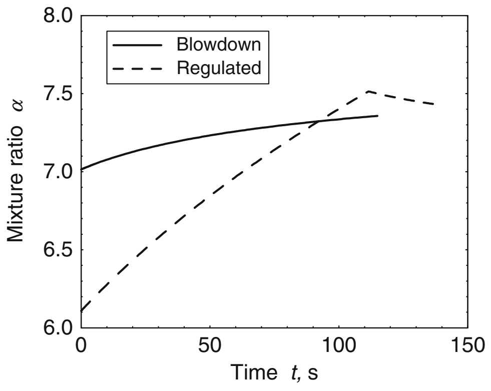


Fig. 14.5 Mixture ratio history during launcher upper stage operation

Table 14.4 Optimal design and performance for triangular ports (TP) and quadrilateral ports (QP)

Ports shape	Feed option	N	F_i kN	α_i	$(p_t)_i$ bar	ε	$\bar{I}_{sp}$ s	m_{res} kg	L m	$(T/m)_{max}$ g	m_u kg
TP	Blowdown	6	365.1	7.08	20.15	11.83	272.3	43.6	12.3	5.77	1,796.1
QP	Blowdown	6	359.2	7.09	20.72	12.14	272.9	31.0	12.2	5.67	1,813.8
TP	Blowdown	8	414.1	7.02	19.77	11.30	271.4	57.2	12.0	6.64	1,774.9
QP	Blowdown	8	388.2	7.01	20.76	11.92	272.6	31.8	11.9	6.18	1,810.7
TP	Regulated	6	209.2	6.22	13.02	11.91	272.2	51.7	9.8	5.49	1,989.4
QP	Regulated	6	203.2	6.21	13.54	12.39	273.0	37.3	9.6	5.35	2,011.9
TP	Regulated	8	230.5	6.16	12.33	11.17	270.8	67.6	9.4	6.33	1,970.6
QP	Regulated	8	213.4	6.11	13.32	12.09	272.5	38.6	9.3	5.82	2,013.4

tank pressure are needed to have appropriate values of thrust level without requiring large initial volumes of the pressurizing gas. Benefit can be obtained with a partially regulated feed system. An additional tank is required for the pressurizing gas, but the oxidizer propellant tank is lighter. The benefits are however penalized by a stronger mixture ratio shifting during the constant-pressure phase. Additional improvements can be obtained by using a turbopump feed system [10].

The grain geometry plays a fundamental role, as it affects grain size, volumetric efficiency, motor operation, and sliver. Triangular or quadrilateral ports (wagon-wheel geometry) with $N = 6$ and $N = 8$ are here considered to highlight the influence of the port number and shape. Table 14.4 shows the optimal values of design parameters and payload m_u for the corresponding four combinations that are considered here. As far as N is concerned, better performance are obtained with six holes except for the case where a wagon-wheel geometry is adopted with a regulated feed system. When compared to triangular ports, the wagon-wheel design exhibits a lower sliver m_{res}, thanks to the presence of the central circular hole.

Anyway, the corresponding mass saving (which ranges from 12 to 29 kg) cannot explain alone the resulting payload improvement (which ranges from 17 to 43 kg). In addition to sliver reduction, the wagon-wheel geometry mitigates the mixture ratio shifting and thus improves the average specific impulse $\bar{I}_{sp}$. The resulting specific impulse improvement is not large (about 1%), but gives a significant propellant mass saving because it is exploited during the final part of the ascent trajectory. Furthermore, the engine features are chosen as a compromise between different requirements: for a given feed system, the tank pressure for quadrilateral port grain is higher, while thrust level is lower. As a result the propellant tank is heavier, while the grain is shorter and lighter. The lower thrust level adopted with quadrilateral ports also determines a milder acceleration with a reduced peak value $(T/m)_{\max}$.

14.6 Final Remarks

The examples reported in Sect. 14.5 show that the optimization of this peculiar yet appealing propulsion system requires expertise and proper tools. The optimization is characterized by the presence of opposite requirements (e.g., high thrust levels to have low velocity losses vs low thrust levels to reduce propulsion system dry mass) and is further complicated by the relation existing between thrust level and mixture ratio. Only tools which couple design parameters and trajectory optimization may fully highlight the true benefits that can be obtained with HRMs. In fact, the optimization analysis provides counter intuitive conclusions, which are instructive guide and basis for engineering design. These results may be lost if simplifications are introduced which do not take into account the nature of the problem. In order to efficiently perform a coupling of the HRM design parameter and trajectory optimization, the procedure developed at the Politecnico di Torino uses a direct/indirect nested method. This approach has been proved to be fast and reliable.

References

1. Abel, T.M., Velez, T.A.: Electrical drive system for rocket engine propellant Pump. US Patent 6457306 B1, USA (2002)
2. Abramowitz, M., Stegun, I.A. (eds.): Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing. Dover, New York (1972)
3. Barrere, M., Jaumotte, A., De Veubeke, B.F., Vandenkerckhove J.: Rocket Propulsion. Elsevier Publishing Company, New York (1960)
4. Ben-Yakar, A., Gany, A.: Hybrid engine design and analysis. Paper AIAA 93–2548. AIAA, Reston VA USA (1993)
5. Bryson, A.E., Ho, Y.C.: Applied Optimal Control, rev. edn. Hemisphere, Washington DC (1975)

6. Carmicino, C., Russo Sorge, A.: Role of injection in hybrid rockets regression rate behaviour. J. Propul. Power (2005). doi: 10.2514/1.39578
7. Casalino, L., Pastrone, D.: Oxidizer control and optimal design of hybrid rockets for small satellites. J. Propul. Power (2005). doi: 10.2514/1.6556
8. Casalino, L., Pastrone, D.: Optimal design of hybrid rocket motors for microgravity platform. J. Propul. Power (2008). doi: 10.2514/1.30548
9. Casalino, L., Pastrone, D.: Optimization of hybrid sounding rockets for hypersonic testing. AIAA 2009–4841. AIAA, Reston VA USA (2009)
10. Casalino, L., Pastrone, D.: Optimization of a hybrid rocket upper stage with electric pump feed system. AIAA-2010-6954. AIAA Reston VA USA (2010)
11. Casalino, L., Pastrone, D.: Optimal design of hybrid rocket motors for launchers upper stages. J. Propul. Power (2010). doi: 10.2514/1.41856
12. Casalino, L., Colasurdo, G., Pastrone, D.: Optimal low-thrust escape trajectories using gravity assist. J. Guid. Contr. Dynam. (1999). doi:10.2514/2.4451
13. Casalino, L., Pastrone, D., Simeoni, F.: Hybrid rocket upper stage optimization: Effects of grain geometry. AIAA-2011-6024. AIAA, Reston VA USA (2011)
14. Doran, E., Dyer, J., Lohner, K., Dunn, Z., Cantwell, B., Zilliac, G.: Nitrous oxide hybrid rocket motor fuel regression rate characterization. Paper AIAA 2007–5352. AIAA, Reston VA USA (2007)
15. Dornheim, M.A.: Reaching 100 km. Aviat. Week Space Tec. **161**(6), 45–46 (2004)
16. Evans, B., Boyer, E., Kuo, K.: Hybrid rocket investigations at Penn State university's high pressure combustion laboratory: Overview and recent results. AIAA 2009–5346. AIAA, Reston VA USA (2009)
17. Farbar, E., Louwers, J., Kaya, T.: Investigation of metallized and nonmetallized hydroxyl terminated polybutadiene/hydrogen peroxide hybrid rockets. J. Prop. Power (2007). doi: 10.2514/1.22091
18. Gordon, S., McBride, B.J.: Computer program for calculation of complex chemical equilibrium compositions and applications. NASA RP-1311, NASA (1996)
19. Isakowitz, S.J., Hopkins, J.A., Hopkins, J.A. Jr.: International reference guide to space launch systems, 4th edn. AIAA, Reston VA USA (1994)
20. Karabeyoglu, M.A., Cantwell, B.J., Altman, D.: Development and testing of paraffin-based hybrid rocket fuels. AIAA-2001-4503. AIAA, Reston VA USA (2001)
21. Karabeyoglu, M.A., Altman, D., Cantwell, B.J.: Combustion of liquefying hybrid propellants: Part 1 general theory. J. Propul. Power (2002). doi: 10.2514/2.5975
22. Karabeyoglu, A., Zilliac, G., Cantwell, B., DeZilwa, S., Castellucci, P.: Scale-up tests of high regression rate paraffin-based hybrid rocket fuels. J. Propul. Power (2004). doi: 10.2514/1.2188
23. Knuth, W.H., Chiaverini, M.J., Sauer, J.A., Gramer, D.J.: Solid-fuel regression rate behavior of vortex hybrid rocket engines. J. Propul. Power (2002). doi: 10.2514/2.5974
24. Linden, D., Reddy, T.B. (ed.): Handbook of Batteries, 3rd edn. McGraw-Hill, New York (2002)
25. Maggi, F., Gariani, G., Galfetti, L., De Luca, L.T.: Theoretical analysis of hydrides in solid and hybrid rocket propulsion. Int. J. Hydrogen Energ. (2011). International Journal of Hydrogen Energy. http://dx.doi.org/10.1016/j.ijhydene.2011.10.018 **37**(2):1760–1769 (2012)
26. Nagata, H.: New Fuel Configurations for Advanced Hybrid Rockets. IAF-98-S.3.09. IAF, Paris (1998)
27. NASA: Turbopump systems for liquid rocket engines. NASA SP-8107, NASA (1974)
28. Peretz, A., Einav, O., Hashmonay, B.A., Birnholz, A., Sobe, Z.: Development of a laboratory-scale system for hybrid rocket motor testing. J. Propul. Power (2011). doi: 10.2514/1.47521
29. Rhee, I., Lee, C., Lee, J.W.: Optimal design for hybrid rocket engine for air launch vehicle. J. Mech. Sci. Technol. (2008). doi: 10.1007/s12206-008-0514-6
30. Rocker, M.: Modeling of nonacoustic combustion instability in simulations of hybrid motor tests. NASA/TP-000-209905, NASA (2000)

31. Saad, T., Majdalani, J.: Energy based mean flow solutions for slab hybrid rocket chambers. AIAA 2008–5021. AIAA, Reston VA USA (2008)
32. Sutton, G.P., Biblarz, O.: Rocket Propulsion Elements, 7th edn. Wiley, New York (2001)
33. Vonderwell, D.J., Murray, I.F., Heister, S.D.: Optimization of hybrid-rocket-booster fuel-grain design. J. Spacecraft Rockets (1995). doi: 10.2514/3.26716
34. Wernimont, E.H., Heister, S.D.: Combustion experiments in hydrogen peroxide/polyethylene hybrid with catalytic ignition. J. Propul. Power (2000) doi: 10.2514/2.5571

Chapter 15
Mathematical Models of Placement Optimisation: Two- and Three-Dimensional Problems and Applications

Yuri Stoyan and Tatiana Romanova

Abstract We study NP-hard placement optimisation problems, which cover a wide spectrum of industrial applications, including space engineering. This chapter considers tools of mathematical modelling and a solution strategy of placement problems illustrated with examples and pictures. A class of 2D and 3D geometric objects, called *phi*-objects, is introduced and considered as mathematical models of real objects. We review the main concept of our studies, i.e. *phi*-functions. One may also find a clear definition of *phi*-function as an analytical tool for describing placement constraints, including containment, non-overlapping, allowable distances, prohibited areas, object translations and rotations. A mathematical model of a basic placement problem is constructed as constrained optimisation problem. We propose a solution strategy for placement problems. The reader will get acquainted with an application problem of the basic placement problem encountered in space engineering and find a number of computational results for 2D and 3D applications.

Keywords Mathematical modelling • Packing and cutting • Phi-functions • Optimisation

15.1 Introduction

The placement problems in question, being a part of operational research and computational geometry, have multiple 2D and 3D applications. They can extensively be used in garment industry, sheet metal cutting, furniture making, shoe manufacturing for 2D case. Other engineering applications involve 3D geometry:

Y. Stoyan (✉) • T. Romanova
Department of Mathematical Modeling, Institute for Mechanical Engineering Problems of the National Academy of Sciences of Ukraine, Kharkov, Ukraine
e-mail: tarom27@yahoo.com

space engineering, mechanical engineering, car manufacturing, shipbuilding, 3D laser cutting, modelling granular media and liquids, radio-surgery treatment planning, medicine, materials science, nanotechnology, robotics, coding, pattern recognition systems, control systems, space apparatus control systems, etc.

The common placement problem lies in arranging a given set of objects within a given region (a container) in order to minimise waste of industrial materials, to minimise the use of space or maximise the number of objects, to minimise deviation from the centre of gravity, etc.

In spite of the variety of practical and scientific applications, the placement problems may be reduced to the following basic placement optimisation problem.

Basic Problem Place a set of geometric objects (from now on simply: objects) T_i, $i \in \{1, 2, ..., n\} = I_n$ into a container Ω, so that the given restrictions on the placement of the objects are fulfilled and the given objective function reaches its extreme value.

These restrictions include containment of objects into a container, non-overlapping of objects, given allowable distances between objects, rotations of objects, prohibited areas and other specific technological restrictions (static stability constraints, moments of inertia constraints, equilibrium constraint).

A multiplicity of shapes of T_i and Ω, as well as a variety of restrictions and forms of objective functions, create a wide spectrum of different subsequent problems of the basic one. We offer a universal approach to all the totality of such subsequent problems with the goal of developing efficient algorithms for solving placement problems.

These problems are NP-hard, and, as a result, solution methodologies generally employ heuristics (e.g., [1–4]). Some researchers develop systematic approaches based on mathematical modelling and general optimisation procedures, e.g. [5–7]. We refer the reader to recent tutorials [8, 9] presenting the history of placement problems and basic techniques for their solution. The most popular and most frequently cited tool in the modern literature on placement problems is the so-called no-fit polygon (e.g. [8, 10]); it works only for polygonal objects that can be translated without rotations. In a recent paper [1] the concept was also extended to objects bounded by circular arcs. Rotations of polygons are considered in [11]. In [12] the authors offer an implementation of Minkowski sum (we provide the definition below in Sect. 15.3) for objects bounded by line segments and circular arcs. There are many interesting algorithms which involve 3D geometry, among them [13–21] to mention a few. Paper [22] reviews layout algorithms for 3D objects.

The goal of this chapter is to present placement problems in a formal mathematical manner. In these studies we deal with objects (called *phi*-objects) and characterize their placements by means of special functions (called *phi*-functions). The concepts of the *phi*-object and the *phi*-function have their roots in topology, but *phi*-functions are very convenient for practical solution of placement problems. Our principal aim is to present here the *phi*-function technique and demonstrate practical benefits of the concept.

In this chapter the reader will find theoretical results earlier published in our works [23–29].

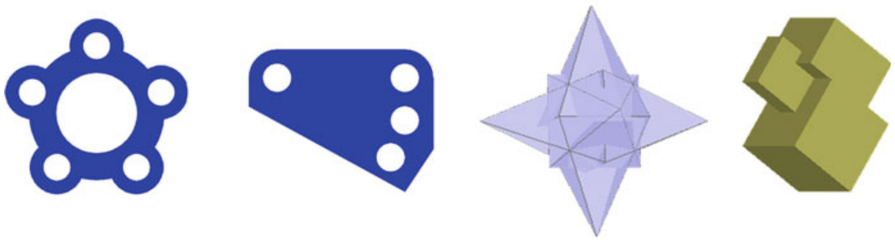


Fig. 15.1 2D and 3D *phi*-objects

This chapter is organised as follows: in Sect. 15.2, we introduce *phi*-objects; in Sect. 15.3, we define *phi*-functions and overview their properties; in Sect. 15.4, we show the use of *phi*-functions to reduce the placement problem to a constrained optimisation problem; and in Sect. 15.5, we describe an approach to its solution illustrated with some computational results.

15.2 *Phi*-Objects

In order to model mathematically real objects and their relations, we single out a class of the so-called *phi*-objects (e.g. [23, 24]) from a collection of all point subsets of R^2 or R^3.

Definition 1 Point set, $A \subset R^d$, $d = 2, 3$, is called a *phi*-object (Fig. 15.1) if it has the following characteristics:

1. A is canonically closed, i.e. $A = \mathrm{cl}^*A = \mathrm{clint}A$.
2. A has the same homotopic type as its interior.
3. For any point $z \in \mathrm{fr}A$, there exists an open neighbourhood U_z of z, such that $U_z \subset \mathrm{int}A$ is a connected set.

where int$A$, cl$A$, fr$A$ are the interior, the closure and the frontier of a *phi*-object A.

This can be explained intuitively as the point sets A, int$A$ and cl$A$ are of the same homotopic type. If they can be transformed into one another by continuous deforming, i.e., by bending, shrinking and expanding, then A is a *phi*-object. This definition excludes point sets with isolated points, deleted points or curves and nowhere dense point sets and objects with self-intersection of their frontiers. (The above definition can be stated for 2D objects by eliminating item 3.)

An important property of *phi*-objects is that if A is a *phi*-object, then the closure of its complement, i.e. $A^* = R^d \backslash \mathrm{int}A$, where $d = 2, 3$, is a *phi*-object, too.

Any *phi*-object in R^2 is called a *phi-polygon* if its frontier is shaped by straight lines, rays, and line segments. An ordinary polygon is a *phi*-polygon, but there are also unbounded *phi*-polygons — a half-plane, a cone, etc. (see illustrations in [23]). A *phi*-object in R^3 is called a *phi-polytope* if its frontier is shaped by *phi*-polygons.

Other objects can be approximated by polygons and polytopes, which is a common practice, but we handle some curvilinear objects directly. Hereinafter, we only deal with *phi*-objects.

Each object A may be explicitly given in R^d, $d = 2, 3$, by its space form c (shape of A), metric characteristics (sizes of A) $m = (m_1, m_2, ..., m_\mu)$ and placement parameters $u = (v, \theta)$ (a translation vector of the origin of eigen-coordinate system of A and rotation angles). Thus, a geometric information tuple $g = (c, m, u)$ generates object A in R^d, $d = 2, 3$. We suppose that each element of m and u may be variable. Further, we will use notation $A(m, u)$, where $m \in M_c \subset R^\mu$. The set M_c holds space form c of object A.

The class of *phi*-objects can be further divided into primary and composed objects. These two groups are distinguished to facilitate the generation of *phi*-functions.

15.2.1 Primary and Composed Objects

In most applications, the frontiers of 2D *phi*-objects are made by straight lines and circular arcs. The frontiers of 3D *phi*-objects mostly consist of flat sides, spherical, cylindrical and conical surfaces.

We call a primary *phi*-object in R^2 a circle (C) or a convex *phi*-polygon (K). In 3D, a primary object is a solid sphere (**S**), a convex polytope (**P**), a right circular cylinder (**C**) and a circular cone (**T**). In addition, if *phi*-object A is a 2D or a 3D primary object, then the closure of its complement (in R^2 or R^3, respectively), denoted by A*, is regarded as a primary object as well, where $A^* = R^d \backslash \mathrm{int} A$.

All other *phi*-objects that are a composition of a finite number of primary objects are called composed objects.

If we let A be a composed *phi*-object, then

$$A = (A_1 \circ {}_1 A_2 \circ_2 \cdots \circ_{k-1} A_k), \tag{15.1}$$

where A_i is a primary *phi*-object and $\circ_i \in \{\cup, \cap\}$.

Geometric information, generating object A formed by (15.1), is given by formula

$$g_A = (g_1 \circ_1 g_2 \circ_2 \cdots g_i \circ_i g_{i+1} \cdots \circ_{k-1} g_k, u), \tag{15.2}$$

where u is placement parameters of A and symbol $\circ_i \in \{\cup, \cap\}$ is understood in the following sense: $g_1 \cap g_2$ means that we consider $A_1 \cap A_2$, and $g_1 \cap \cdots \cap g_j \cup g_{j+1} \cap \cdots \cap g_k$ means that we take a union of objects $A_1 \cap \cdots \cap A_j$ and $A_{j+1} \cap \cdots \cap A_k$.

Indeed, every *phi*-object can be represented by unions and/or intersections of primary objects, according to our formula (15.1); see Fig. 15.1.

In 2D case, we went further. In paper [24] it has been proved that any arbitrary shaped *phi*-object A, formed by circular arcs and line segments, may always be

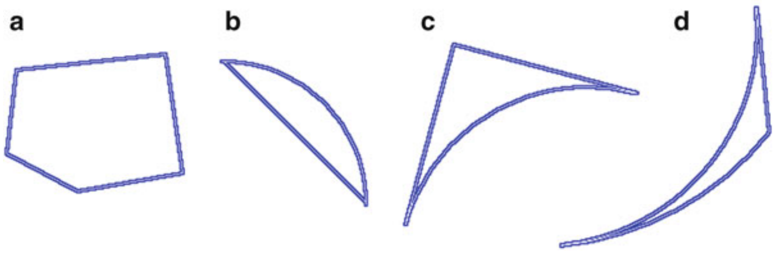


Fig. 15.2 Four types of basic 2D *phi*-objects: K, D, H and V

presented as a union of, the so-called, basic objects of four types (Fig. 15.2), i.e., $\circ_i \in \{\cup\}$, and A_i is a basic *phi*-object in (15.1). The detailed information about basic *phi*-objects and their decomposition is given in [24].

We give a convex *phi*-polygon (K) as an intersection of half-planes (Fig. 15.2a); a circular segment (D) as an intersection of a circle and a triangle T with two tangent sides to the circle C, i.e. $D = T \cap C$ (Fig. 15.2b); a "hat" (H) as an intersection of a triangle and the complement to a circle, i.e., $H = T \cap C^*$ (Fig. 15.2c); and a "horn" (V) as an intersection of a triangle, a (larger) circle and the complement to a (smaller) circle, i.e., $V = T \cap C_1 \cap C_2^*$ (Fig. 15.2d).

15.2.2 Relations Between Phi-Objects

We describe here relations between a pair of *phi*-objects considering the basic problem restrictions from the set-theoretical viewpoint.

Consider two objects $A(m_1, u_1)$ and $B(m_2, u_2)$. We differentiate four types of relations between the objects:

- *Interior intersecting*: When there are common interior points of $A(m_1, u_1)$ and $B(m_2, u_2)$, i.e., $\text{int}A(m_1, u_1) \cap \text{int}B(m_2, u_2) \neq \varnothing$.
- *Touching*: When there are common frontier points only of $A(m_1, u_1)$ and $B(m_2, u_2)$, i.e. $\text{int}A(m_1, u_1) \cap \text{int}B(m_2, u_2) \neq \varnothing$
  $\text{fr}A(m_1, u_1) \cap \text{fr}B(m_2, u_2) \neq \varnothing$.
- *Non-intersecting*: Where there are no common points of $A(m_1, u_1)$ and $B(m_2, u_2)$, i.e. $A(m_1, u_1) \cap B(m_2, u_2) = \varnothing$.
- *Containment*: The containment of one object into another, e.g. $A(m_1, u_1) \subset B$ (m_2, u_2), is equivalent to the interior non-intersecting of $A(m_1, u_1)$ and $B^*(m_2, u_2)$, i.e. $A(m_1, u_1) \subset B(m_2, u_2) \Leftrightarrow$
  $\text{int}A(m_1, u_1) \cap \text{int}B^*(m_2, u_2) = \varnothing$, $B^* = R^d \backslash \text{int}B$.

15.3 *Phi*-Functions

Let two objects A and B be given by their geometric information tuples $g_1 = (c_1, m_1, u_1)$ and $g_2 = (c_2, m_2, u_2)$.

Remark Hereinafter, we suppose that at least one of these objects (either A or B) is a bounded object. It comes from the basic problem statement.

Definition 2 Any everywhere defined continuous function $\Phi : R^{\xi} \to R^1$ is called a *phi*-function of objects $A(m_1, u_1)$ and $B(m_2, u_2)$, where $m_i \in M_i$, $i = 1{,}2$, if the following characteristics hold:

$$\begin{aligned}
&\Phi(m_1, u_1, m_2, u_2) > 0, \text{ if } A(m_1, u_1) \cap B(m_2, u_2) = \varnothing,\\
&\Phi(m_1, u_1, m_2, u_2) = 0, \text{ if } \begin{cases} \mathrm{fr}A(m_1, u_1) \cap \mathrm{fr}B(m_2, u_2) \neq \varnothing,\\ \mathrm{int}A(m_1, u_1) \cap \mathrm{int}B(m_2, u_2) = \varnothing,\end{cases}\\
&\Phi(m_1, u_1, m_2, u_2) < 0, \text{ if } \mathrm{int}A(m_1, u_1) \cap \mathrm{int}B(m_2, u_2) \neq \varnothing,
\end{aligned} \tag{15.3}$$

where $\xi = \mu + \eta$, $\eta = \eta_1 + \eta_2$, η_i is the number of elements of translation and rotation parameters of $A(m_1, u_1)$ and $B(m_2, u_2)$, $\eta_i \leq 3$, in 2D case, and $\eta_i \leq 6$, in 3D case, $\mu = \mu_1 + \mu_2$, μ_i is the number of metric characteristics of $A(m_1, u_1)$ and $B(m_2, u_2)$, $i = 1,\ 2$.

The function provides a value that indicates the state of the relations between the two objects, as described from the set-theoretical viewpoint in the previous section. Given the placement vector of each object, the *phi*-function will output a value of zero if the two objects touch, a positive value if the two objects are separated and a negative value if the two objects intersect. We further expect that the value of the *phi*-function should give at least an estimation of the distance between the objects when they are separated and some intersection "measure" when they are overlapped.

For the sake of simplicity, we further associate object $A(m_1, u_1)$ with A and *phi*-function $\Phi(m_1, u_1, m_2, u_2)$ for objects A and B with notation Φ^{AB}.

Suppose that objects A and B have fixed metrical characteristics and rotation angles, then *phi*-functions will possess the following useful features (for details of relative proves we refer the reader to [23–25]):

1. $\Phi^{AB}(v_1, v_2) = \Phi^{AB}(v_1 - v_2, 0) = \Phi^{AB}(0, v_2 - v_1)$. Since the object space forms and metrics stay constant, this condition deals only with the relative position of each object. In each of the three cases, the relative position of the two objects is identical.
2. The zero level of the *phi*-function is congruent ($\cong$) to the frontier of the Minkowski sum of A and $(-1)B$ provided one of the two objects has a fixed placement position, i.e. $\gamma_{12} = \{(0, v_2) \in R^{\xi} | \Phi(0, v_2) = 0\}$, $\gamma_{12} \cong \mathrm{fr}T_{12}$, where $T_{12}(v_2) = A(0) \oplus (-1)B(v_2)$ and $\gamma_{21} = \{(v_1, 0) \in R^{\xi} | \Phi(v_1, 0) = 0\}$, $\gamma_{21} \cong \mathrm{fr}T_{21}$, where $T_{21}(v_1) = B(0) \oplus (-1)A(v_1)$, $A \oplus B = C = \{c = a + b, a \in A, b \in B\}$ is Minkowski sum of

A and B. Note that $\gamma_{12} = \mathrm{fr}T_{12}$ if $T_{12} = A(0) \oplus (-1)B(0)$ and, $\gamma_{21} = \mathrm{fr}T_{21}$, where $T_{21} = B(0) \oplus (-1)A(0)$.
3. $\Phi^{AB}(v,0) = \Phi^{AB}(0,-v)$ and $\gamma_{12} = -\gamma_{21}$.
4. If A and B are centrally symmetric, then there exists a *phi*-function such that $\Phi^{AB}(v,0) = \Phi^{AB}(0,v)$.
5. Each *phi*-function for 2D *phi*-objects is a finite composition of minima and maxima of infinitely differentiable functions.
6. Each *phi*-function for 3D *phi*-objects is a finite composition of minima and maxima of piecewise-smooth functions.

15.3.1 Phi-Functions for Objects with Distance Constraints

In order to model given allowable distances between objects, we use normalised *phi*-functions (e.g. [23]) and adjusted *phi*-functions (e.g. [24, 25]).

Let the allowable distance ρ between *phi*-objects A and B be given, i.e.

$$\mathrm{dist}(A,B) \geq \rho^- \text{, if } \rho = \rho^- \text{ and } \mathrm{dist}(A,B) \leq \rho^+\text{, if } \rho = \rho^+, \tag{15.4}$$

where ρ^- (ρ^+) is minimal (maximal) allowable distance ρ between *phi*-objects A and B. Here, $\mathrm{dist}(A,B) = \min\limits_{a \in A, b \in B} d(a,b)$, $d(a,b)$ is the Euclidean distance between two points a and b in R^n.

Definition 3 A *phi*-function Φ^{AB} is said to be normalised if its values are equal to dist(A,B), subject to $\mathrm{int}A \cap \mathrm{int}B = \varnothing$.

We denote the ρ-level surface of the normalised *phi*-function by γ^ρ.

For example, surfaces $\gamma^\rho_{12} = \{u_2 \in R^3 \,|\tilde{\Phi}(0,u_2) = \rho, \rho \in R^+\}\}$ for pairs **C** and **T**, **P**$_1$ and **P**$_2$, **P** and **T**, **C**$_1$ and **C**$_2$, **P** and **C**, **T**$_1$ and **T**$_2$ are shown in Fig. 15.3. In terms of *phi*-functions, constraints dist(A,B) $\geq \rho^-$ and dist(A,B) $\leq \rho^+$ may be specified in the form $\widetilde{\Phi}^{AB} \geq \rho^-$ and $0 \leq \widetilde{\Phi}^{AB} \leq \rho^+$, respectively, where $\widetilde{\Phi}^{AB}$is the normalised *phi*-function of objects A and B.

However, normalised *phi*-functions inevitably involve radicals, which are sometimes unfriendly with gradient optimisation methods. That is why we offer here an alternative way to take into account the distance constraints.

Definition 4 An everywhere defined function $\overset{\frown}{\Phi}^{AB}$ is called an adjusted (pseudonormalised) *phi*-function of A and B if the following characteristics hold true:

$$\begin{aligned} \overset{\frown}{\Phi}^{AB} &> 0, \text{ if } \mathrm{dist}(A,B) > \rho, \\ \overset{\frown}{\Phi}^{AB} &= 0, \text{ if } \mathrm{dist}(A,B) = \rho, \\ \overset{\frown}{\Phi}^{AB} &< 0, \text{ if } \mathrm{dist}(A,B) < \rho. \end{aligned} \tag{15.5}$$

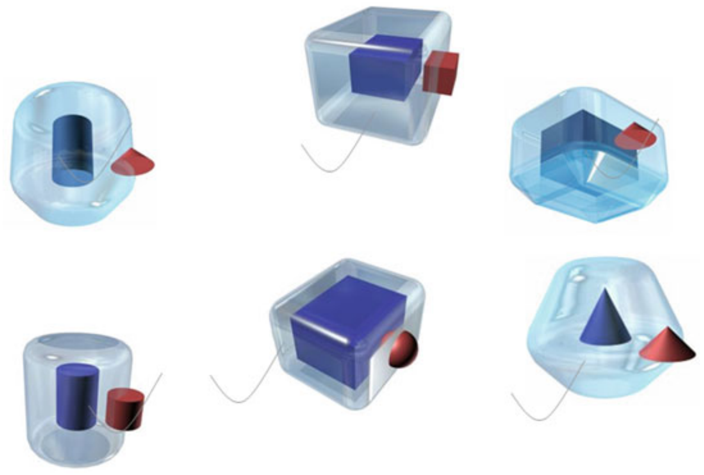


Fig. 15.3 γ^{ρ} illustrated by transparent surfaces for pairs of 3*D* primary objects

In particular, we have

$$\mathrm{dist}(A,B) \geq \rho^{-} \Leftrightarrow \widetilde{\Phi}^{AB} \geq \rho^{-} \Leftrightarrow \overset{\frown}{\Phi}^{AB} \geq 0, \tag{15.6}$$

$$\mathrm{dist}(A,B) \leq \rho^{+} \Leftrightarrow 0 \leq \widetilde{\Phi}^{AB} \leq \rho^{+} \Leftrightarrow \left\{ \breve{\Phi}^{AB} = \min\{\Phi^{AB}, -\widetilde{\Phi}^{AB}\} \right\} \geq 0,$$

$$\text{Thus, } \widetilde{\Phi}^{AB} = \rho \text{ implies } \overset{\frown}{\Phi}^{AB} = 0.$$

15.3.2 Phi-Functions for Primary and Composed Objects

We offer a complete class of *phi*-functions for all combinations of oriented 2D and 3D primary objects considering non-overlapping and containment constraints in [23, 26, 27, 29] and also give a technique of constructing *phi*-functions for composed objects in [23, 28]. In addition, we provide a complete class of ready-to-use radical-free *phi*-functions for all combinations of 2D basic objects, considering translations, rotations and homothetic transformations of the objects. The class of *phi*-functions covers all realistic cases of 2D objects mentioned above. One can find *phi*-functions for 2D basic objects in the recent works [24, 25].

Below we give *phi*-functions for some of the basic objects. We assume that each point $(x_0, y_0) \in A$ in the eigen-coordinate system of A is transformed into the point (x, y) $x = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta + x_t$, $y = -x_0 \cdot \sin\theta + y_0 \cdot \cos\theta + y_t$; here θ is a rotation parameter; (x_t, y_t) is a translation vector of A.

Convex Polygons K_1 and K_2 Let $(x_i^{'}, y_i^{'})$, $i = 1, 2, ..., m_1$, be vertices of K_1, and $(x_j^{''}, y_j^{''})$, $j = 1, 2, ..., m_2$, be vertices of K_2; $\varphi_i = \alpha_i^{'} x + \beta_i^{'} y + \gamma_i^{'} = 0$, and $\Psi_j = \alpha_j^{''} x + \beta_j^{''} y + \gamma_j^{''} = 0$ be side equations of K_1 and K_2, subject to $\varphi_i \le 0$ and $\Psi_j \le 0$ for all points belonging to K_1 and K_2, respectively, then

$$\Phi^{K_1 K_2} = \max\{\max_{1 \le i \le m_1} \min_{1 \le j \le m_2} \varphi_{ij}, \max_{1 \le j \le m_2} \min_{1 \le i \le m_1} \Psi_{ji}\}, \tag{15.7}$$

$$\text{where } \varphi_{ij} = \alpha_i^{'} x_j^{''} + \beta_i^{'} y_j^{''} + \gamma_i^{'} \text{ and } \Psi_{ji} = \alpha_j^{''} x_i^{'} + \beta_j^{''} y_i^{'} + \gamma_j^{''}.$$

Circles C_1 and C_2 Let (x_{C_i}, y_{C_i}) be the centre point and r_{C_i} be a radius of C_i, $i = 1, 2$, then

$$\Phi^{C_1 C_2} = \omega = (x_{C_1} - x_{C_2})^2 + (y_{C_1} - y_{C_2})^2 - (r_{C_1} + r_{C_2})^2. \tag{15.8}$$

Circle C and Convex Polygon K Let (x_i, y_i), $i = 1, 2, ..., m$, be vertices; $\chi_i = \alpha_i x + \beta_i y + \gamma_i = 0$ beside equations of K; (x_C, y_C) be the centre point and r — a radius of C , then

$$\Phi^{CK} = \max\{\chi_i, \min\{\omega_i, \Psi_i\}, i = 1, 2, ..., m\}, \tag{15.9}$$

$$\chi_i = \alpha_i x + \beta_i y + \gamma_i - r,\ \alpha_i^2 + \beta_i^2 = 1,$$

$$\omega_i = (x_C - x_i)^2 + (y_C - y_i)^2 - r^2,$$

$$\Psi_i = (\beta_{i-1} - \beta_i)(x_C - x_i) - (\alpha_{i-1} - \alpha_i)(y_C - y_i) + r(\alpha_{i-1}\beta_i - \alpha_i\beta_{i-1}).$$

Circular Segment D and Convex Object E Let $D = T \cap C^{'}$, where T is a triangle, $C^{'}$ is a circle and $E \in \{C, K, D\}$, then

$$\Phi^{DE} = \max\{\Phi^{TE}, \Phi^{C^{'}E}\} \tag{15.10}$$

where $\Phi^{TE}, \Phi^{C^{'}E}$ are given by (15.7)–(15.9).

For constructing *phi*-functions for concave basic objects, we use the following *phi*-function.

Object C^* and Circular Segment D Let $D = T \cap C_D$, where T is a triangle, C_D is a circle of radius r_D with the centre point (x_D, y_D), providing two sides of T tangent to C_D at endpoints (x_i, y_i), $i = 1, 2$.

If $r_D < r_C$, then

$$\Phi^{C^* D} = \min\{\Psi_0, \max\{\Phi^{C^* C_D}, \chi_1, -\chi_2\}\}, \tag{15.11}$$

where

$$\Psi_0 = \min_{i=1,2} (r_C{}^2 - (x_C - x_i)^2 - (y_C - y_i)^2),$$

$$\chi_i = (x_D - x_C)(y_i - y_D) - (y_D - y_C)(x_i - x_D), \ i = 1, 2.$$

If $r_D \geq r_C$, then we use $\Phi^{C^*D} = \Psi_0$.

We also consider some of the *phi*-functions for a pair of 3D primary objects.

Spheres $\mathbf{S}_1$ *and* $\mathbf{S}_2$ Let $m_1 = (r_1)$, $m_2 = (r_2)$, then $\Phi^{\mathbf{S}_1\mathbf{S}_2} = \varphi$, where $\varphi = x^2 + y^2 + z^2 - (r_1 + r_2)^2$, here and later on: $(x, y, z) = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$.

The normalised *phi*-function has the form $\widetilde{\Phi}^{\mathbf{S}_1\mathbf{S}_2} = \tilde{\varphi}$, where $\tilde{\varphi} = \sqrt{x^2 + y^2 + z^2} - (r_1 + r_2)$.

Parallelepipeds $\mathbf{P}_1$ *and* $\mathbf{P}_2$ Let $m_1 = (a_1, b_1, h_1)$, $m_2 = (a_2, b_2, h_2)$ and $A = a_1 + a_2$ $B = b_1 + b_2, H = h_1 + h_2$, then $\Phi^{\mathbf{P}_1\mathbf{P}_2} = \chi$, where

$$\chi = \max_{i=1,\dots,6} \chi_i, \tag{15.12}$$

$$\chi_1 = x - A, \chi_2 = y - B, \chi_3 = -x - A,$$

$$\chi_4 = -y - B, \chi_5 = z - H, \chi_6 = -z - H. \tag{15.13}$$

Parallelepiped $\mathbf{P}$ *and Sphere* $\mathbf{S}$ Let $m_1 = (a, b, h)$, $m_2 = (r)$, and let $\chi_i(x, y, z)$, $i = 1, 2, \dots, 6$, be given by (13), where $A = a + r$, $B = b + r$, $H = h + r$, then the normalised *phi*-function has the form $\widetilde{\Phi}^{\mathbf{PS}} = \omega$, where

$$\omega = max\{\chi, \zeta, \tau\},$$

$$\zeta = \max_{i=1,\dots 8} \zeta_i, \ \tau = \max_{i=1,\dots 12} \tau_i, \tag{15.14}$$

$$\zeta_i = \min\{\phi_i, \tilde{\chi}_i, \widetilde{\phi}_{ij}, j = 1, 2, 3\}, \ \tau_i = \min\{\Psi_i, \tilde{\tilde{\chi}}_i\},$$

$$\tilde{\chi}_i = \varphi_x + \varphi_y + \varphi_z - d, \tag{15.15}$$

$$\phi_i = \sqrt{\varphi_x^2 + \varphi_y^2 + \varphi_z^2} - r, \ \tilde{\phi}_{i1} = \sqrt{\varphi_x^2 + \varphi_y^2} + \varphi_z - r,$$

$$\tilde{\phi}_{i2} = \sqrt{\varphi_y^2 + \varphi_z^2} + \varphi_x - r, \ \tilde{\phi}_{i3} = \sqrt{\varphi_x^2 + \varphi_z^2} + \varphi_y - r;$$

$$\tilde{\tilde{\chi}}_{j_1} = \varphi_x + \varphi_y - d_1, \ \tilde{\tilde{\chi}}_{j_2} = \varphi_y + \varphi_z - d_2, \ \tilde{\tilde{\chi}}_{j_3} = \varphi_x + \varphi_z - d_3, \tag{15.16}$$

$$\Psi_{j_1} = \sqrt{\varphi_x^2 + \varphi_y^2} - r,\ \Psi_{j_2} = \sqrt{\varphi_y^2 + \varphi_z^2} - r,\ \Psi_{j_3} = \sqrt{\varphi_x^2 + \varphi_z^2} - r,$$

$$\varphi_x = (-1)^{k_x} \cdot x - a,\ \varphi_y = (-1)^{k_y} \cdot y - b,\ \varphi_z = (-1)^{k_z} \cdot z - h, \tag{15.17}$$

where $d = r, d_1 = d_2 = d_3 = d, i = k_x + 2k_y + 4k_z + 1, j_1 = k_x + 2k_y + 1, j_2 = k_y + 2k_z + 5, j_3 = k_x + 2k_z + 9, k_x, k_y, k_z \in I_2, I_2 = \{0, 1\}$.

Remark If we assume $r = 0, a = a_1 + a_2, b = b_1 + b_2, h = h_1 + h_2, d = a + b + h, d_1 = a + b, d_2 = b + h, d_3 = a + h$ in (15.15)–(15.17), then we get the normalised *phi*-function for parallelepipeds $\mathbf{P}_1$ and $\mathbf{P}_2$.

Cylinders $\mathbf{C}_1$ *and* $\mathbf{C}_2$ Let $m_1 = (r_1, h_1), m_2 = (r_2, h_2)$ and $R = r_1 + r_2, H = h_1 + h_2$. We assume $f = \sqrt{x^2 + y^2}$ (here and later on), then $\Phi^{\mathbf{C}_1\mathbf{C}_2} = \chi$, where $\chi = \max_{i=1,2,3} \chi_i$ and

$$\chi_1 = z - H, \chi_2 = -z - H, \chi_3 = f - R. \tag{15.18}$$

The normalised *phi*-function is defined as follows: $\widetilde{\Phi}^{\mathbf{C}_1\mathbf{C}_2} = \omega$, where

$$\omega = max\{\chi, \zeta\},\ \zeta = \max_{i=1,2} \zeta_i,\ \zeta_i = min\{\Psi_i, \phi_i\}, \tag{15.19}$$

$$\phi_1 = f + z - d, \phi_2 = f - z - d,\ d = R + H, \tag{15.20}$$

$$\Psi_1 = \sqrt{(f - R)^2 + (z - H)^2}, \Psi_2(u) = \sqrt{(f - R)^2 + (z + H)^2}.$$

Cylinder $\mathbf{C}$ *and Sphere* $\mathbf{S}$ Let $m_1 = (r_1, h), m_2 = (r_2)$, and functions $\phi_i, i = 1, 2$, are given by (15.20), where $H = h + r_2,\ \ R = r_1 + r_2,\ \ d = h + R$, then the normalised *phi*-function has the form

$$\widetilde{\Phi}^{\mathbf{CS}} = \omega, \tag{15.21}$$

where ω is defined by (15.19), $\chi = \max_{i=1,2,3} \chi_i$, $\chi_i(u)$ is given by (15.18), and $\Psi_1 = \sqrt{(f - r_1)^2 + (z - h)^2} - r_2,\ \Psi_2 = \sqrt{(f - r_1)^2 + (z + h)^2} - r_2.$

Cylinder $\mathbf{C}$ *and Parallelepiped* $\mathbf{P}$ Let $m_1 = (a, b, h_1), m_2 = (r, h_2)$ and $H = h_1 + h_2, A = a + r, B = b + r,\ d_1 = a + b + r, d = d_1 + H$. And let χ and $\chi_i, i = 1, 2, ..., 6$, be given by (15.12) and (15.13), respectively. Assuming $\tilde{\tilde{\chi}}_i = \varphi_x + \varphi_y - d_1$ and $\Psi_i = \sqrt{\phi_x^2 + \phi_y^2} - r$, where φ_x, φ_y are specified by (15.17), $j = k_x + 2k_y + 1$, $k_x, k_y \in I_2, I_2 = \{0, 1\}$, we set $\Phi^{\mathbf{CP}} = \omega$, where $\omega = \max\{\chi, \lambda\}$, $\lambda = \max_{i=1,2,...,4} \lambda_i$, $\lambda_i = \min\{\Psi_i, \tilde{\tilde{\chi}}_i\}$. Then $\tilde{\Phi}^{\mathbf{CP}} = \tilde{\omega}$ is the normalised *phi*-function, where $\tilde{\omega} = \max\{\chi, \zeta, \tau\}$, $\varsigma = \max_{i=1,2,...,8} \varsigma_i$, $\zeta_i = \min\{\phi_i, \tilde{\chi}_i, \widetilde{\phi}_{ij}, j = 1, 2, 3\}$, $\tau = \max_{i=1,2,...,12} \tau_i$, $\tau_i = \min\{\Psi_i, \tilde{\tilde{\chi}}_i\}, \tilde{\chi}_i = \phi_x + \phi_y + \phi_z - d,$

$$\phi_i = \sqrt{\left(\sqrt{\varphi_x^2+\varphi_y^2}-r\right)^2+\phi_z^2},\ \tilde{\phi}_{i1} = \sqrt{\varphi_x^2+\varphi_y^2}+\phi_z-r,$$

$$\tilde{\phi}_{i2} = \sqrt{\phi_y^2+\phi_z^2}+\varphi_x-r,\ \tilde{\phi}_{i3} = \sqrt{\phi_x^2+\phi_z^2}+\varphi_y-r,$$

$$\Psi_{j_1} = \sqrt{\varphi_x^2+\varphi_y^2}-r,\ \Psi_{j_2} = \sqrt{\phi_y^2+\phi_z^2}-r,\ \Psi_{j_3} = \sqrt{\phi_x^2+\phi_z^2},$$

$$\phi_x = (-1)^{k_x}\cdot x - A,\ \phi_y = (-1)^{k_y}\cdot y - B,\ \phi_z = (-1)^{k_z}\cdot z - H, \qquad (15.22)$$

φ_x, φ_y, φ_z are given by (15.17), $i = k_x + 2k_y + 4k_z + 1, j_1 = k_x + 2k_y + 1,$ $j_2 = k_y + 2k_z+5, j_3 = k_x + 2k_z + 9, k_x, k_y, k_z \in I_2, I_2 = \{0, 1\}$.

The research of normalised *phi*-functions revealed an interesting result. We formulate the result as the following assertion.

Let $\tilde{\Phi}^{AB}$ be the normalised *phi*-function for a pair of primary objects $A(m_1, u_1)$ and $B(m_2, u_2)$. And let $\widehat{A}(\widehat{m}_1, 0) = A(m_1, 0) \oplus C(r_1, 0)$, $\widehat{B}(\widehat{m}_2, 0) = B(m_2, 0) \oplus C(r_2, 0)$ for 2D case and $\widehat{A}(\widehat{m}_1, 0) = A(m_1, 0) \oplus \mathbf{S}(r_1, 0)$, $\widehat{B}(\widehat{m}_2,0) = B(m_2,0) \oplus \mathbf{S}(r_2,0)$ in 3D case, where $\widehat{m}_i = (m_i, r_i)$, $i = 1,2$. Then $\tilde{\Phi}^{\widehat{A}\widehat{B}} = \tilde{\Phi}^{AB} + r_1 + r_2$ is the normalised *phi*-function for $\widehat{A}(\widehat{m}_1, u_1)$ and $\widehat{B}(\widehat{m}_2, u_2)$. In addition, $\tilde{\Phi}^{\widehat{A}\widehat{B}}$ produces a family of normalised *phi*-functions $\tilde{\Phi}^{\tilde{A}\tilde{B}}$ of all pairs of objects $\tilde{A}(\tilde{m}_1, u_1)$ and $\tilde{B}(\tilde{m}_2, u_2)$, which appear immediately from objects $\widehat{A}(\widehat{m}_1, u_1)$ and $\widehat{B}(\widehat{m}_2, u_2)$ by degenerating their metrical characteristics $\tilde{m}_i \subset \widehat{m}_i$, i.e. $\tilde{\Phi}^{\tilde{A}\tilde{B}} = \tilde{\Phi}^{\widehat{A}\widehat{B}|_{\tilde{m}_i \subset \widehat{m}_i}}$.

For example, the normalised *phi*-function for an oriented rectangle R with metrical characteristics $m = (a, b)$ and a circle C of radius r with the centre point (x_C, y_C) has the form

$$\tilde{\Phi}^{CR} = \max\{\chi_i, \min\{\omega_i, \Psi_i\}, i = 1, ..., 4\}, \qquad (15.23)$$

where, assuming that $A = a + r$, $B = b + r$, $s = a + b + r$, we define

$$\omega_i = \sqrt{(x_C - x_i)^2 + (y_C - y_i)^2} - r,$$

$$\chi_1 = x - A,\ \chi_2 = y - B,\ \chi_3 = -x - A,\ \chi_4 = -y - B,$$

$$\Psi_1 = x + y - s,\ \Psi_2 = -x + y - s,\ \Psi_3 = -x - y - s,\ \Psi_4 = x - y - s.$$

We may model the relations of all combinations of pairs of objects having shape from the family of space forms $\{R, C, A, A_a, A_b\}$ with metric characteristics $m = (a, b, 0)$, $m = (0, 0, r)$, $m = (a, b, r)$, $m = (a, 0, r)$, $m = (0, b, r)$, respectively,

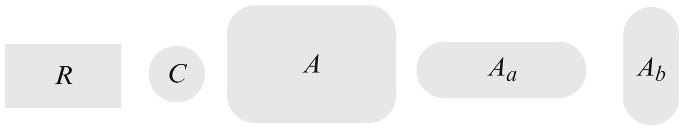


Fig. 15.4 The family of object space forms generated by $A = R(0) \oplus C(0)$

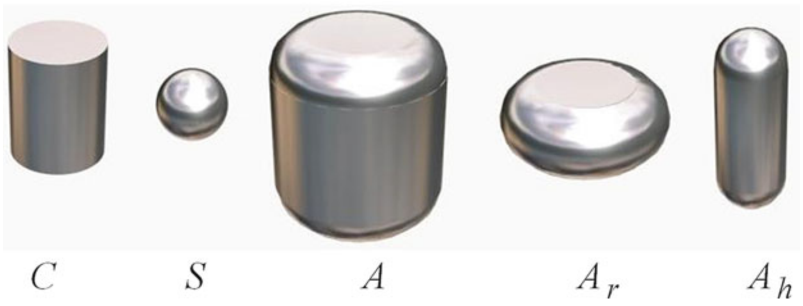


Fig. 15.5 The family of object space forms generated by $\mathbf{A} = \mathbf{C}(0) \oplus \mathbf{S}(0)$

using *phi*-function $\tilde{\Phi}^{CR}$. The family is generated by $A = R(0) \oplus C(0)$ and shown in Fig. 15.4. We refer the reader to [23] for details.

The similar generalisation can be made for normalised phi-functions for 3D primary objects. For example, using the normalised *phi*-function $\tilde{\Phi}^{\mathbf{CS}}$ given by (21) for a cylinder $\mathbf{C}$ with metric characteristics $m = (r_1, h)$ and a sphere $\mathbf{S}$ of radius r_2, we may model relations for all combinations of pairs of objects having shapes from the family of space forms $\{\mathbf{C}, \mathbf{S}, \mathbf{A}, \mathbf{A}_r, \mathbf{A}_h\}$ with metric characteristics $m = (r_1, h, 0)$, $m = (0, 0, r_2)$, $m = (r_1, h, r_2)$, $m = (r_1, 0, r_2)$, $m = (0, h, r_2)$, respectively. The family is generated by $\mathbf{A} = \mathbf{C}(0) \oplus \mathbf{S}(0)$ and shown in Fig. 15.5.

15.3.3 Phi-Functions for Composed Objects

Phi-function for composed objects operates with *phi*-functions of 2D basic (or primary) and 3D primary objects. The technique of constructing *phi*-functions for composed objects is given in details in [23, 28].

Let objects A and B be given in the form

$$A = A_1 \cup \ldots \cup A_{n'}\,,\ B = B_1 \cup \ldots \cup B_{n''}, \tag{15.24}$$

where A_i and B_j are primary or basic objects. Then in these cases, *phi*-function for A and B has the form

$$\Phi^{AB} = \min\{\Phi_{ij}, i = 1, 2, ..., n', j = 1, 2, ..., n''\}, \tag{15.25}$$

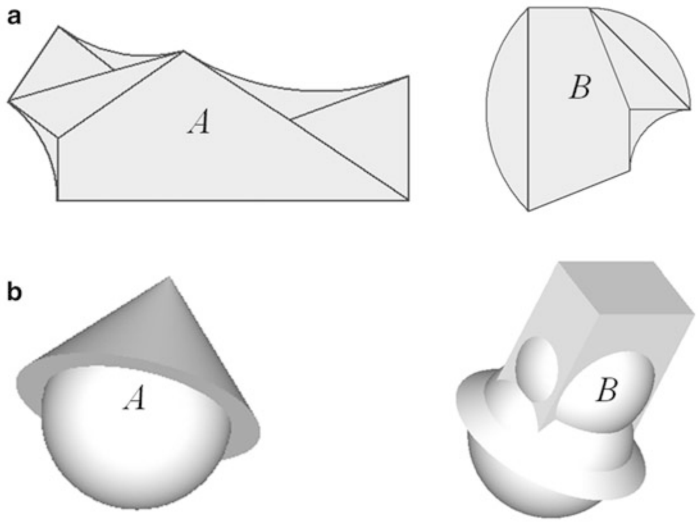


Fig. 15.6 Two composed objects: (**a**) formed by 2D basic objects; (**b**) formed by 3D primary objects

where Φ_{ij} is a *phi*-function for objects A_i and B_j.

Let us consider an example of pairs of composed objects specified by (15.24) and shown in Fig. 15.6, for which *phi*-functions are defined by formula (15.25). In order to decompose a 2D composed object into basic ones we use special algorithm described in [24].

Further, let A be a composed object (with holes) of the non-empty intersection of object A_0 and objects A_i^*, $i = 1, 2, ..., n$. And let B be a bounded *phi*-object, then

$$\Phi^{AB} = \max\{\Phi^{A_0B}, \Phi^{A_i^*B}, i = 1, 2, ..., n\}. \tag{15.26}$$

We illustrate the case with Fig. 15.7a, where A_0 is an oval, $A_i^* = R^2 \setminus \text{int}C_i$, $i = 1, 2, 3$, C_i are circles and B is a convex polygon**.** We apply formula (15.26) and derive *phi*-function for A and B, i.e. $\Phi^{AB} = \max\{\Phi^{A_0B}, \Phi^{A_1^*B}, \Phi^{A_2^*B}, \Phi^{A_3^*B}\}$.

Indeed, $\Phi^{AB} \geq 0$ if one of $\Phi^{A_i^*B} \geq 0$ (polygon B does not overlap A_i^*; it means that $B \subset C_i$; see Fig. 15.7a), or $\Phi^{A_0B} \geq 0$ (object B does not overlap A_0; see Fig. 15.7b).

Now, let A be a composed 2D object of the non-empty intersection of two primary one-connected objects A_1 and A_2, and let B be a bounded *phi*-object. For instance, $A = P \cap C$ and $B = K$, where P is a half-plane and C is a circle**.** We arrange A and B as it is shown in Fig. 15.8a**.** One can see that (it means that $\Phi^{PK} < 0$) and $C \cap K \neq \varnothing$ (it means that $\Phi^{CK} < 0$), while $A \cap K = \varnothing$ (it means that Φ^{AB} is going to be positive)**.** Therefore, $\Phi^{AB} \neq \max\{\Phi^{RK}, \Phi^{C^*K}\}$. We refer the reader to [28] for set-theoretical proofs of the situation.

However, if we take $A = T \cap C$, where T is a triangle with two tangent sides to the circle C (see Fig. 15.8b), then

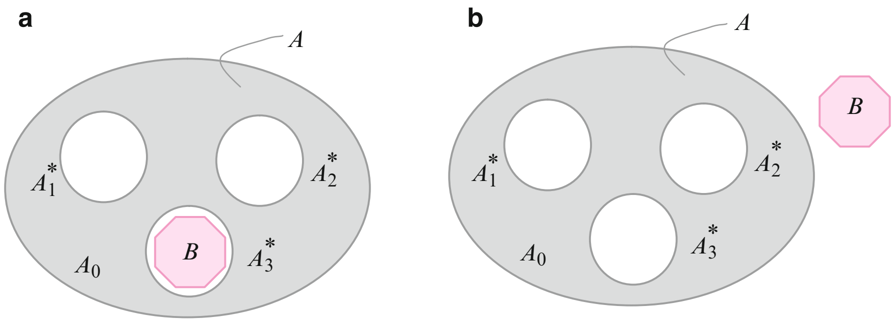


Fig. 15.7 Instance for object A with holes: (**a**) polygon B does not overlap object A_3^*; (**b**) object B does not overlap oval A_0

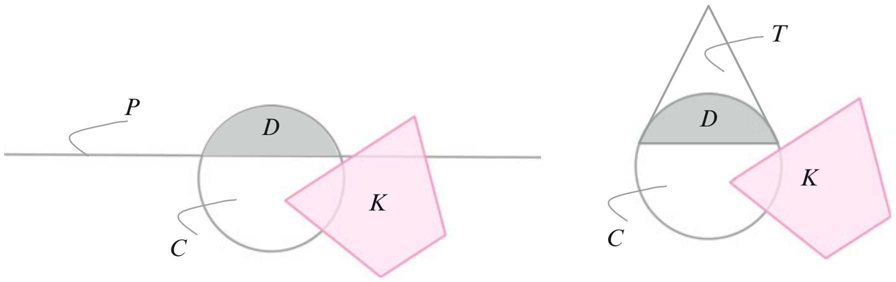


Fig. 15.8 Instance for circular segment **D** and convex polygon K: (**a**) $\mathbf{D} = \mathbf{C} \cap \mathbf{P}$, (**b**) $\mathbf{D} = \mathbf{C} \cap \mathbf{T}$

$$\Phi^{AB} = \max\left\{\Phi^{TK},\ \Phi^{CK}\right\}. \tag{15.27}$$

By analogy, we give 3D instance. Let object **D** be a sphere segment presented in the form $\mathbf{D} = \mathbf{S} \cap \mathbf{T}$ (see Fig. 15.9a), then

$$\Phi^{\mathbf{D}B} = \max\{\Phi^{\mathbf{S}B}, \Phi^{\mathbf{T}B}\}, \tag{15.28}$$

where B is a composed 3D object (e.g. see Fig. 15.9b).

In this case, we make rigid demands to primary object **T** such that the cone generator is *tangent* to **S**; otherwise, function (15.28) will not be a *phi*-function.

Let us consider "mixed" case of union and intersection, e.g. $A = (A_1 \cup A_2) \cap A_3^*$, subject to $\mathrm{fr}(A_1 \cup A_2) \cap frA_3^* = \varnothing$, then for any 2D-object B we may apply the following formula:

$$\Phi^{AB} = \max\{\min\{\Phi^{A_1B}, \Phi^{A_2B}\}, \Phi^{A_3^*B}\}. \tag{15.29}$$

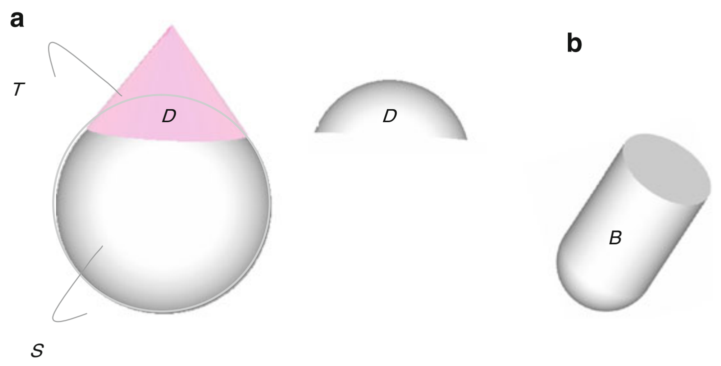


Fig. 15.9 Instance for sphere segment $\mathbf{D}$ = and object B: (**a**) $\mathbf{D} = \mathbf{S} \cap \mathbf{T}$, (**b**) 3D-object B

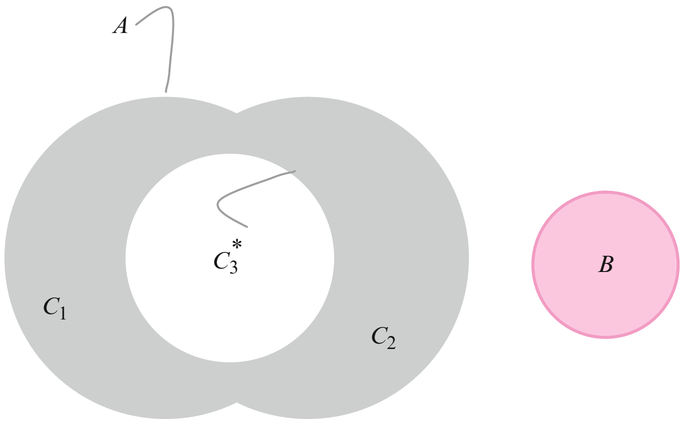


Fig. 15.10 Instance for "mixed" case: object $A = (C_1 \cup C_2) \cap C_3^*$ and object B

Instance for "mixed" case: object $A = (C_1 \cup C_2) \cap C_3^*$ and $B = C$ (see Fig. 15.10), then phi-function of objects A and B has the form (15.29), i.e. $\Phi^{AB} = \max\{\min\{\Phi^{C_1C}, \Phi^{C_2C}\}, \Phi^{C_3^*C}\}$.

The way for constructing a *phi*-function for general case of composed objects one can find in [23, 28].

15.4 Mathematical Model of the Basic Problem

In terms of *phi*-functions, the placement constraints mentioned above can be presented in the following form:(1) *containment* of T_i into Ω^*: $\Phi_i^* \geq 0$, where Φ_i is a *phi*-function of objects T_i and Ω^*; (2) *non-overlapping* of T_i and T_j: $\Phi_{ij} \geq 0$, where Φ_{ij} is a *phi*-function of objects T_i and T_j; (3) *distance constraints* for Ω^* and T_i: (a) on *minimal allowable distance*: $\overset{\frown}{\Phi}{}_i^* \geq 0$; (b) on *maximal allowable distance*: $\overset{\smile}{\Phi}{}_i^* = \min\{\Phi_i^*, -\overset{\frown}{\Phi}{}_i^*\} \geq 0$, where $\overset{\frown}{\Phi}{}_i^*$ is an adjusted *phi*-function of objects T_i and Ω^*; and (4) *distance constraints* for T_i and T_j: (a) on *minimal allowable distance* : $\overset{\frown}{\Phi}_{ij} \geq 0$; (b) on *maximal allowable distance*: $\overset{\smile}{\Phi}_{ij} = \min\{\Phi_{ij}, -\overset{\frown}{\Phi}_{ij}\} \geq 0$, where $\overset{\frown}{\Phi}_{ij}$ is an adjusted *phi*-function of objects T_i and T_j.

Taking into account (1)–(4), we present the basic problem as a constrained optimisation problem:

$$\operatorname{extr} F(U) \text{ s.t. } u \in W \subset R^{\mu+\xi}, \tag{15.30}$$

where $F(U)$ is an objective function, $U = (m, u)$, $m = (m_0, m_1, m_2, \ldots, m_n) \in R^{\mu}$, $u = (u_0, u_1, u_2, \ldots, u_n) \in R^{\xi}$, m is a vector of variable metrical characteristics, u is a vector of variable placement parameters, where solution space W is specified as follows:

$$W = \{(U) \in R^{\mu+\xi} : \Lambda \geq 0, \Lambda^* \geq 0\}, \tag{15.31}$$

where $\Lambda = \min\{\Lambda_{ij}, i<j = 1, 2, \ldots, n\}$, $\Lambda^* = \min\{\Lambda_i^*, i = 1, 2, \ldots, n\}$,

$$\Lambda_{ij} = \min\{\overset{\frown}{\Phi}_{ij}, \overset{\smile}{\Phi}_{ij}\},\ \Lambda_i^* = \min\{\overset{\frown}{\Phi}{}_i^*, \overset{\smile}{\Phi}{}_i^*\}. \tag{15.32}$$

If allowable distances are not given, functions in (15.32) take the form

$$\Lambda_{ij} = \Phi_{ij},\ \Lambda_i^* = \Phi_i^*. \tag{15.33}$$

Let us consider the basic characteristics of solution space W given by (15.31):

1. The solution space W is a disconnected set. Each connected component of W may have a complicated structure; in particular, it may have multiple holes and cavities.
2. The solution space W can be naturally represented as $W = \bigcup\limits_{j=1}^{J} W_j$, where each W_j is specified by a system of inequalities resulting from our *phi*-functions. It should be noted that J (the number of W_j's) may be huge.
3. The frontier of W is usually made of non-linear surfaces containing valleys, ravines, etc.
4. Our constrained optimisation problem is NP-hard and always has multiple local extrema.

The following example will clarify item 2. Assume *phi*-function for objects A and B has the form $\Phi^{AB} = \min\{\min\{f_1, f_2\}, \max\{\varphi_1, \varphi_2, \varphi_3\}, \max\{\tau_1, \tau_2\}\}$.

Then $\Phi^{AB} \geq 0$ if $\min\{f_1, f_2\} \geq 0$ and $\max\{\varphi_1, \varphi_2, \varphi_3\} \geq 0$ and $\max\{\tau_1, \tau_2\} \geq 0$. Therefore, $\Phi^{AB} \geq 0$ generates the following collection of inequality systems:

$$\begin{cases} f_1 \geq 0 \\ f_2 \geq 0 \\ \varphi_1 \geq 0 \\ \tau_1 \geq 0 \end{cases}, \begin{cases} f_1 \geq 0 \\ f_2 \geq 0 \\ \varphi_1 \geq 0 \\ \tau_2 \geq 0 \end{cases}, \begin{cases} f_1 \geq 0 \\ f_2 \geq 0 \\ \varphi_2 \geq 0 \\ \tau_1 \geq 0 \end{cases}, \begin{cases} f_1 \geq 0 \\ f_2 \geq 0 \\ \varphi_2 \geq 0 \\ \tau_2 \geq 0 \end{cases}, \begin{cases} f_1 \geq 0 \\ f_2 \geq 0 \\ \varphi_3 \geq 0 \\ \tau_1 \geq 0 \end{cases}, \begin{cases} f_1 \geq 0 \\ f_2 \geq 0 \\ \varphi_3 \geq 0 \\ \tau_2 \geq 0 \end{cases}. \tag{15.34}$$

Thus, for each inequality $\Phi^{AB} \geq 0$, we may build a tree. An inequality system of the kind (15.34) corresponds to each terminal node of the tree there.

Now problems (15.30)–(15.32) result in the following problem:

$$F(U^*) = \text{extr}\{F(U_1^*), F(U_2^*), ..., F(U_\eta^*)\}, \tag{15.35}$$

where

$$F(U_k^*) = \underset{U \in W_k \subset R^{\xi+\mu}}{\text{extr}} F(U),\ k = 1, 2, ..., \eta. \tag{15.36}$$

Here η is the number of terminal nodes of the solution tree of problems (15.30)–(15.32), where $\eta = \eta_1 \cdot \eta_2 \cdot ... \cdot \eta_\lambda$, $\lambda = n(n+1)/2$, n is the number of placement objects and λ is the number of *phi*-functions in (15.31) forming solution space W.

Complete enumeration of terminal nodes of the solution tree of problems (15.30)–(15.32) is, in general, just a theoretical goal even for two objects. However, we consider the following major attributes of the problem.

Indeed, the number of the terminal nodes, we deal with, is evaluated by the number η^*, which is considerably less than η. The reason for that lies in the following statements. First, a great number of terminal nodes from the set $\{\upsilon_k, k = 1, 2, ..., \eta\}$ are associated with inconsistent inequality systems $f_k(u) \geq 0$ which results in $W_k = \varnothing$. Secondly, in many cases, $u_{k_1}^*$ and $u_{k_2}^*$, $k_1 \neq k_2$, from (15.36) may be identical. Thirdly, not all local extrema of problems (15.36) are local extrema of problems (15.30)–(15.32); in particular, there are cases, when $W_{k_2} \subset W_{k_1}$, $k_1 \neq k_2$. For solving problems (15.35) and (15.36), we apply the algorithm discussed in Sect. 15.5.

15.4.1 An Application Problem

We shall now consider the following 3D packing problem that is representative of some space applications (e.g. re-entry vehicles). Let's introduce a container Ω, where $\Omega = \mathbf{Q} \cap G$, $\mathbf{Q} = \{u \in \mathbf{R}^3 : z + x^2 + y^2 - a \leq 0\}$, $G = \{u \in \mathbf{R}^3 : -z \leq 0\}$, $u = (x, y, z)$, and a set M of items. The items have the form either of cylinders $\mathbf{C}_i$, $i \in I_1 = \{1, 2, ..., n_1\}$, with metrical characteristics (r_i, h_i) or parallelepipeds $\mathbf{P}_i$, $i \in I_2 = \{n_1 + 1, ..., n_1 + n_2 = n\}$, with metrical characteristics (w_i, l_i, h_i). We denote the base of Ω by S_1. There are two shelves S_k, $k = 2, 3$, parallel to S_1 within container Ω (Fig. 15.11a), such that the distance between S_1 and S_2 is t_1

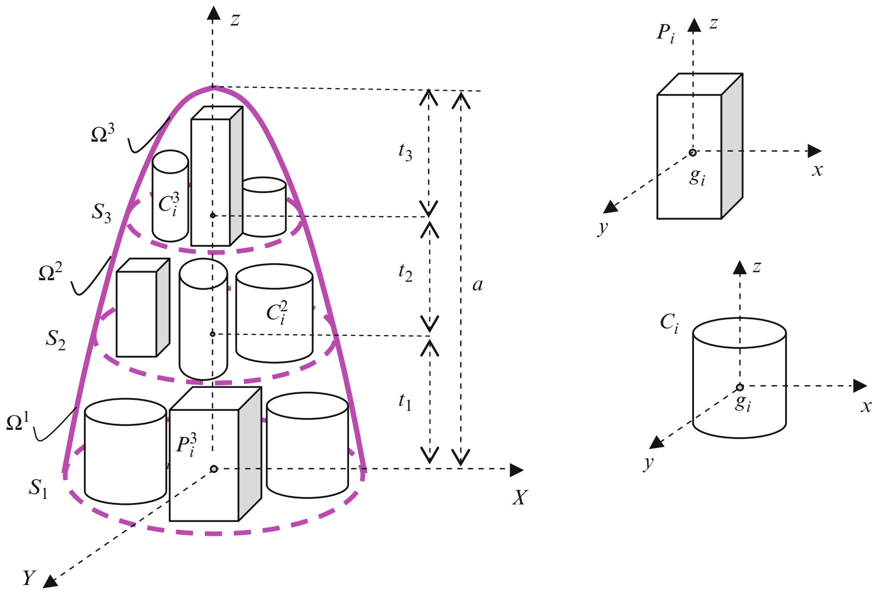


Fig. 15.11 (**a**) Container Ω and subcontainers Ω^k; (**b**) items $\mathbf{C}_i$ and $\mathbf{P}_i$

and between S_2 and S_3 is t_2. Thus, S_1,S_2 and S_3 are circles of radii $r_1 = \sqrt{a}$, $r_2 = \sqrt{a - t_1}$ and $r_3 = \sqrt{a - t_1 - t_2}$. We denote $a - t_1 - t_2$ by t_3 (Fig. 15.11a). Shelves S_2 and S_3 partition container Ω into subcontainers Ω^1, Ω^2 and Ω^3.

We partition set M into three groups: $M^k = \{\mathbf{C}_i, i \in I_1^k, \mathbf{P}_i, i \in I_2^k\}$, $k = 1, 2, 3$, where $I_1^1 = \{1, 2, ..., n_1^1\}$, $I_2^1 = \{n_1 + 1, n_1 + 2, ..., n_2^1\}$, $I_1^2 = \{n_1^1 + 1, n_1^1 + 2, ..., n_1^2\}$, $I_2^2 = \{n_2^1 + 1, n_2^1 + 2, ..., n_2^2\}$, $I_1^3 = \{n_1^2 + 1, n_1^2 + 2, ..., n_1^3\}$, $I_2^3 = \{n_2^2 + 1, n_2^2 + 2, ..., n_2^3\}$, and $n_1^3 = n_1$, $n_2^3 = n$. Assuming that $h^k \leq t^k$ subject to $h^k = \max\{h_i^k, i \in I^k\}$, we place each item of M^k on shelf S_k within subcontainer Ω^k.

The minimal allowable distances σ_{ij} and σ_i are given between each pair of items $M_i^k \in M^k$ and $M_j^k \in M^k$, $i, j \in I^k = \{I_1^k \cup I_2^k\}, i \neq j$, as well as between each item $M_i^k \in M^k$, $i \in I^k$, and the lateral surface of Ω^k, respectively.

Each item from set M has mass m_i, $i \in I_n$, and homogeneous density. We set the centre of gravity g_i of each module at the origin 0_i of coordinate system $0_i xyz$, i.e. $g_i = (0, 0, 0)$ (Fig. 15.11b).

We define the gravity centre $g = (x_g, y_g, z_g)$ of set M as

$$x_g = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i}, \quad y_g = \frac{\sum_{i=1}^{n} m_i y_i}{\sum_{i=1}^{n} m_i},$$

$$z_g = \frac{\sum\limits_{i=1}^{n_1^1} m_i h_i}{\sum\limits_{i=1}^{n} m_i} + \frac{\sum\limits_{i=n_1+1}^{n_2^1} m_i h_i}{\sum\limits_{i=1}^{n} m_i} + \frac{\sum\limits_{i=n_1^1+1}^{n_1^2} m_i(t_1 + h_i)}{\sum\limits_{i=1}^{n} m_i} + \frac{\sum\limits_{i=n_2^1+1}^{n_2^2} m_i(t_1 + h_i)}{\sum\limits_{i=1}^{n} m_i}$$
$$+ \frac{\sum\limits_{i=n_1^2+1}^{n_1^3} m_i(t_1 + t_2 + h_i)}{\sum\limits_{i=1}^{n} m_i} + \frac{\sum\limits_{i=n_2^2+1}^{n_2^3} m_i(t_1 + t_2 + h_i)}{\sum\limits_{i=1}^{n} m_i}.$$

Arrangement of $\mathbf{C}_i$ on shelves S_k within Ω^k is defined by placement parameters $u_i = (v_i, h_i)$ for $i \in I_1^1, u_i = (v_i, t_1 + h_i)$ for $i \in I_1^2$, and $u_i = (v_i, t_1 + t_2 + h_i)$ for $i \in I_1^3$, where $v_i = (x_i, y_i)$. By analogy, we define placement parameters of $\mathbf{P}_i$ as follows: $u_i = (v_i, \theta_i, h_i)$ for $i \in I_2^1$, $u_i = (v_i, \theta_i, t_1 + h_i)$ for $i \in I_2^2$, and $u_i = (v_i, \theta_i, t_1 + t_2 + h_i)$ for $i \in I_2^3$ where $v_i = (x_i, y_i)$, θ_i is a rotation angle of $\mathbf{P}_i$ around the axis $0_i z$.

Thus, vector $u = (u_1, u_2, ..., u_n) \in \mathbf{R}^{3n_1+4n_2}$ defines in $\mathbf{R}^3$ the specific arrangement of the items of set M. Because of the fixed coordinates $z_i,\ i \in I^k,$ for each item $M_i^k \in M^k$, the vector u of variables results in $v = (v_1, v_2, ..., v_n) \in \mathbf{R}^{\xi}$, $\xi = 2n_1 + 3n_2$.

Let the gravity point s of container Ω be given, $s = (x_s, y_s, z_s) \in \Omega$.

Balance Optimisation Define vector $v \in \mathbf{R}^{\xi}$ so that distance d between the centre of gravity g of set M and the given gravity point $s \in \Omega$ will reach its minimal value, considering the given distance constraints.

Mathematical models (15.30)–(15.32) take the form:

$$F(v^*) = \min F(v) \text{ s.t. } v \in W, \tag{15.37}$$

where

$$F(v) = (x_g - x_s)^2 + (y_g - y_s)^2 + (z_g - z_s)^2,$$

$$W = \{v \in \mathbf{R}^{\xi} : \widetilde{\Phi}_{ij}^{\mathbf{CC}} - \sigma_{ij} \geq 0, i < j \in I_1^k, \widetilde{\Phi}_{ij}^{\mathbf{PP}} - \sigma_{ij} \geq 0, i < j \in I_2^k, \tag{15.38}$$

$$\widetilde{\Phi}_{ij}^{\mathbf{CP}} - \sigma_{ij} \geq 0, i \in I_1^k, j \in I_2^k, \widetilde{\Phi}_i^{\mathbf{C}} - \sigma_i \geq 0, i \in I_1^k, \widetilde{\Phi}_i^{\mathbf{P}} - \sigma_i \geq 0, i \in I_2^k, k = 1, 2, 3\},$$

$\widetilde{\Phi}_{ij}^{\mathbf{CC}}$, $\widetilde{\Phi}_{ij}^{\mathbf{PP}}$, $\widetilde{\Phi}_{ij}^{\mathbf{CP}}$ are normalised *phi*-functions for each pair of items $M_i^k \in M^k$ and $M_j^k \in M^k$, $i, j \in I^k, i \neq j$; $\widetilde{\Phi}_i^{\mathbf{C}}$ and $\widetilde{\Phi}_i^{\mathbf{P}}$ are normalised *phi*-functions for each item $M_i^k \in M^k$, $i \in I^k$, and the complement of Ω^k.

Taking into account the features of the problem, normalised *phi*-functions listed in (15.38) may be replaced with normalised *phi*-functions for circles and rectangles in the following manner:

1. For non-overlapping constraints: $\widetilde{\Phi}_{ij}^{\mathbf{CC}} \Rightarrow \widetilde{\Phi}_{ij}^{CC}$, $\widetilde{\Phi}_{ij}^{\mathbf{PP}} \Rightarrow \widetilde{\Phi}_{ij}^{RR}$, $\widetilde{\Phi}_{ij}^{\mathbf{CP}} \Rightarrow \widetilde{\Phi}_{ij}^{CR}$, $i, j \in I^k, i \neq j$, where $\widetilde{\Phi}_{ij}^{CC}$ is the normalised *phi*-function for two circles C_i and

C_j, $\widetilde{\Phi}^{RR}_{ij}$ is the normalised *phi*-function for two rotating rectangles R_i and R_j and $\widetilde{\Phi}^{CR}_{ij}$ is the normalised *phi*-function for C_i and rotating R_j. Metrical characteristics of C_i and R_j are defined by (r_i) and (w_i, l_i), respectively.

2. For containment constraints: $\widetilde{\Phi}^{\mathbf{C}}_i \Rightarrow \widetilde{\Phi}^{S^*_{ki}C}_i$, $i \in I^k_1$, where $\widetilde{\Phi}^{S^*_{ki}C}_i$ is the normalised *phi*-function for C_i and $S^*_{ki} = R^2 \backslash \text{int} S_{ki}$, S_{ki} is a circle of radius $r_{1i} = \sqrt{a - h_i}$ for $k = 1$, $r_{2i} = \sqrt{a - t_1 - h_i}$ for $k = 2$, $r_{3i} = \sqrt{a - t_1 - t_2 - h_i}$ for $k = 3$, $\widetilde{\Phi}^{\mathbf{P}}_i \Rightarrow \widetilde{\Phi}^{S^*_{ki}R}_i$, $i \in I^k_2$, where $\widetilde{\Phi}^{S^*_{ki}R}_i$ is the normalised *phi*-function for rotating R_i and S^*_{ki}.
   Thus, (15.38) takes the form

$$W = \{v \in \mathbf{R}^{\xi} : \widetilde{\Phi}^{CC}_{ij} - \sigma_{ij} \geq 0, i < j \in I^k_1, \widetilde{\Phi}^{RR}_{ij} - \sigma_{ij} \geq 0, i < j \in I^k_2,$$

$$\widetilde{\Phi}^{CR}_{ij} - \sigma_{ij} \geq 0, i \in I^k_1, j \in I^k_2, \widetilde{\Phi}^{S^*_{ki}C}_i - \sigma_i \geq 0, i \in I^k_1,$$

$$\widetilde{\Phi}^{S^*_{ki}R}_i - \sigma_i \geq 0, i \in I^k_2, k = 1, 2, 3\}.$$

In terms of adjusted *phi*-functions formula (15.38) is reduced to

$$W = \{v \in \mathbf{R}^{\xi} : \widetilde{\Phi}^{CC}_{ij} \geq 0, i < j \in I^k_1, \widetilde{\Phi}^{RR}_{ij} \geq 0, i < j \in I^k_2, \widetilde{\Phi}^{CR}_{ij} \geq 0, i \in I^k_1, j \in I^k_2,$$

$$\widetilde{\Phi}^{S_{ki}C}_i \geq 0, i \in I^k_1, \widetilde{\Phi}^{S_{ki}R}_i \geq 0, i \in I^k_2, k = 1, 2, 3\},$$

where notation $\widetilde{\Phi}$ means an adjusted *phi*-function, as defined above.

We shall illustrate the problem with the following instance.

Let $a = 16$, $t_1 = 4$, $t_2 = 2$, $s = (0, 0, 0)$; $M = \{\mathbf{C}_i, i = 1, ..., 7, \mathbf{P}_i, i = 8, ..., 12\}$ r_1=1, r_2=1.1, r_3=1.2, r_4=1, r_5=1.1, r_6=1, r_7=1, $(w_i, l_i) = (1.3333, 0.75)$, $i = 8$, ..., 12, $h_i = 0.88$, $i = 1, ..., 12$; $m_1 = 3.1416$, $m_2 = 3.8013$, $m_3 = 4.5239$, $m_4 = 3.1416$, $m_5 = 3.8013$, $m_6 = 3.1416$, $m_7 = 3.1416$, $m_i = 4$, $i = 8, ..., 12$; $\sigma_i = 0.2$, $i = 1, 2, \ldots, 12$, $\sigma_{ij} = 0.2$, $i \neq j \in \{1, ..., 12\}$. We partition M in the following way: $M^1 = \{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \mathbf{P}_8, \mathbf{P}_9\}$, $M^2 = \{\mathbf{C}_4, \mathbf{C}_5, \mathbf{P}_{10}, \mathbf{P}_{11}\}$, $M^3 = \{\mathbf{C}_6, \mathbf{C}_7, \mathbf{P}_{12}\}$.

The vector of variables is $v = (v_1, v_2, ..., v_{12}) \in \mathbf{R}^{29}$.

In Fig. 15.12 we give the optimal arrangement of 2D objects $\{C_i, i = 1, ..., 7, R_i, i = 8, ..., 12\}$ corresponding to point v^* with respect to the optimal arrangement of 3D objects $\{\mathbf{C}_i, i = 1, ..., 7, \mathbf{P}_i, i = 8, ..., 12\}$, where

$$v^* = (x^*_1, y^*_1, ..., x^*_7, y^*_7, x^*_8, y^*_8, \theta^*_8, ..., x^*_{12}, y^*_{12}, \theta^*_{12}),$$

$$(x^*_1, y^*_1) = (-1.29956, -1.71262), (x^*_2, y^*_2) = (0.00122, 0.45696),$$

$$(x^*_3, y^*_3) = (1.09994, -1.78878), \; (x^*_4, y^*_4) = (1.99438, 0.14595),$$

$$(x^*_5, y^*_5) = (-1.88631, 0.17553), (x^*_6, y^*_6) = (-0.16247, 1.49016),$$

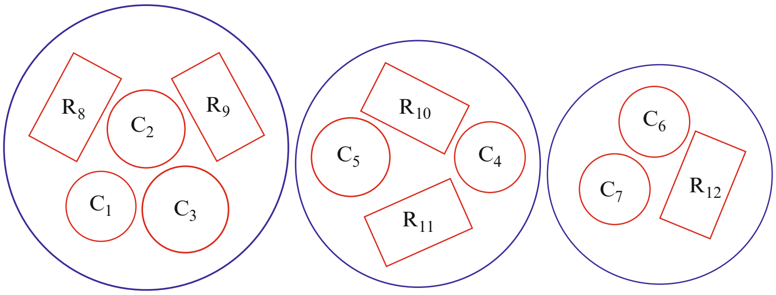


Fig. 15.12 The optimal arrangement of objects corresponding to point v^*

$$(x_7^*, y_7^*) = (-1.24515, -0.42938), (x_8^*, y_8^*, \theta_8^*) = (-1.98972, 1.08949, 5.21332),$$

$$(x_9^*, y_9^*, \theta_9^*) = (1.995774, 1.078259, 1.075465),$$

$$(x_{10}^*, y_{10}^*, \theta_{10}^*) = (-0.124421, 1.553304, 0.498449),$$

$$(x_{11}^*, y_{11}^*, \theta_{11}^*) = (0.023846, -1.616274, 2.700283),$$

$$(x_{12}^*, y_{12}^*, \theta_{12}^*) = (1.201675, -0.285530, 1.973668).$$

In the example the number of all inequality systems is $\eta = 4947802324992$. In order to search for the global minimum just $\eta^* = 723$ inequality systems have been actually processed (running time for searching for a local minimum is less than a second).

15.5 Sketch of Solution Algorithm

Here we discuss an approach to the solution of the basic problem described in the previous section, i.e. finding the global extremum (or, at least, a good approximation to it) of the objective function F. Our algorithm involves methods of constructing starting points and methods of local and global optimisation.

In [24], we give a survey of techniques for the construction of starting points. We have generated starting points in several ways, so far, but the most frequently used is described hereinafter (see [30]). First, we approximate the container Ω and objects $A_1,\ldots,A_n$ by rectangular polygons (polytopes) $P_0,P_1,\ldots,P_n$ with sides parallel to fixed coordinate axes. Then we place figures $P_1,\ldots,P_n$ into P_0 consecutively, according to the object sequence $P_{i_1}, P_{i_2}, ..., P_{i_n}$ generated by a modification of the decremental neighbourhood method (see [31]). This procedure employs a probabilistic search and is designed to find the most promising object sequences. The object sequences will correspond to some starting points $U_1,\ldots,U_k$ in W.

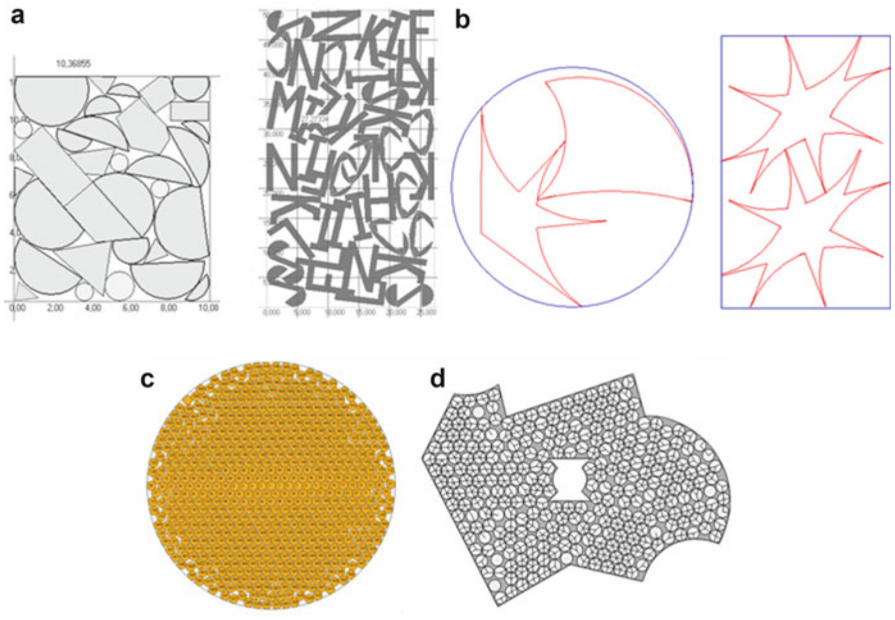


Fig. 15.13 Examples of 2D packing problems

The features of *phi*-functions are essential for smooth performance of local minimisation schemes. Our local search employs a modification of the Zoutendijk algorithm of feasible directions [32] and the concept of active inequalities.

Let $U_1 \in W$ be a starting point. The algorithm forms subset $W_{j,1} \subset W$, such that $U_1 \in W_{j,1}$. Then it finds a local minimum $F(U_1^*) = \min F(U)$, s.t. $U \in W_{j,1}$ and derives the steepest descent vector Z_1 from U_1^* for the problem $\min F(U)$, s.t. $U \in W$. If $Z_1 \neq 0$ it defines point $U_2 = (U_1^* + tZ_1) \in W$, where $t \geq 0$. It forms subset $W_{j,2} \subset W$, such that $U_2 \in W_{j,2}$. Then it finds a local minimum $F(U_2^*) = \min F(U)$, s.t. $U \in W_{j,2}$ and derives the steepest descent vector Z_2 from U_2^* for the problem min $F(U)$, s.t. $U \in W$. If $Z_2 \neq 0$ it defines point $U_3 = (U_2^* + tZ_2) \in W$. It forms subset $W_{j,3} \subset W$, such that $U_3 \in W_{j,3}$. Then it finds a local minimum $F(U_3^*) = \min F(U)$, s.t. $U \in W_{j,3}$ and so on. We repeat the procedure N times until it gets $t \leq \varepsilon$, where $\varepsilon > 0$ is the solution accuracy. Thus, U_N^* is a point of local minimum of the problem $\min F(U)$, s.t. $U \in W$.

The algorithm produces points of local minima $U_1^*, .., U_k^*$ of F. Eventually, we choose the point of local minimum which gives the smallest value of F. This completes our algorithm.

Below we give a few computational results published in our papers (see, [24, 25, 30, 31, 33–35]): (1) packing primary and composed objects with rotations into a rectangle of minimal length (Fig. 15.13a); (2) the optimal packing of arbitrarily shaped objects with rotations into a circular or rectangular container of minimal

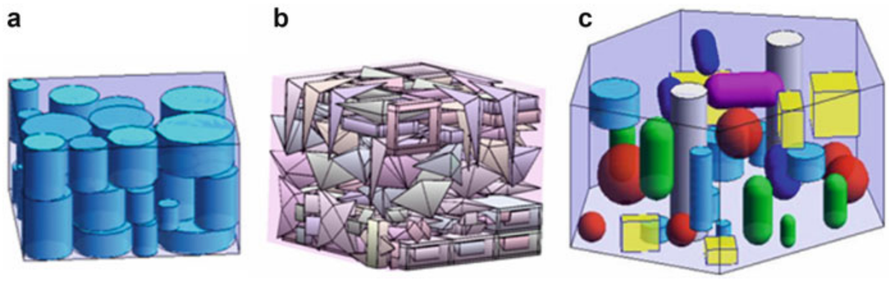


Fig. 15.14 Examples of 3D packing problems

area (Fig. 15.13b); (3) packing equal circles into a larger circle in order to maximise the number of circles (Fig. 15.13c); (4) packing equal circles into a region with prohibited areas, in order to maximise the number of circles (Fig. 15.13d); (5) packing cylinders into a box of minimal height (Fig. 15.14a); (6) packing non-convex polytopes into a box of minimal height (Fig. 15.14b); and (7) packing 3D objects into a prism of minimal height, considering the given prohibited zones and minimal allowable distances between objects, as well as between each object and the border of the container (Fig. 15.14c). For visualisation of the local optimisation procedure, see http://www.math.uab.edu/~chernov/CP.

We have also progressed in solving covering problems which one can find in our recent papers (e.g. [36, 37]). The results may be applied for air and space observation systems.

15.6 Conclusions

In this chapter we demonstrate how the use of phi-functions can improve the performance of 2D and 3D cutting and packing algorithms. Our phi-functions have the following features: they can be applied for non-overlapping and containment constraints to 2D and 3D objects (these include non-convex objects, some curved shapes, regions with holes and cavities, etc.); phi-functions may take into account continuous translations and rotations of objects, variable metric characteristics of objects, prohibited areas, possible restrictions on the allowable distances between objects and from the objects to the walls of the container; phi-functions are useful when dealing with overlapping objects, as they measure the degree of overlap (this is useful for covering problems). In addition, the phi-functions are defined by simple formulas, which allow us to use optimisation algorithms of mathematical programming. Phi-functions allow us to enlarge the class of optimisation placement problems that can be effectively solved and apply parallel computing. We are constantly working on the improvement of our algorithms.

References

1. Burke, E., Hellier, R., Kendall, G., Whitwell, G.: Irregular packing using the line and arc no-fit polygon. Oper. Res. **58**(4), 948–970 (2010)
2. Costa, M.T., Gomes, A.M., Oliveira, J.F.: Heuristic approaches to large-scale periodic packing of irregular shapes on a rectangular sheet. Eur. J. Oper. Res. **192**, 29–40 (2009)
3. Cui, Y.: Generating optimal multi-segment cutting patterns for circular blanks in the manufactoring of electric motors. Eur. J. Oper. Res. **169**, 30–40 (2006)
4. Gomes, A.M., Oliveira, J.F.: Solving irregular strip packing problems by hybridising simulated annealing and linear programming. Eur. J. Oper. Res. **171**, 811–829 (2006)
5. Birgin, E.G., Martinez, J.M., Nishihara, F.H., Ronconi, D.P.: Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization. Comput. Oper. Res. **33**, 3535–3548 (2006)
6. Kallrath, J.: Cutting circles and polygons from area-minimizing rectangles. J. Glob. Optim. **43**, 299–328 (2009)
7. Milenkovic, V.J., Daniels, K.: Translational polygon containment and minimal enclosure using mathematical programming. Int. Trans. Oper. Res. **6**, 525–554 (1999)
8. Bennell, J.A., Oliveira, J.F.: The geometry of nesting problems: A tutorial. European J. Oper. Res. **184**, 397–415 (2008)
9. Wascher, G., Hauner, H., Schumann, H.: An improved typology of cutting and packing problems. Eur. J.Oper. Res. **183**(3, 16), 1109–1130 (2007)
10. Bennell, J.A., Song, X.: A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums. Comput. Oper. Res. **35**(1), 267–281 (2006)
11. Milenkovic, V.J.: Rotational polygon overlap minimization and compaction. Comput. Geom. **10**, 305–318 (1998)
12. Milenkovic, V.J., Sacks, E.: Two approximate Minkowski sum algorithms. Int. J. Comput. Geom. Appl. **20**(4), 485–509 (2010)
13. Aladahalli, C., Cagan, J., Shimada, K.: Objective function effect based pattern search — theoretical framework inspired by 3D component layout. J. Mech. Design **129**, 243–254 (2007)
14. Birgin, E.G., Martínez, J., Ronconi, D.: Optimizing the packing of cylinders into a rectangular container: A nonlinear approach. European J. Oper. Res. **160**(1), 19–33 (2005)
15. Birgin, E.G., Sobral, F.N.C.: Minimizing the object dimensions in circle and sphere packing problems. Comput. Oper. Res. **35**, 2357–2375 (2008)
16. Egeblad, J., Nielsen, B.K., Brazil, M.: Translational packing of arbitrary polytopes. Comput. Geom. **42**(4), 269–288 (2009)
17. Egeblad, J., Nielsen, B.K., Odgaard, A.: Fast neighborhood search for two- and three-dimensional nesting problems. Eur. J. Oper. Res. **183**(3), 1249–1266 (2007)
18. Fasano, G.: MIP-based heuristic for non-standard 3D-packing problems. 4OR: Quar. J. Belgian French Italian Oper. Res. Soc. **6**(3), 291–310 (2008)
19. Gan, M., Gopinathan, N., Jia, X., Williams, R.A.: Predicting packing characteristics of particles of arbitrary shapes. KONA **22**, 82–93 (2004)
20. Jia, X., Gan, M., Williams, R.A., Rhodes, D.: Validation of a digital packing algorithm in predicting powder packing densities. Powder Tech. **174**, 10–13 (2007)
21. Torquato, S., Stillinger, F.H.: Jammed hard-particle packings: From Kepler to Bernal and beyond. Rev. Modern Phys. **82**(3), 2633–2672 (2010)
22. Cagan, J., Shimada, K., Yin, S.: A survey of computational approaches to three-dimensional layout problems. Comput. Aided Design **34**, 597–611 (2002)
23. Bennell, J.A., Scheithauer, G., Stoyan, Yu, Romanova, T.: Tools of mathematical modelling of arbitrary object packing problems. J. Annal. Oper. Res. **179**(1), 343–368 (2010)
24. Chernov, N., Stoyan, Y., Romanova, T.: Mathematical model and efficient algorithms for object packing problem. Comput. Geom.: Theor. Appl. **43**(5), 535–553 (2010)
25. Chernov, N., Stoyan, Y., Romanova, T., Pankratov, A.: *Phi*-functions for 2D objects formed by line segments and circular arcs. Adv. Oper. Res. (2012). doi:10.1155/2012/346358

26. Scheithauer, G., Stoyan, Yu, Romanova, T.: Mathematical modeling of interactions of primary geometric 3D objects. Cybernet. Syst. Anal. **41**, 332–342 (2005)
27. Stoyan, Y., Gil, M., Terno, J., Romanova, T., Scheithauer, G.: Construction of a *Phi*- function for two convex polytopes. Applicationes Mathematicae **2**(29), 199–218 (2002)
28. Stoyan, Y., Gil, N., Romanova, T., Scheithauer, G.: *Phi*-functions for complex 2D objects. 4OR: Quar. J. Belgian French Italian Oper. Res. Soc. **2**(1), 69–84 (2004)
29. Stoyan, Y., Terno, J., Scheithauer, G., Gil, N., Romanova, T.: *Phi*-function for 2D primary objects. Studia Informatica **2**(1), 1–32 (2002)
30. Stoyan, Y., Pankratov, A., Sheithauer, G.: Translational packing non-convex polytopes into a parallelepiped. J. Mech. Eng. **12**(3), 67–76 (2009)
31. Stoyan, Y., Chugay, A.: Packing cylinders and rectangular parallelepipeds with distances between them. Eur. J. Oper. Res. **197**, 446–455 (2008)
32. Zoutendijk, G.: Nonlinear Programming, Computational Methods, Integer and Nonlinear Programming. North-Holland, Amsterdam (1970)
33. Stoyan, Y., Chugay, A.: An optimization problem of packing identical circles into a multiply connected region, Part 2. A solution method and its realisation. J. Mech. Eng. **14**(2), 52–61 (2011)
34. Stoyan, Y., Gil, M., Scheithauer, G., Pankratov, A., Magdalina, I.: Packing of convex polytopes into a parallelepiped. Optimization **54**(2), 215–235 (2005)
35. Stoyan, Y., Yaskov, G.: Packing congruent hyperspheres into a hypersphere. J. Global Optim. (2012). doi:10.1007/s10898-011-9716-z
36. Scheithauer, G., Stoyan, Yu, Romanova, T., Krivulya, A.: Covering a polygonal region by rectangles. Comput. Optim. Appl. **48**(3), 675–695 (2011)
37. Stoyan, Y., Patsuk, V.: Covering a compact polygonal set by identical circles. Comput. Optim. Appl. **46**, 75–92 (2010)

Chapter 16
Optimization of Low-Energy Transfers

Francesco Topputo and Edward Belbruno

Abstract In this chapter the optimization of low-energy transfers is treated. The concepts of trajectory optimization are recalled, and the direct transcription strategy is briefly sketched. The restricted three- and four-body problems are described, and their properties are discussed. Two different types of low-energy transfers are optimized: impulsive Earth–Moon exterior low-energy transfers and low-energy, low-thrust transfers to the L_1 periodic orbits of the Earth–Moon system.

Keywords Low-energy transfers • Trajectory optimization • Ballistic capture • Low-thrust propulsion

16.1 Introduction

The patched-conic method represents the classical technique adopted to design a lunar or an interplanetary transfer. In the case of Earth–Moon, the Hohmann transfer takes typically a few days and requires a cost that depends on the altitudes of the initial and final orbits about the Earth and the Moon, respectively. Since the Hohmann transfer is obtained by patching together two different conic arcs (i.e., an ellipse and a hyperbola in the Earth–Moon case) a hyperbolic excess velocity upon Moon arrival arises. The magnitude of this velocity determines the size of the maneuver required to put the spacecraft into a stable, final Moon orbit.

F. Topputo (✉)
Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Via La Masa 34, 20156 Milano, Italy
e-mail: francesco.topputo@polimi.it

E. Belbruno
Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012, USA
e-mail: belbruno@cims.nyu.edu

The propellant spent to accomplish such transfers is a monotonic function of the sum of all velocity changes or Δv.

Low-energy transfers have been found in the attempt of reducing the Δv of Earth–Moon transfers. The idea behind a low-energy transfer is to extend the model in which the orbits are designed. When two or more gravitational attractions simultaneously act on the spacecraft, the more complex dynamics can be exploited to improve the performances of the transfer trajectories. Thus, in a low-energy transfer, the classical Keplerian decomposition is avoided, and the natural dynamics is exploited in a more efficient way.

A method to obtain Earth-to-Moon transfers with no hyperbolic excess velocity at Moon arrival was found about two decades ago by exploiting the intrinsic nature of the Sun–Earth–Moon dynamics [1, 3, 4]. The idea of the exterior low-energy transfers to the Moon consists in departing from a given point near the Earth and eventually flying by the Moon to gain enough energy to go at a distance of approximately four Earth–Moon distance units (1.5×10^6 km). In this region, due to the high sensitivity to initial conditions, a negligible Δv is used to put the spacecraft into a lunar capture trajectory that leads to an unstable ellipse around the Moon. Once there, another maneuver is performed to put the spacecraft into a stable lunar circular orbit. It has been demonstrated that these new transfers are more efficient than the Hohmann transfer although the time of flight increases [4]. The mechanism that governs the low-energy transfers has been related to the invariant manifolds associated to the periodic orbits of the restricted three-body problem [2]. In particular, it has been shown how a low-energy trajectory can be constructed by patching two manifold-related orbits, each one defined in a different restricted three-body problem [18]. A low-energy transfer was used in the rescue of the Japanese spacecraft Hiten [3], a ballistic capture at arrival has been considered for the interplanetary missions BepiColombo [15], and the recent NASA GRAIL mission used a low-energy transfer analogous to the one Hiten used to send two spacecraft into lunar orbit [25].

In analogy with low-energy transfers, the transfers to the Lagrangian periodic orbits are designed by exploiting the dynamics of the restricted three-body problem. Indeed, introducing the concept of stable and unstable manifolds associated to a periodic orbit, a systematic approach to design trajectories for libration point missions has been developed: in order to reach the final orbit at a zero cost, the spacecraft has to be placed on its stable manifold [10, 14]. To this aim, it is important to check whether or not the stable manifold associated to a certain orbit approaches the Earth. In the Sun–Earth system this happens for a wide class of orbits, but this is not the case of the Earth–Moon system. This means that a single-impulse transfer from a low Earth orbit to an Earth–Moon libration point orbit is not allowed. Low-thrust propulsion has been introduced to fill this gap [20].

Although low-energy transfers allow us to define solutions requiring less propellant than patched-conic transfers, they require an optimization step for the purpose of further cost reduction and mission constraints satisfaction [21, 22, 23, 24]. The optimization of low-energy transfers is nontrivial as the high nonlinearities characterizing these orbits lead most optimization algorithms to fail. In this contribution we show how low-energy transfers are optimized to find efficient solutions requiring moderate transfer times.

16.2 Trajectory Optimization

The trajectory optimization is an optimal control problem. In this section we define the optimal control problem and show how it can be solved with a direct approach. A simple problem with fixed terminal time and no path constraints is considered for brevity sake. The reader can refer to [5, 6, 7, 9, 30] for a more general treatment.

16.2.1 Optimal Control

The optimal control problem requires that, given a set of first-order differential equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \ , \tag{16.1}$$

the control functions $\mathbf{u}(t)$ must be determined within initial, final time t_i, t_f, such that the performance index

$$J = \varphi(\mathbf{x}(t_f), t_f) + \int_{t_i}^{t_f} L(\mathbf{x}, \mathbf{u}, t)\, dt \tag{16.2}$$

is minimized while satisfying the two-point conditions

$$\mathbf{x}(t_i) = \mathbf{x}_i, \qquad \Psi(\mathbf{x}(t_f), \mathbf{u}(t_f), t_f) = 0 \ . \tag{16.3}$$

The problem consists in finding a solution that represents a stationary point of the augmented performance index

$$\begin{aligned} \bar{J} = {} & \varphi(\mathbf{x}(t_f), t_f) + \nu^T \Psi(\mathbf{x}(t_f), \mathbf{u}(t_f), t_f) \\ & + \int_{t_i}^{t_f} \left[L(\mathbf{x}, \mathbf{u}, t) + \lambda^T (\mathbf{f}(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}}) \right] dt \ , \end{aligned} \tag{16.4}$$

where λ is the vector of costates and ν is the multiplier of the boundary condition. The necessary conditions for optimality, also referred to as Euler–Lagrange equations, are

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \lambda}, \qquad \dot{\lambda} = -\frac{\partial H}{\partial \mathbf{x}}, \qquad \frac{\partial H}{\partial \mathbf{u}} = 0 , \tag{16.5}$$

where H, the Hamiltonian, is

$$H(\mathbf{x}, \lambda, \mathbf{u}, t) = L(\mathbf{x}, \mathbf{u}, t) + \lambda^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) . \tag{16.6}$$

The differential–algebraic system (16.5) must be solved together with the boundary conditions (16.3) and the final condition

$$\lambda(t_f) = \left[\frac{\partial \varphi}{\partial \mathbf{x}} + \left(\frac{\partial \Psi}{\partial \mathbf{x}} \right)^T v \right]_{t=t_f} . \tag{16.7}$$

16.2.2 Direct Transcription

Indirect approaches consider (16.5) to solve the trajectory design problem. Another philosophy, known as direct approach, consists in translating the continuous optimal control into a nonlinear programming (NLP) problem and solving it for a finite set of variables.

A uniform time grid $t_i = t_1 < t_2 < \ldots < t_N = t_f$ is constructed. The time labels t_j, $j = 1, \ldots, N$ are referred to as mesh points, and $h = (t_N - t_1)/(N-1)$ is the step size of the discretization. The states and the controls are discretized over the time grid by defining $\mathbf{x}_j = \mathbf{x}(t_j)$ and $\mathbf{u}_j = \mathbf{u}(t_j)$. The vector of variables of the NLP problem is

$$\mathbf{y} = \{\mathbf{x}_1, \mathbf{u}_1, \ldots, \mathbf{x}_N, \mathbf{u}_N, t_1, t_N\}^T , \tag{16.8}$$

where the inclusion of initial, final time allows us to achieve optimal departure and transfer duration.

The dynamics (16.1) are replaced by a finite set of defects, derived by numerical integration. For instance, if a Runge–Kutta scheme is used, the defects takes the following form:

$$\zeta_i \equiv \mathbf{x}_{i+1} - \mathbf{x}_i - h_i \sum\nolimits_{j=1}^{k} \beta_j \mathbf{f}_{ij} , \quad i = 1, \ldots, N , \tag{16.9}$$

with appropriate definitions of $\mathbf{f}_{ij}$ [6]. The equality constraints are

$$\mathbf{c}(\mathbf{y}) = \{\zeta_1, \zeta_2, \ldots, \zeta_{N-1}, \Psi\}^T , \tag{16.10}$$

and, with a similar approach, the objective function (16.2) can be written in terms of $\mathbf{y}$, namely $F = F(\mathbf{y})$.

Definition 1 With direct approach, the NLP problem for trajectory optimization is

$$\min_{\mathbf{y}} F(\mathbf{y}) \quad \text{subject to} \quad \mathbf{c}(\mathbf{y}) = 0 \, . \tag{16.11}$$

16.2.3 Nonlinear Programming

The trajectory optimization problem may be solved with numerical methods incorporating some type of iterations. Let the scalar, Lagrangian function of Eq. (16.11) be

$$L(\mathbf{y}, \lambda) = F(\mathbf{y}) - \lambda^T \mathbf{c}(\mathbf{y}) \, , \tag{16.12}$$

where another set of Lagrange multipliers λ is introduced. The necessary conditions for a constrained optimum are

$$\begin{aligned} \mathbf{g}(\mathbf{y}) - \mathbf{G}^T(\mathbf{y})\lambda = 0 \, , \\ \mathbf{c}(\mathbf{y}) = 0 \, , \end{aligned} \tag{16.13}$$

where $\mathbf{g}$ and $\mathbf{G}$ are the gradient of $F(\mathbf{y})$ and the Jacobian of $\mathbf{c}(\mathbf{y})$, respectively. The nonlinear algebraic system (16.13) can be solved via a Newton's method. Thus, given a generic initial guess ($\mathbf{y}$, λ), its corrections ($\Delta\mathbf{y}$, Δλ), necessary to construct the new solution $(\mathbf{y} + \Delta\mathbf{y}, \lambda + \Delta\lambda)$, are found by solving

$$\begin{bmatrix} \mathbf{H}_L & -\mathbf{G}^T \\ \mathbf{G} & 0 \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{x} \\ \Delta\lambda \end{Bmatrix} = \begin{Bmatrix} -\mathbf{g} \\ -\mathbf{c} \end{Bmatrix} , \tag{16.14}$$

where $\mathbf{H}_L$ is the Hessian of Eq. (16.12). Equations (16.14) are known as the Karush–Kuhn–Tucker system. These are solved iteratively until a convergence criterion is satisfied.

16.3 Equations of Motion

In this section the equations of motion of the planar circular restricted three-body problem (PCRTBP) are given, and the linear structure of the phase space about two equilibrium points of this system is briefly discussed. We show how this dynamics is modified to accommodate the low-thrust propulsion. The perturbation of the Sun is taken into account in the planar bicircular restricted four-body problem (PBRFBP).

16.3.1 Planar Circular Restricted Three-Body Model

In the PCRTBP two primary bodies P_1, P_2 of masses $m_1 > m_2 > 0$, respectively, move under mutual gravity on circular orbits about their common center of mass. The third body, P, assumed of infinitesimal (zero) mass, moves under the gravity of the primaries and in their same plane. The motion of the primaries is not affected by P. In the present case, P represents a spacecraft, and P_1, P_2 represent the Earth and the Moon, respectively. Let the mass ratio of P_2 over the total mass be $\mu = m_2/(m_1 + m_2)$; $\mu = 1.21506683 \times 10^{-2}$ is considered in the following.

The motion of P relative to a corotating coordinate system (x, y) with the origin at the center of mass of the two bodies, and in normalized units of distance, mass, and time, is described by [27]

$$\ddot{x} - 2\dot{y} = \frac{\partial \Omega}{\partial x}\,, \qquad \ddot{y} + 2\dot{x} = \frac{\partial \Omega}{\partial y}\,, \tag{16.15}$$

where the effective potential, Ω, is given by

$$\Omega(x,y) = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{1}{2}\mu(1-\mu)\,, \tag{16.16}$$

with $r_1 = \left[(x+\mu)^2 + y^2\right]^{1/2}$, $r_2 = \left[(x+\mu-1)^2 + y^2\right]^{1/2}$ as P_1, P_2 are located at $(-\mu, 0)$, $(1-\mu, 0)$, respectively.

The system (16.15) admits an integral of motion, the Jacobi integral,

$$J(x,y,\dot{x},\dot{y}) = 2\Omega(x,y) - (\dot{x}^2 + \dot{y}^2)\,, \tag{16.17}$$

that defines the energy manifold

$$\mathscr{J}(C) = \{(x,y,\dot{x},\dot{y}) \in \mathbb{R}^4 | J(x,y,\dot{x},\dot{y}) = C\}, \tag{16.18}$$

whose projection onto the configuration space (x, y) is called Hill's or zero-velocity curves. In the PCRTBP, the motion of P is always confined to the Hill's curves, which vary with the Jacobi energy C (see Fig. 16.1).

To model the controlled motion of P under both the gravitational attractions of P_1, P_2, and the low-thrust propulsion, the following differential equations are considered:

$$\ddot{x} - 2\dot{y} = \frac{\partial \Omega}{\partial x} + \frac{T_x}{m}\,, \qquad \ddot{y} + 2\dot{x} = \frac{\partial \Omega}{\partial y} + \frac{T_y}{m}\,, \qquad \dot{m} = -\frac{T}{I_{sp}\, g_0}\,, \tag{16.19}$$

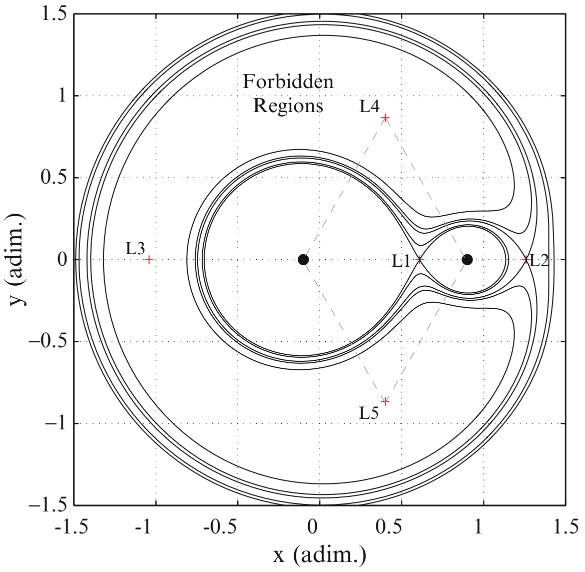


Fig. 16.1 Lagrangian points and Hill's curves for various energy levels ($\mu = 0.\ 1$)

where $T = (T_x^2 + T_y^2)^{1/2}$ is the thrust magnitude, I_{sp} the specific impulse (it has time dimension), and g_0 the gravitational acceleration at sea level. The ballistic motion (16.15) is a fourth-order system, whereas the controlled motion (16.19) is described by a fifth-order system of differential equations. Continuous variations of the spacecraft mass, m, are taken into account when the low-thrust propulsion is considered. This causes a singularity arising when $m \to 0$, besides the well-known singularities given by impacts of P with P_1 or P_2.

16.3.2 The Phase Space Around L_1 and L_2

The motion described by Eq. (16.15) has five equilibrium points, labeled L_k, $k = 1, \ldots, 5$, known as the Euler–Lagrange libration points. Three of these, L_1, L_2, L_3, lie along the x-axis; the other two points, L_4, L_5, lie at the vertices of two equilateral triangles with common base extending from P_1 to P_2 (see Fig. 16.1). Only the first two collinear points are considered as their dynamics adheres to low-energy transfer applications [2, 11, 12, 17, 18, 19, 28, 29].

The linearized solution of Eq. (16.15), written in a frame centered at either L_1 or L_2, is

$$\begin{aligned} x(t) &= A_u e^{\lambda t} + A_s e^{-\lambda t} + A_x \cos(\omega t + \varphi)\ , \\ y(t) &= -k_1 A_u e^{\lambda t} + k_1 A_s e^{-\lambda t} - k_2 A_x \sin(\omega t + \varphi)\ , \end{aligned} \tag{16.20}$$

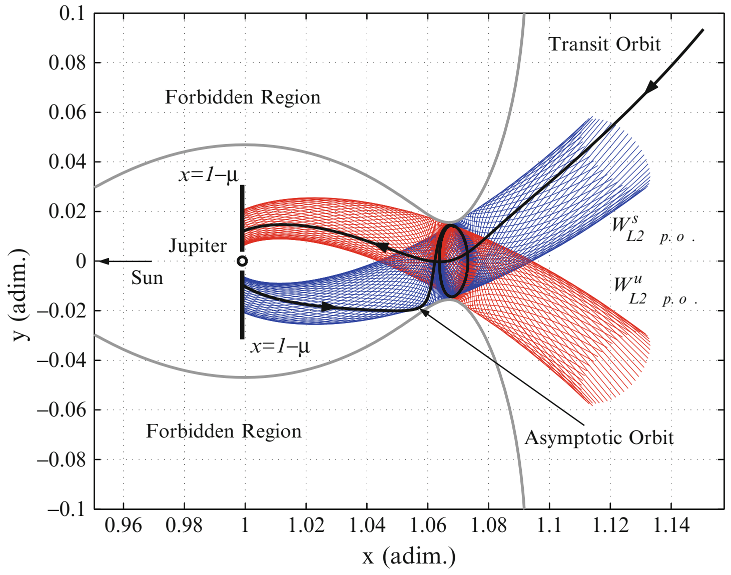


Fig. 16.2 Invariant manifolds associated to a planar Lyapunov orbit about L_2 in the Sun–Jupiter system. Example of transit and asymptotic orbits

where A_s, A_u, and A_x are amplitudes and φ is the phase angle; the two (positive) eigenvalues, λ and ω, as well as the two constants, k_1 and k_2, depend on μ [8, 16]. Equations (16.20) show that, locally, the dynamics of the PCRTBP is equivalent to the product of a saddle $\times$center; i.e., there are one positive and one negative eigenvalues (saddle) and a pair of complex conjugate eigenvalues (center). The amplitudes of the solution associated with the saddle, A_s and A_u, regulate the stable and unstable motion, respectively, and A_x is associated to the size of the in-plane Lyapunov periodic orbits. The reader can consult [13, 16] for more details.

The dynamics of motion defined by the saddle $\times$ center yield invariant manifolds associated to the periodic orbits. These are two-dimensional invariant subsets, $W^s_{\gamma_i}$ and $W^u_{\gamma_i}$, called stable and unstable manifolds, respectively (γ_i is the periodic orbit around L_i, $i = 1, 2$). The manifolds have the following property: every point on the stable manifold spirals toward the periodic orbit in forward time; a similar argument applies to the unstable manifold in backward time. The invariant manifolds associated to the Lyapunov orbits act as separatrices, and they split different regimes of motion (see Fig. 16.2).

16.3.3 Planar Bicircular Restricted Four-Body Model

The PBRFBP describes the dynamics of the Earth–Moon PCRTBP when the Sun perturbation is considered [26, 31]. The equations of motion are

$$\ddot{x} - 2\dot{y} = \frac{\partial \Omega_4}{\partial x}, \qquad \ddot{y} + 2\dot{x} = \frac{\partial \Omega_4}{\partial y}, \tag{16.21}$$

where the four-body potential is

$$\Omega_4(x, y, \theta_S) = \Omega(x, y) + \frac{m_S}{r_3} - \frac{m_S}{\rho^2}(x \cos \theta_S + y \sin \theta_S). \tag{16.22}$$

and $\Omega(x, y)$ is the three-body potential expressed in Eq. (16.16). The phase angle of the Sun, θ_S in Eq. (16.22), is given by

$$\theta_S(t) = \theta_{S,0} + \omega_S t, \tag{16.23}$$

where $\theta_{S,0} = \theta_S(t_0)$. The position of the Sun is $\{\rho\cos\theta_S, \rho\sin\theta_S\}^T$ and therefore the Sun–spacecraft distance is

$$r_3 = \left[(x - \rho \cos \theta_S)^2 + (y - \rho \sin \theta_S)^2\right]^{1/2}. \tag{16.24}$$

The dimensionless physical parameters of the Sun have to be consistent with the normalized Earth–Moon corotating frame. Thus, the distance between the Sun and the Earth–Moon barycenter is $\rho = 3.88811145 \times 10^2$, the mass of the Sun is $m_S = 3.28900541 \times 10^5$, and its angular velocity with respect to the Earth–Moon synodic system is $\omega_S = -9.25195985 \times 10^{-1}$.

The bicircular problem is not a well-defined model because the primaries do not satisfy Newton's equations. However, this model turns out to yield a good approximation of the real four-body dynamics because the eccentricities of both the Earth's and Moon's orbits are equal to 0. 0167 and 0. 0549, respectively, and the Moon's orbit is inclined to the ecliptic by only 5°.

16.4 Optimal Low-Energy Transfers

Optimal low-energy transfers are presented in this section. Two-impulse optimal Earth–Moon ballistic transfers are shown in Sect. 16.4.1, whereas low-thrust transfers to Lagrangian point orbits in the Earth–Moon system are treated in Sect. 16.4.2.

16.4.1 Ballistic Earth–Moon Transfers

Ballistic Earth–Moon transfers are briefly defined. The spacecraft is assumed to be in a circular orbit of radius r_i about the Earth. At the initial time, t_i, an impulse of magnitude Δv_i places the spacecraft onto a transfer trajectory; Δv_i is aligned with the local circular velocity whose magnitude is $\sqrt{(1-\mu)/r_i}$. At the final time, t_f, an impulse of magnitude Δv_f inserts the spacecraft into the final orbit around the Moon. This orbit has radius r_f, and Δv_f is aligned with the local circular velocity whose magnitude is $\sqrt{\mu/r_f}$. The total cost of the transfer is $\Delta v = \Delta v_i + \Delta v_f$, and the transfer time is $\Delta t = t_f - t_i$.

Let $\mathbf{x}_i = (x_i, y_i, \dot{x}_i, \dot{y}_i)$, $\mathbf{x}_f = (x_f, y_f, \dot{x}_f, \dot{y}_f)$ be the initial, final transfer state, and let the problem boundary conditions be

$$(x_i + \mu)^2 + y_i^2 - r_i^2 = 0 \, , \tag{16.25}$$

$$(x_f + \mu - 1)^2 + y_f^2 - r_f^2 = 0 \, , \tag{16.26}$$

$$(x_i + \mu)(\dot{x}_i - y_i) + y_i(\dot{y}_i + x_i + \mu) = 0 \, , \tag{16.27}$$

$$(x_f + \mu - 1)(\dot{x}_f - y_f) + y_f(\dot{y}_f + x_f + \mu - 1) = 0 \, , \tag{16.28}$$

where Eqs. (16.25) and (16.26) constrain $\mathbf{x}_i$ and $\mathbf{x}_f$ to be at a physical distance r_i and r_f from the Earth and the Moon, respectively, whereas Eqs. (16.27) and (16.28) constrain the initial and final velocity vectors to be aligned with the local circular velocity. Under conditions (16.25)–(16.28), the costs for the initial and final maneuvers respectively are

$$\Delta v_i = \sqrt{(\dot{x}_i - y_i)^2 + (\dot{y}_i + x_i + \mu)^2} - \sqrt{\frac{1-\mu}{r_i}} \, , \tag{16.29}$$

$$\Delta v_f = \sqrt{(\dot{x}_f - y_f)^2 + (\dot{y}_f + x_f + \mu - 1)^2} - \sqrt{\frac{\mu}{r_f}} \, . \tag{16.30}$$

The final transfer state is a function of the initial state, initial and final time through $\mathbf{x}_f = \phi(\mathbf{x}_i, t_i; t_f)$, where $\mathbf{x}(t) = \phi(\mathbf{x}_i, t_i; t)$ is a solution of Eqs. (16.21). The optimization problem of two-impulse Earth–Moon transfers is defined as follows.

Definition 2 Find $\mathbf{y} = \{\mathbf{x}_i, t_i, t_f\}$, $t_f > t_i$, that satisfies

$$\min_{\mathbf{y}} \Delta v(\mathbf{y}) \ \text{ subject to } \ \Psi(\mathbf{y}) = 0 \, , \tag{16.31}$$

where $\Delta v = \Delta v_i + \Delta v_f$ given by Eqs. (16.29)–(16.30) and Ψ is the vector of boundary conditions (16.25)–(16.28).

This problem has been solved with a direct transcription and multiple shooting approach (see Sect. 16.2.2), starting from initial guess solutions achieved with a direct shooting (note that there is no continuous control in this case). The altitudes of the departure and arrival circular orbits are 167 km (around the Earth) and 100 km (around the Moon), respectively. In Fig. 16.3 the convergence history of a sample solution is reported.

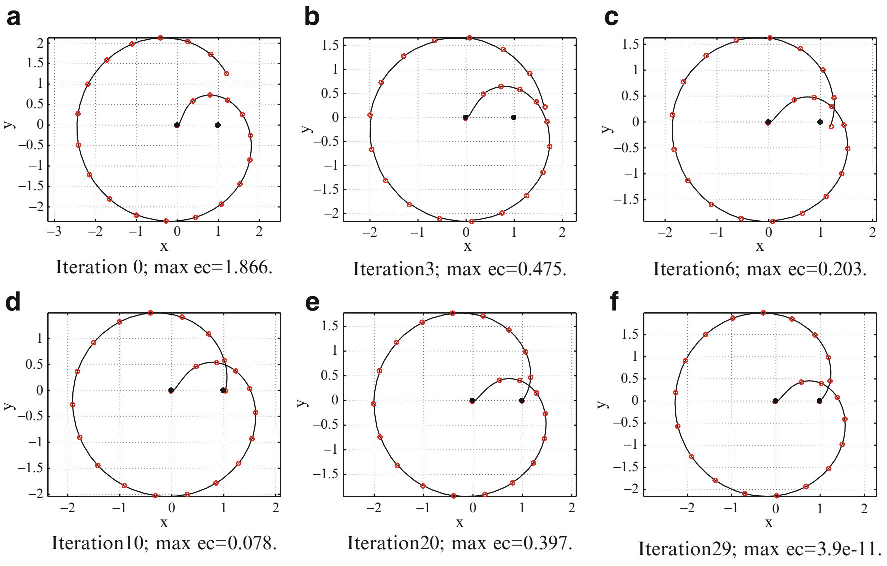


Fig. 16.3 Iterative solutions of problem (16.31) shown in the Earth–Moon rotating frame. Each figure reports the iteration and the maximum equality constraint violation (max ec). The first guess (Iteration 0) is in Fig. 16.3a. This solution has $N = 23$ mesh points (bullets in the figures), i.e., 94 variables and 92 equality constraints. The convergence has been achieved in 29 iterations

Table 16.1 Cost and transfer time for a sample set of low-energy transfers

Sol	Δv_i (m/s)	Δv_f (m/s)	Δv (m/s)	Δt (days)
(1)	3,133.8	635.7	3,769.5	82.2
(2)	3,135.8	664.2	3,800.1	80.5
(3)	3,133.7	638.8	3,772.6	87.5
(4)	3,197.0	655.8	3,852.8	83.3
(5)	3,133.8	668.5	3,802.3	75.5
(6)	3,197.0	637.3	3,834.3	92.8

A sample set of low-energy Earth–Moon transfers is reported in Table 16.1, where the magnitude of the departure and arrival maneuvers, as well as the total cost and the transfer time, are reported. These six solutions in Table 16.1 are shown in Fig. 16.4 in the Earth-centered frame. It can be noticed that, when optimized, the total cost of low-energy transfers is much less than that of Hohmann transfers, although the transfer time increases.[1] However, transfers lasting less than 90 days

[1] The cost for an Hohmann transfer between similar orbits is about 3,950 m/s; the transfer time is about 5 days.

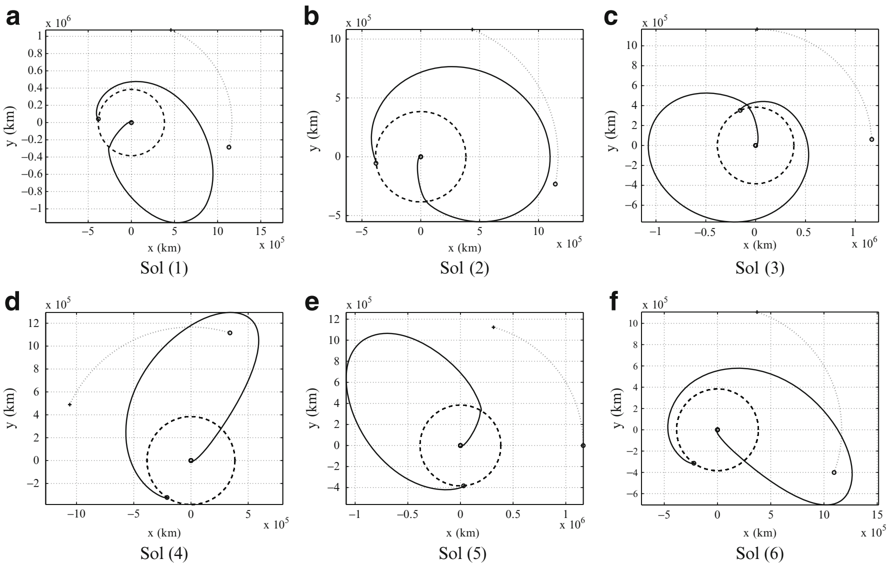


Fig. 16.4 Low-energy Earth–Moon transfers in the Earth-centered rotating frame. The Moon's orbit (*dashed*), the apparent orbit of the Sun (*dotted*, not to scale), and the position of the Sun at departure (" ∘ ") and arrival (" + ") are also shown

are nowadays deemed acceptable, as the increased transfer duration is balanced by the propellant mass saving [25].

16.4.2 Low-Thrust Transfers to Lagrangian Point Orbits

The transfers to Lagrangian point orbits exploit the stable manifold. To accomplish such transfers it is enough to place the spacecraft on the stable manifold emanating from the final libration point orbit. Thus, the optimization problem aims at targeting a point on the stable manifold while minimizing a performance index. This is done by using low-thrust propulsion.

Let us consider the dynamics of the controlled PCRTBP (16.19), and let us assume that the spacecraft lies on a circular Earth orbit of radius r_i. Thus, the first transfer state, $\mathbf{x}_i = (x_i, y_i, \dot{x}_i, \dot{y}_i, m_i)$, must satisfy the following initial boundary condition:

$$(x_i + \mu)^2 + y_i^2 - r_i^2 = 0 \, , \tag{16.32}$$

$$(x_i + \mu)(\dot{x}_i - y_i) + y_i(\dot{y}_i + x_i + \mu) = 0 \ , \tag{16.33}$$

$$(\dot{x}_i - y_i)^2 + (\dot{y}_i + x_i + \mu)^2 - (1 - \mu)/r_i = 0 \ . \tag{16.34}$$

It can be demonstrated that a generic point of the stable manifold $\mathbf{x}_s = (x_s, y_s, \dot{x}_s, \dot{y}_s)$ can be uniquely identified by using two scalars, i.e., $\mathbf{x}_s(\tau_1, \tau_2)$, where τ_1, τ_2 are two time variables [20]. In order to belong to the stable manifold, the final transfer state, $\mathbf{x}_f = (x_f, y_f, \dot{x}_f, \dot{y}_f, m_f)$, must satisfy the final boundary condition

$$x_f - x_s = 0 \ , \quad y_f - y_s = 0 \ , \quad \dot{x}_f - \dot{x}_s = 0 \ , \quad \dot{y}_f - \dot{y}_s \ . \tag{16.35}$$

The transfer is to use the least propellant as possible, and therefore the performance index is

$$J = m_i - m_f \ , \tag{16.36}$$

which corresponds to the propellant spent. Finally, the low-thrust magnitude, $T = (T_x^2 + T_y^2)^{1/2}$, must not exceed the maximum available thrust, $T_{\max}$. This is enforced through the following inequality condition:

$$\left[T_x^2(t) + T_y^2(t)\right]^{1/2} - T_{max} \le 0, \quad t \in [t_i,\, t_f] \ , \tag{16.37}$$

where t_i, t_f are the initial and final transfer times, respectively. Note that the reverse problem (i.e., transfer from a libration point orbit to an Earth orbit) can be defined by inverting the order of the boundary conditions (Eqs. (16.32)–(16.34) in place of Eqs. (16.35) and vice versa). The problem stated by Eqs. (16.32)–(16.37) can be transformed into a NLP through the direct transcription shown in Sect. 16.2.2. Thus, the low-thrust transfers can be stated as follows.

Definition 3 Find $\mathbf{y} = \{\mathbf{x}_1, \mathbf{T}_1, \ldots, \mathbf{x}_N, \mathbf{T}_N, t_1, t_N, \tau_1, \tau_2\}$ that satisfies

$$\min_{\mathbf{y}} F(\mathbf{y}) \text{ subject to } \mathbf{c}(\mathbf{y}) = 0 \text{ and } \mathbf{g}(\mathbf{y}) \le 0 \ , \tag{16.38}$$

where $F(\mathbf{y})$, $\mathbf{c}(\mathbf{y})$, and $\mathbf{g}(\mathbf{y})$ are the transcribed objective function (16.36), equality constraints (16.32)–(16.35), and inequality constraints (16.37), respectively.

The problem stated in Definition 3 has been solved with direct transcription and multiple shooting. In particular, an L_1–Earth–L_1 tour has been studied in the framework of an Earth–Moon transportation system. The system is made up by two trajectories, defined as follows:

- Trajectory (a)—in this transfer, a spacecraft (S1) moves from a given L_1 orbit and goes to a circular Earth orbit. The initial mass is equal to 50,000 kg.

Table 16.2 Parameters of the low-thrust transfers to/from the periodic orbit about L_1

Traj	m_i (kg)	m_f (kg)	m_p (kg)	t_m (days)	t_{lt} (days)	Δt (days)
(a)	50,000.0	47,553.6	2,446.4	38.24	15.96	54.20
(b)	75,153.6	71,685.6	3,468.0	38.24	24.47	62.71

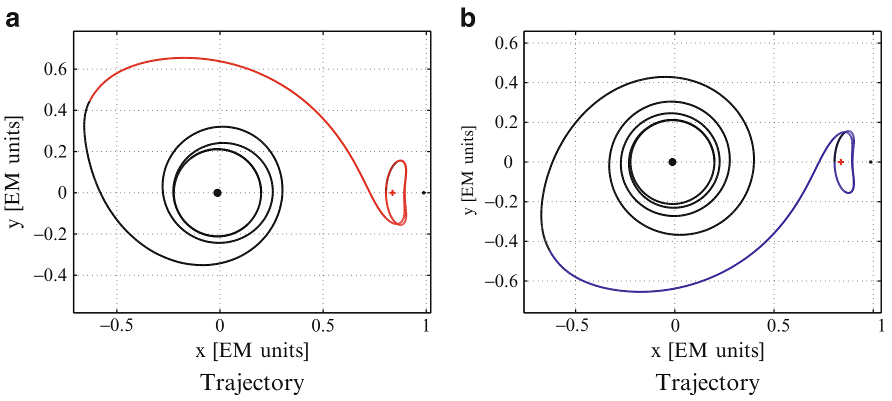


Fig. 16.5 Trajectory (**a**) and (**b**) shown in the Earth–Moon rotating frame; low-thrust orbit (*black*), unstable manifold (*red*), stable manifold (*blue*)

- Trajectory (b)—once in the Earth circular orbit, S1 docks with another spacecraft (S2) of mass 27,600 kg, and they move together (S1+S2) again to the L_1 orbit.

The Lyapunov orbit about L_1 has semi-amplitudes of 25,400 km and 60,000 km along x and y, respectively, whereas a circular Earth orbit of altitude 75,000 km has been selected. The maximum available thrust is 47 N, and the specific impulse is 1,620 s.

The optimal results are summarized in Table 16.2 and shown in Fig. 16.5 for each of the two trajectory legs. The columns of Table 16.2 report the initial and final mass of each leg (m_i and m_f) as well as the propellant mass needed for the transfer ($m_p = m_i - m_f$); t_m is the time spent on the manifold (unstable for Traj (a), stable for Traj (b)), t_{lt} is the duration of the low-thrust arc, and therefore the total transfer time is $\Delta t = t_m + t_{lt}$. These results show that orbits about the Earth–Moon L_1 can be accessed in reasonable times (departing from high-altitude Earth orbits) even when the thrust-to-mass ratio is small (on the order of 10^{-4} for both trajectory legs).

16.5 Conclusions

In this chapter the preliminary design and assessment of low-energy transfers have been treated under an optimization perspective. The principles of optimal control and trajectory optimization are applied to the dynamics of the three- and four-body problem through a direct transcription approach. This method's robustness allows us to deal with the high nonlinearities that characterize the low-energy transfers. Two different application cases are shown. The first is a classic Earth–Moon exterior low-energy transfer defined in the vector field generated by the Sun, the Earth, and the Moon. Optimal solutions requiring about 200 m/s less ΔV than the Hohmann transfers have been achieved. In the second application case, the low-thrust propulsion is used to reach a L_1 periodic orbit in the Earth–Moon system. It has been shown that the resulting low-energy, low-thrust transfers can be used in the context of an Earth–Moon transportation system.

References

1. Belbruno, E.: Lunar capture orbits, a method of constructing Earth–Moon trajectories and the lunar GAS mission. In: AIAA Paper 97–1054, Proceedings of the AIAA/DGLR/JSASS International Electric Propulsion Conference, Colorado Springs, Colorado (1987)
2. Belbruno, E.: The dynamical mechanism of ballistic lunar capture transfers in the four-body problem from the perspective of invariant manifolds and Hill's regions. In: Technical Report, Centre De Recerca Matematica, Barcelona, Spain (1994)
3. Belbruno, E.: Capture Dynamics and Chaotic Motions in Celestial Mechanics: With Applications to the Construction of Low Energy Transfers. Princeton University Press, New Jersey (2004)
4. Belbruno, E., Miller, J.: Sun-perturbed Earth-to-Moon transfers with ballistic capture. J. Guid. Contr. Dynam. **16**, 770–775 (1993)
5. Betts, J.: Survey of numerical methods for trajectory optimization. J. Guid. Contr. Dynam. **21**, 193–207 (1998)
6. Betts, J.: Practical Methods for Optimal Control Using Nonlinear Programming. SIAM, Philadephia (2000)
7. Bryson, A., Ho, Y.: Applied Optimal Control. Wiley, New York (1975)
8. Canalias, E., Cobos, J., Masdemont, J.: Impulsive transfers between lissajous libration point orbits. J. Astronaut. Sci. **51**, 361–390 (2003)
9. Enright, P., Conway, B.: Discrete approximations to optimal trajectories using direct transcription and nonlinear programming. J. Guid. Contr. Dynam. **15**, 994–1002 (1992)
10. Gómez, G., Jorba, A., Masdemont, J., Simó, C.: Study of the transfer from the Earth to a Halo Orbit around the Equilibrium Point L_1. Celestial Mech. Dyn. Astron. **56**, 541–562 (1993)
11. Gómez, G., Koon, W., Lo, M., Marsden, J., Masdemont, J., Ross, S.: Invariant manifolds, the spatial three-body problem and space mission design. Adv. Astronaut. Sci. **109**(1), 3–22 (2001)
12. Gómez, G., Masdemont, J., Mondelo, J.: Libration point orbits: a survey from the dynamical point of view. In: Proceedings of the Libration Point Orbits and Application Conference, Girona, Spain (2002)
13. Gómez, G., Mondelo, J.: The dynamics around the collinear equilibrium points of the RTBP. Phys. D **157**, 283–321 (2001)

14. Howell, K., Barden, B., Lo, M.: Application of dynamical systems theory to trajectory design for libration point missions. J. Astronaut. Sci. **45**, 161–178 (1997)
15. Jehn, R., Campagnola, S., García, D., Kemble, S.: Low-thrust approach and gravitational capture at mercury. In: Proceedings of the 18th International Symposium on Space Flights Dynamics, vol. 584, p. 487 (2004)
16. Jorba, A., Masdemont, J.: Dynamics in the center manifold of the collinear points of the restricted three body problem. Phys. D **132**, 189–213 (1999)
17. Koon, W., Lo, M., Marsden, J., Ross, S.: Heteroclinic connections between periodic orbits and resonance transitions in celestial mechanics. Chaos **10**, 427–469 (2000)
18. Koon, W., Lo, M., Marsden, J., Ross, S.: Low energy transfer to the Moon. Celestial Mech. Dyn. Astron. **81**, 63–73 (2001)
19. Lo, M., Ross, S.: Low energy interplanetary transfers using the invarian manifolds of L_1, L_2, and halo orbits. In: Paper AAS 98–136, Proceedings of the AAS/AIAA Space Flight Mechanics Meeting (1998)
20. Mingotti, G., Topputo, F., Bernelli-Zazzera, F.: Combined optimal low-thrust and stable-manifold trajectories to the earth–moon halo orbits. AIP Conference Proceedings **886**, 100–110 (2007). DOI 10.1063/1.2710047
21. Mingotti, G., Topputo, F., Bernelli-Zazzera, F.: Low-energy, low-thrust transfers to the moon. Celestial Mech. Dyn. Astron. **105**(1–3), 61–74 (2009). DOI 10.1007/ s10569-009-9220-7
22. Mingotti, G., Topputo, F., Bernelli-Zazzera, F.: Earth-Mars transfers with ballistic escape and low-thrust capture. Celestial Mech. Dyn. Astron. **110**(2), 169–188 (2011). DOI 10.1007/ s10569-011-9343-5
23. Mingotti, G., Topputo, F., Bernelli-Zazzera, F.: Optimal low-thrust invariant manifold trajectories via attainable sets. J. Guid. Contr. Dynam. **34**(6), 1644–1656 (2011). DOI 10.1007/s10569-011-9343-5
24. Mingotti, G., Topputo, F., Bernelli-Zazzera, F.: Efficient invariant-manifold, low-thrust planar trajectories to the moon. Comm. Nonlinear. Sci. Numer. Simulat. **17**(2), 817–831 (2012). DOI 10.1016/j.cnsns.2011.06.033
25. Roncoli, R., Fujii, K.: Mission design overview for the gravity recovery and interior laboratory (GRAIL) mission. In: Paper AIAA 2010–8383, AIAA Guidance, Navigation, and Control Conference, Toronto, Ontario, Canada, 2–5 August, 2010
26. Simó, C., Gómez, G., Jorba, A., Masdemont, J.: The bicircular model near the triangular libration points of the RTBP. From Newton to Chaos, pp. 343–370. Plenum Press, New York (1995)
27. Szebehely, V.: Theory of Orbits: The Restricted Problem of Three Bodies. Academic, London (1967)
28. Topputo, F., Vasile, M., Bernelli-Zazzera, F.: Earth-to-Moon low energy transfers targeting L_1hyperbolic transit orbits. Ann. New York Acad. Sci. **1065**, 55–76 (2005)
29. Topputo, F., Vasile, M., Bernelli-Zazzera, F.: Low energy interplanetary transfers exploiting invariant manifolds of the restricted three-body problem. J. Astronaut. Sci. **53**(4), 353–372 (2005)
30. Von Stryk, O., Bulirsch, R.: Direct and indirect methods for trajectory optimization. Ann. Oper. Res. **37**(1), 357–373 (1992)
31. Yagasaki, K.: Sun-perturbated earth-to-moon transfers with low energy and moderate flight time. Celestial Mech. Dynam. Astron. **90**, 197–212 (2004)