Lecture Notes in Applied and Computational Mechanics Vol. 49

Subhendu Bikash Hazra

Large-Scale PDE-Constrained Optimization in Applications

Springer

Lecture Notes in Applied and Computational Mechanics

Volume 49

Lecture Notes in Applied and Computational Mechanics

Edited by F. Pfeiffer and P. Wriggers

Further volumes of this series found on our homepage: springer.com

Vol. 49: Hazra, S.B.
Large-Scale PDE-Constrained Optimization in Applications
201 p. 2010 [978-3-642-01501-4]

Vol. 48: Su, Z.; Ye, L.
Identification of Damage Using Lamb Waves
346 p. 2009 [978-1-84882-783-7]

Vol. 47: Studer, C.
Numerics of Unilateral Contacts and Friction
191 p. 2009 [978-3-642-01099-6]

Vol. 46: Ganghoffer, J.-F., Pastrone, F. (Eds.)
Mechanics of Microstructured Solids
136 p. 2009 [978-3-642-00910-5]

Vol. 45: Shevchuk, I.V.
Convective Heat and Mass Transfer in Rotating Disk Systems
300 p. 2009 [978-3-642-00717-0]

Vol. 44: Ibrahim R.A., Babitsky, V.I., Okuma, M. (Eds.)
Vibro-Impact Dynamics of Ocean Systems and Related Problems
280 p. 2009 [978-3-642-00628-9]

Vol.43: Ibrahim, R.A.
Vibro-Impact Dynamics
312 p. 2009 [978-3-642-00274-8]

Vol. 42: Hashiguchi, K.
Elastoplasticity Theory
432 p. 2009 [978-3-642-00272-4]

Vol. 41: Browand, F., Ross, J., McCallen, R. (Eds.)
Aerodynamics of Heavy Vehicles II: Trucks, Buses, and Trains
486 p. 2009 [978-3-540-85069-4]

Vol. 40: Pfeiffer, F.
Mechanical System Dynamics
578 p. 2008 [978-3-540-79435-6]

Vol. 39: Lucchesi, M., Padovani, C., Pasquinelli, G., Zani, N.
Masonry Constructions: Mechanical Models and Numerical Applications
176 p. 2008 [978-3-540-79110-2]

Vol. 38: Marynowski, K.
Dynamics of the Axially Moving Orthotropic Web
140 p. 2008 [978-3-540-78988-8]

Vol. 37: Chaudhary, H., Saha, S.K.
Dynamics and Balancing of Multibody Systems
200 p. 2008 [978-3-540-78178-3]

Vol. 36: Leine, R.I.; van de Wouw, N.
Stability and Convergence of Mechanical Systems with Unilateral Constraints
250 p. 2008 [978-3-540-76974-3]

Vol. 35: Acary, V.; Brogliato, B.
Numerical Methods for Nonsmooth Dynamical Systems: Applications in Mechanics and Electronics
545 p. 2008 [978-3-540-75391-9]

Vol. 34: Flores, P.; Ambrósio, J.; Pimenta Claro, J.C.; Lankarani Hamid M.
Kinematics and Dynamics of Multibody Systems with Imperfect Joints: Models and Case Studies
186 p. 2008 [978-3-540-74359-0

Vol. 33: Nies ony, A.; Macha, E.
Spectral Method in Multiaxial Random Fatigue
146 p. 2007 [978-3-540-73822-0]

Vol. 32: Bardzokas, D.I.; Filshtinsky, M.L.; Filshtinsky, L.A. (Eds.)
Mathematical Methods in Electro-Magneto-Elasticity
530 p. 2007 [978-3-540-71030-1]

Vol. 31: Lehmann, L. (Ed.)
Wave Propagation in Infinite Domains
186 p. 2007 [978-3-540-71108-7]

Vol. 30: Stupkiewicz, S. (Ed.)
Micromechanics of Contact and Interphase Layers
206 p. 2006 [978-3-540-49716-5]

Vol. 29: Schanz, M.; Steinbach, O. (Eds.)
Boundary Element Analysis
571 p. 2006 [978-3-540-47465-4]

Vol. 28: Helmig, R.; Mielke, A.; Wohlmuth, B.I. (Eds.)
Multifield Problems in Solid and Fluid Mechanics
571 p. 2006 [978-3-540-34959-4

Vol. 27: Wriggers P., Nackenhorst U. (Eds.)
Analysis and Simulation of Contact Problems
395 p. 2006 [978-3-540-31760-9]

Large-Scale PDE-Constrained Optimization in Applications

Subhendu Bikash Hazra

Springer

Dr. habil. Subhendu Bikash Hazra
FG Strömungsdynamik
FB Maschinenbau
TU Darmstadt
Petersenstr. 30
64287 Darmstadt
Germany

ISBN: 978-3-642-01501-4 e-ISBN: 978-3-642-01502-1

DOI 10.1007/ 978-3-642-01502-1

Lecture Notes in Applied and Computational Mechanics ISSN 1613-7736

e-ISSN 1860-0816

Library of Congress Control Number: 2009940445

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1 0

springer.com

Preface

With continuous development of modern computing hardware and applicable numerical methods, computational fluid dynamics (CFD) has reached certain level of maturity so that it is being used routinely by scientists and engineers for fluid flow analysis. Since most of the real-life applications involve some kind of optimization, it has been natural to extend the use of CFD tools from flow simulation to simulation based optimization. However, the transition from simulation to optimization is not straight forward, it requires proper interaction between advanced CFD methodologies and state-of-the-art optimization algorithms. The ultimate goal is to achieve optimal solution at the cost of few flow solutions. There is growing number of research activities to achieve this goal.

This book results from my work done on simulation based optimization problems at the Department of Mathematics, University of Trier, and reported in my postdoctoral thesis *("Habilitationsschrift")* accepted by the Faculty-IV of this University in 2008. The focus of the work has been to develop mathematical methods and algorithms which lead to efficient and high performance computational techniques to solve such optimization problems in real-life applications. Systematic development of the methods and algorithms are presented here. Practical aspects of implementations are discussed at each level as the complexity of the problems increase, supporting with enough number of computational examples. It consists of two parts: first part deals with time dependent optimization problems with applications in environmental engineering and the second part deals with steady state optimization problems, in which the PDEs are solved using semi-iterative or pseudo-time-stepping techniques, with applications in aerodynamics.

This book will be useful for scientists and engineers who are looking for efficient numerical methods for PDE-constrained optimization problems. It will be helpful for graduate and Ph.D. students in applied mathematics, aerospace engineering, mechanical engineering, civil engineering and computational engineering during their training and research. This also will provide exciting research and development areas involving realistic applications.

Beside the acknowledgments of the thesis appearing in the next page, I would like to thank Prof. V. Schulz, Prof. E. Sachs and Prof. O. Ghattas for acting as

referees of my postdoctoral thesis. Also, thanks are due to Prof. M. Oberlack and his group at the chair of fluid dynamics, Technical University of Darmstadt, for their moral support and encouragements towards this publication. Finally, thanks are due to Dr. T. Ditzinger and Ms. H. King of Springer-Verlag Heidelberg for their help and cooperations.

Darmstadt,
August 2009 Subhendu Bikash Hazra

Acknowledgements

First of all, I would like to express my sincere thanks to Prof. Dr. Volker Schulz for all his support, cooperations and discussions during my stay in Trier as well as in Berlin. It has been exciting to extend the CFD applications to the filed of nonlinear optimization with his initiation.

Some of the works reported here have been resulted from direct collaborations with other Universities and research Institutions. I would like to thank Prof. Dr. Rainer Helmig, Prof. Dr. Gabriel Wittum and Prof. Dr. Peter Bastian for the collaboration on parameter identification problems and for their cooperations and discussions on MUFTE-UG code. Also, thanks are due to their group members, in particular to, Dr. Holger Class, Dr. Hussam Sheta and Mr. David for their discussions and cooperations. Thanks to Dr. Arne Farber for providing with experimental data from VEGAS which have been used in these applications.

The other direct collaboration has been with DLR, Braunschweig, in the problems of aerodynamic shape optimization. I would like to thank this organization for giving access to the FLOWer code. Also thanks are due to Dr. Nicolas Gauger and Mr. Joel Brazeilon for all their discussions and cooperations on FLOWer code. Thanks to Prof. Dr. Olaf Fromann for his cooperations on SynapaPointerPro optimization platform.

During my stay in Trier, I have profitted very much from my two research stays in USA. Prof. Omar Ghattas gave me an opportunity to visit the Computational Science group of Civil Engineering Department, Carnegie Mellon University. My sincere thanks to Prof. Ghattas for giving me this opportunity as well as for his continuing support, cooperations and discussions. A large number of people were working in his group on various application areas of PDE-constrained optimization and I enjoyed very much the discussions and the experiences that the group shared during my stay there. Thanks to all of them for all discussions, cooperations as well as their enjoyable company during the whole visit, specially to Dr. Volkan Accenik and Dr. Alexander Cunha with whom I worked directly. Through Prof. Ghattas's initiation, I also had opportunity to have discussions with Prof. Shlomo Ta'asan and Prof. Larry Biegler during my stay at CMU. Thanks are due to both of them for their time and sharing their knowledge and experience in this field. Also it was great

experience in meeting with Prof. Jacob Bielek and Prof. Amit Acharya in the CS group.

Prof. Antony Jameson gave me an opportunity to visit his Department at Stanford University. It has been great experience to have direct collaboration with him in aerodynamic shape optimization problems. My sincere thanks are due to him for giving me the opportunity to visit him, as well as for all his support, cooperations, discussions and also for giving access to his SYN103 code. Thanks are due to his group members, specially to, Dr. Sriram Shankaran, Dr. Georg May, and Dr. Arathi Gopinath for their discussions and cooperations.

I would like to thank the colleagues in the Department for all support, cooperations and discussions, academic as well as non-academic, during last 5 years. I enjoyed the nice working environment provided by them. Also thanks are due to Dr. Manfred Ries and Mr. Benedikt Wilbertz for providing with nice computing environment and instant solution to any problem related to computers.

I am grateful to DFG for providing financial support to attend AIAA conferences, from which I have profitted very much.

Special regards and extreme gratefulness are due to my parents and other family members in India for their great patience and continuous moral support throughout my stay abroad. My wife and son have been here with me and they are the one who gave me all inspiration and moral support throughout. I have taken away a lot of time from them to complete this work. Thanks for all their support and sacrifices.

Subhendu Bikash Hazra
Trier

Contents

List of Figures

List of Tables

Acronyms

A	approximate Jacobian
B	reduced Hessian
B_F	farfield boundary
B_S	solid body (3D)
C	solid wall (2D)
C_D	drag coefficient
C_L	lift coefficient
C_M	pitching Moment
C_{ref}	chord length
C_p	pressure coefficient
c_p	specific heat at constant pressure
c_v	specific heat at constant volume
E	total energy
g	reduced gradient
$\mathbf{g}$	vector of gravitational acceleration
H	total enthalpy
I	objective functional
J	Jacobian
$\mathbf{K}$	permeability tensor
$k_{r\alpha}$	relative permeability
L	Lagrangian
M	Mach number
$\mathbf{n}$	unit outward normal
P	preconditioner
Re	Reynolds number
p	pressure
p_α	pressure of phase α
p_c	capillary pressure
Pr	Prandtl number
q	vector of design variables
s	enthalpy per unit mass

S_{ref} reference area
S_α saturation of phase α
T absolute temperature
u velocity vector
(u_1, u_2, u_3) velocity components
v_α velocity of phase α
w vector of state variables
X_α^κ mass fraction of component κ in phase α
(x, y, z) Cartesian coordinates
(x_1, x_2, x_3) Cartesian coordinates
α angle of attack
γ ratio of specific heats
λ vector of adjoint variables
ρ density
ρ_α density of phase α
ϕ porosity
μ dynamic viscosity
Ω flow field domain
$\partial\Omega$ flow field boundary
(ξ_1, ξ_2, ξ_3) generalized coordinates

Chapter 1
Introduction

Due to advancement in computer technology and availability of efficient numerical algorithms, scientific computing has become an essential tool for scientists and engineers in academia as well as in industry. Most of the natural and physical processes are modeled mathematically by a system of nonlinear Partial Differential Equations (PDEs). Because of the complexity of the PDEs or of the application domains, analytical solutions to these equations do not exist in general. The only way out is to look for approximate numerical solutions. Therefore, PDE-simulation is wide spread in scientific and engineering applications. As progress has been made in scientific computing tools, more and more accurate, and hence more complex, PDE-system is being considered for model prediction.

Optimization is also involved in almost all natural, industrial and physical processes. The processes which are described by PDEs, optimization in those processes means optimization involving PDEs. That is why these optimization problems are called PDE-constrained optimization or simulation based optimization. Since the PDEs, in most cases, are nonlinear and/or the objective function involved in the optimization problem is nonlinear, they are also called Nonlinear Programming Problems (NLPs). For these problems also analytical solutions are nonexistent and, therefore, numerical solutions are sought applying scientific computing. Few examples of such problems are optimal design, optimal control and parameter identification.

The optimization problems are challenging in one hand and on the other hand they are very demanding since they are cost effective alternative to the high cost experiments. Parallel research has been going on in PDE-simulation and in numerical optimization. Currently there is a growing tendency of cooperation and collaboration among these two communities in order to achieve the best possible results in practical applications. Key challenges and open problems in PDE constrained optimization are discussed in [16]. One of the key challenges lies in efficient integration of nonlinear programming algorithms with advanced PDE solvers.

PDE-simulation is challenging due to various reasons. Firstly, the equations are nonlinear, posing the difficulty in existence and uniqueness of solutions. Also, in many cases the type of equations change with the change in physical nature of the problem at different parts of the computational domain, i.e., in some part they are

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 1–4.
springerlink.com

of hyperbolic type, in some other parts they are of elliptic or parabolic type. This requires special treatment in order to get physically meaningful solutions. The physical domain of the problem is sometimes quite complicated posing the difficulty in satisfying the boundary conditions correctly, which is essential for finite dimensional representation. Another difficulty lies in the fact that the problem is usually of very large size, specially in 3D, involving large number of state variables and state equations. All these are to be addressed in an efficient PDE-solver.

Challenges also lie in the numerical algorithm for the optimization problems. Since the problems are nonlinear, usually they are solved using Newton-type methods applied to the necessary optimality conditions. If the problem involves inequality constraints, these Newton-type approaches are to be extended to deal with the constraints. One such method is Sequential Quadratic Programming (SQP) methods or its reduced variants, rSQP methods. The SQP or rSQP methods are gradient based methods and require gradient information in order to find an optimum. Depending on handling the constraints (which are the underlying PDEs) in the implementation of the optimization algorithms, different methods can be resulted. If a nonlinear elimination of the state variables is considered by solving the state equations exactly in the PDE-solver, then the optimization problem will be reduced to an unconstrained one involving only the (reduced) objective function. Optimization algorithms for solving the problem in this way are known as black-box or *Nested Analysis and Design* (NAND) implementation. This implementation requires almost no interaction of the NLP algorithm and the PDE-solver. Since the optimization algorithm assumes the state variables are solved exactly by the PDE-solver, in each iteration of this algorithm a well converged state solution is required. This leads to high computational cost of the algorithm.

The other implementation of the optimization considers no elimination of the state variables from the optimization problem. Rather, this involves complete discretization of the PDE model in the KKT-system. Since the PDE-solver is a part of the NLP algorithm, this approach is called *Simultaneous Analysis and Design* (SAND) approach. This is generally faster than NAND approach since it leads to convergence of the PDE-solver and the NLP algorithm simultaneously.

Computation of gradients required by the (gradient based) NLP algorithms is another challenge. One can use, for example, finite difference method which is the simplest and easiest to implement. However, this leads to huge computational cost in case of large number of decision variables since this requires $(n+1)$-state solutions for n-decision variables. The other alternatives are to use *adjoint approach* or *direct tailored approach*.

Several areas of applications have motivated the development of PDE constrained optimization. Few references reporting research findings on this subject are [16, 160, 153, 15]. Few references addressing specific application areas such as optimal control or shape and topology optimization in computational fluid dynamics are [19, 122, 31, 48, 83, 110, 79, 52], control of chemical processes are [154, 166], data assimilation in regional weather prediction modeling is [167], and parameter identification are [147, 152, 35, 101].

Non-stationary PDE constrained optimization problems involve additional levels of difficulty due to transient simulations. Several approaches have been considered for such problems. Reduced order modeling approach based on Proper Orthogonal Decomposition have been considered in [39, 38, 111]. Another approach is to utilize sensitivity calculations for differential algebraic equations (DAEs) for reduced-gradient calculation. By converting a system of PDEs to DAEs, various methods, such as multiple shooting, can be used to discretize in time [50]. Recently, adjoint sensitivity in transient simulation, which are not efficient due to large storage requirements, has also been considered in [2].

This book addresses numerical methods for PDE constrained optimization problems. Special emphasis has been given to two application areas, namely, parameter identification in multiphase flow through porous media and aerodynamic shape optimization. First problem class involve non-stationary PDEs and they have been solved using multiple shooting method for DAEs. This has been possible since in our case the number of parameters is small compared to state variables. In the second problem class, the PDEs are nonlinear hyperbolic or mixed hyperbolic-parabolic type and the CFD is extremely expensive, specially when full convergence of the PDE solution is sought. For these problems we developed a new rSQP methods based one-shot pseudo-time-stepping method. The method is quite efficient and tested for both inviscid as well as viscous flow problems in 2D as well as in 3D. Further efficiency of the method could be brought in through application of a multigrid strategy.

The organization of the book is as follows. In the next chapter we give a brief derivation of the PDEs which model the fluid flow problems. In Chapter 3, we discuss briefly the optimization methods for PDE constrained optimization problems. Rest of the thesis has two parts. In the first part, problems of parameter identification have been considered. In Chapter 4, we present the fluid flow models for multiphase flow through porous media. Since the flow velocity in this case is very small, one need not solve the Navier-Stokes equations, instead generalized Darcy's law can be considered. In Chapter 5, we discuss the numerical method for parameter identification.

In the second part of the thesis, problems of aerodynamic shape optimization have been addressed. In Chapter 6, we discuss in detail the new one-shot pseudo-time-stepping method. The method is applied to an academic test problem in this chapter. In the next chapter, we have applied the method to problems of aerodynamic shape optimization. Since practical problems of aerodynamic shape optimization involve additional state constraints, we extend the method to such problems. We have considered two ways of treating the state constraints. In Chapter 8, we discuss the indirect way of treating the constraint; that means, we perform some kind of reduction strategy to add the constraint (in a Lagrangian way) to the objective function with some weighting and thereby reducing the constrained problem to an unconstrained one. This kind of treatment has been adopted in most of the practical applications of this problem class. However, it is well known that the reduced problem may not always correspond to the original one and the method applied to solve it may not be the efficient one. So, we consider another (more direct) way of treating

the state constraints in the next chapter. The method reduces the computational cost upto 80% of that required by a 'black-box' gradient method. However, the number of optimization iterations is relatively large in one-shot pseudo-time-stepping method. In Chapter 10, we incorporate a multigrid strategy in the context of one-shot pseudo-time-stepping method. The method is applied to shape optimization problem without state constraint in that chapter. This reduces the number of optimization iterations upto 65% of that required by a single grid computation. In the next chapter, we extend the multigrid method to state constrained shape optimization problems. In Chapter 12, we extend the one-shot pseudo-time-stepping method for aerodynamic shape optimization problems in compressible viscous flow governed by Reynolds averaged Navier-Stokes equations. This completes the tests of the method for a wide range of problems in inviscid, as well as in viscous, 2D and 3D problems.

Chapter 2
Partial Differential Equations in Mathematical Modeling of Fluid Flow Problems

2.1 Introduction

In this chapter, a brief description of governing equations modeling fluid flow problems is given. A detailed derivation and explanation of the equations can be found, for example, in [144, 10, 28, 125]. There are two distinct descriptions of fluid motion, namely, Lagrangian and Eulerian, both of which are based on continuum principles. The Lagrangian description is based on identifying the individual 'element' of fluid in motion. This leads to the idea of associating fluid motion with a geometrical transformation represented by a function $x = x(b,t)$ which gives the position vectors x at various times t of the 'element' of fluid identified by the label b (which denotes the position vector at time $t = 0$). The function $x(b,t)$ is assumed to be continuous with respect to both of its arguments, and its inverse is similarly continuous.

Definition 1. (**Fluid Motion**)**:** Let Ω_0 be any open, bounded point-set in $\mathbb{R}^3$ occupied by fluids at time $t = 0$. Fluid motion is described by a transformation ψ_t on the closure $\bar{\Omega}_0$ into $\mathbb{R}^3$ such that the point set $\psi_t \Omega_0$ is that occupied by the same fluid at time t.

Remark: The point set $\psi_t \Omega_0$ is also open and bounded.

Definition 2. (**Fluid Velocity**)**:** The velocity is defined as $u = \frac{\partial}{\partial t} x(b,t)$ on the domain of $x(b,t)$.

A description of fluid motion by means of velocity field u, which is a function of x and t, is called Eulerian.

The basis of fluid dynamics is part Eulerian and part Lagrangian and can be transformed from one representation to the other. If any physical quantity has Eulerian representation $f(x,t)$, its Lagrangian representation is $\hat{f}(b,t) = f(x(b,t),t)$ and its material or convective derivative is defined as

$$\frac{Df}{Dt} = \frac{\partial}{\partial t} \hat{f}(b,t).$$

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 5–18.
springerlink.com

Its relation to the partial derivative $\partial f/\partial t$ is given by

$$\frac{Df}{Dt} = \frac{\partial \hat{f}}{\partial t} = \left(\frac{\partial}{\partial t} + u.\text{grad}\right) f.$$

Definition 3. (**Fluid Acceleration**)**:** Fluid acceleration can be defined as

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + (u.\text{grad})u.$$

Physical system of fluid flow problems are described by basic conservation laws. This means, during the evolution of a fluid, certain properties such as mass, momentum and energy remain conserved. Conservation laws are described in terms of the following convection theorem.

Theorem 1. ***(Convection Theorem)****: If Ω_t is a fluid domain and if $f(x,t) \in C^1(\bar{\Omega}_t)$, then*

$$\frac{D}{Dt}\int\limits_{\Omega_t} f\,dV = \int\limits_{\Omega_t}\left(\frac{\partial f}{\partial t} + div(fu)\right) dV, \tag{2.1}$$

where dV is the volume element.

Proof: Since the domain of integration $\Omega_t = \psi_t\Omega_0$ depends on time t, where Ω_0 is a bounded fixed set in b-space, we use the transformation

$$\int\limits_{\Omega_t} f(x,t)\,dV = \int\limits_{\Omega_0} \hat{f}(b,t)J(b,t)\,dV_0,$$

where $J(b,t)$ is the Jacobian of the transformation $x(b,t)$. Thus

$$\begin{aligned}\frac{D}{Dt}\int\limits_{\Omega_t} f\,dV &= \frac{\partial}{\partial t}\int\limits_{\Omega_0} \hat{f}(b,t)J(b,t)\,dV_0\\ &= \int\limits_{\Omega_0}\frac{\partial}{\partial t}(\hat{f}J)\,dV_0\\ &= \int\limits_{\Omega_0}(J\frac{\partial \hat{f}}{\partial t} + \hat{f}\frac{\partial J}{\partial t})\,dV_0.\end{aligned}$$

We require the time derivative of the Jacobian, which is, as given in the following Lemma,

$$\frac{\partial J}{\partial t} = J\text{div}(u).$$

Hence, we get after substitution,

$$\begin{aligned}\frac{D}{Dt}\int\limits_{\Omega_t} f\,dV &= \int\limits_{\Omega_0}\left\{\frac{\partial \hat{f}}{\partial t}+\hat{f}\,\mathrm{div}(u)\right\}J\,dV_0\\ &= \int\limits_{\Omega_t}\left\{\frac{Df}{Dt}+f\,\mathrm{div}(u)\right\}dV\\ &= \int\limits_{\Omega_t}\left\{\frac{\partial f}{\partial t}+(u.\mathrm{grad})f+f\,\mathrm{div}(u)\right\}dV\\ &= \int\limits_{\Omega_t}\left\{\frac{\partial f}{\partial t}+\mathrm{div}(fu)\right\}dV.\end{aligned} \tag{2.2}$$

□

Lemma 1

$$\frac{\partial J(b,t)}{\partial t} = J(b,t)\,div\,(u\,(x(b,t),t)).$$

Proof: We write the components of x as $\xi(b,t),\eta(b,t)$, and $\zeta(b,t)$. The determinant J of a matrix is multiliner in the columns (or rows), its derivative is the sum of determinants. Thus, holding b fixed throughout, we have

$$\frac{\partial}{\partial t}J = \begin{vmatrix} \frac{\partial}{\partial t} & \frac{\partial \xi}{\partial b_1} & \frac{\partial \eta}{\partial b_1} & \frac{\partial \zeta}{\partial b_1}\\ \frac{\partial}{\partial t} & \frac{\partial \xi}{\partial b_2} & \frac{\partial \eta}{\partial b_2} & \frac{\partial \zeta}{\partial b_2}\\ \frac{\partial}{\partial t} & \frac{\partial \xi}{\partial b_3} & \frac{\partial \eta}{\partial b_3} & \frac{\partial \zeta}{\partial b_3}\end{vmatrix} + \begin{vmatrix} \frac{\partial \xi}{\partial b_1} & \frac{\partial}{\partial t} & \frac{\partial \eta}{\partial b_1} & \frac{\partial \zeta}{\partial b_1}\\ \frac{\partial \xi}{\partial b_2} & \frac{\partial}{\partial t} & \frac{\partial \eta}{\partial b_2} & \frac{\partial \zeta}{\partial b_2}\\ \frac{\partial \xi}{\partial b_3} & \frac{\partial}{\partial t} & \frac{\partial \eta}{\partial b_3} & \frac{\partial \zeta}{\partial b_3}\end{vmatrix} + \begin{vmatrix} \frac{\partial \xi}{\partial b_1} & \frac{\partial \eta}{\partial b_1} & \frac{\partial}{\partial t} & \frac{\partial \zeta}{\partial b_1}\\ \frac{\partial \xi}{\partial b_2} & \frac{\partial \eta}{\partial b_2} & \frac{\partial}{\partial t} & \frac{\partial \zeta}{\partial b_2}\\ \frac{\partial \xi}{\partial b_3} & \frac{\partial \eta}{\partial b_3} & \frac{\partial}{\partial t} & \frac{\partial \zeta}{\partial b_3}\end{vmatrix}.$$

We further have,

$$\begin{aligned}\frac{\partial}{\partial t}\frac{\partial \xi}{\partial b_1} &= \frac{\partial}{\partial b_1}\frac{\partial \xi}{\partial t} = \frac{\partial u}{\partial b_1},\\ \frac{\partial}{\partial t}\frac{\partial \xi}{\partial b_2} &= \frac{\partial}{\partial b_2}\frac{\partial \xi}{\partial t} = \frac{\partial u}{\partial b_2},\\ &\vdots\\ \frac{\partial}{\partial t}\frac{\partial \zeta}{\partial b_3} &= \frac{\partial}{\partial b_3}\frac{\partial \zeta}{\partial t} = \frac{\partial w}{\partial b_3}.\end{aligned}$$

The velocity components u_1,u_2,u_3 of u in these expressions are functions of b_1,b_2 and b_3 through $x(b,t)$; therefore,

$$\frac{\partial u_1}{\partial b_1} = \frac{\partial u_1}{\partial \xi}\frac{\partial \xi}{\partial b_1} + \frac{\partial u_1}{\partial \eta}\frac{\partial \eta}{\partial b_1} + \frac{\partial u_1}{\partial \zeta}\frac{\partial \zeta}{\partial b_1},$$
$$\vdots$$
$$\frac{\partial u_3}{\partial b_3} = \frac{\partial u_3}{\partial \xi}\frac{\partial \xi}{\partial b_3} + \frac{\partial u_3}{\partial \eta}\frac{\partial \eta}{\partial b_3} + \frac{\partial u_3}{\partial \zeta}\frac{\partial \zeta}{\partial b_3}.$$

Substituting these into the above expression, we get

$$\frac{\partial}{\partial t}J = \frac{\partial u_1}{\partial \xi}J + \frac{\partial u_2}{\partial \eta}J + \frac{\partial u_3}{\partial \zeta}J = \operatorname{div}(u)J. \qquad \square$$

Conservation of mass

A function $\rho(x,t)$, the density of the material at time t and position x, is defined on the closure of any fluid domain $\Omega_t = \psi_t \Omega_0$ so that for all t

$$\int_{\Omega_t} \rho \, dV = m(\Omega_0) > 0.$$

m is called the mass of the fluid in the domain Ω_t. It follows from (2.1) that

$$\frac{D}{Dt}\int_{\Omega_t} \rho \, dV = \int_{\Omega_t} \left(\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho u) \right) dV = 0, \tag{2.3}$$

for every fluid domain Ω_t. Using Divergence theorem, one gets

$$\int_{\Omega_1} \operatorname{div}(\rho u)\, dV = \int_{\partial\Omega_1} \rho u \cdot n \, dS, \tag{2.4}$$

and one gets

$$\frac{\partial}{\partial t}\int_{\Omega_1} \rho \, dV = -\int_{\partial\Omega_1} \rho u \cdot n \, dS, \tag{2.5}$$

for every fixed domain Ω_1 with regular boundary $\partial\Omega_1$ occupied by fluid during a time interval. Since equation 2.3 is true for any Ω_t, one gets

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho u) = 0, \tag{2.6}$$

everywhere in the fluid. This is usually called the *mass conservation law* or *equation of continuity*.

A flow is called *incompressible* if $\rho(x,t)$, the density, of each material particle remains the same during the motion. Lemma 1 is useful to understand this notion as stated in the following lemma.

Lemma 2. *The following statements are equivalent:*

a) the flow is incompressible
b) $div(u) = 0$
c) $J = 1$.

Conservation of linear momentum
A vector field f and a tensor field with components p_{ij} are defined on the closure of any fluid domain Ω_t with regular boundary surface $\partial\Omega_t$. If $\tilde{m}$ denotes the momentum of fluid in Ω_t, then

$$\tilde{m} = \int_{\Omega_t} \rho u \, dV.$$

The rate of change of $\tilde{m}$ is given by Newton's second law as:

$$\frac{D\tilde{m}}{Dt} = \sum \text{Forces acting on the fluid.}$$

Two types of forces can be acted on the fluid, namely, the body force, ρf, (such as gravity, centrifugal forces etc.) acting directly on each volume element, and the surface force caused by stress on the surface of the fluid. Thus, one gets

$$\frac{D}{Dt}\int_{\Omega_t} \rho u_i \, dV = \int_{\Omega_t} \rho f_i \, dV + \int_{\partial\Omega_t} p_{ij}\eta_j \, dS, \qquad i = 1,2,3, \tag{2.7}$$

where η is the unit outward normal to $\partial\Omega_t$ and p_{ij} is the stress tensor. This can also be written using the transport theorem, in purely Eulerian form, as

$$\int_{\Omega} \frac{\partial}{\partial t}(\rho u_i) \, dV + \int_{\partial\Omega} \rho u_i u.\eta \, dS = \int_{\Omega} \rho f_i \, dV + \int_{\partial\Omega} p_{ij}\eta_j \, dS, \qquad i = 1,2,3, \tag{2.8}$$

where Ω is any domain in $\mathbb{R}^3$ occupied by the fluid whose boundary $\partial\Omega$ is a regular surface with unit outward normal η.

In order to complete the system of equations, it is necessary to describe the constitutive relationship between p_{ij} and the tensor of velocity derivatives $\frac{\partial u_i}{\partial x_k}$, $(i,k = 1,2,3)$. The tensor of velocity derivatives can be split into its symmetric part, the rate of strain tensor,

$$e_{ik} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\right) = e_{ki},$$

and its antisymmetric part

$$\bar{e}_{ik} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_k} - \frac{\partial u_k}{\partial x_i}\right) = -\bar{e}_{ki}.$$

The antisymmetric tensor $\bar{e}_{ik}$ is related to a vector

$$\text{curl}(u) = \omega,$$

called the vorticity.

Conservation of Angular Momentum
In the absence of a body couple field,

$$\frac{D}{Dt}\int_{\Omega_t} x \wedge (\rho u)\, dV = \int_{\Omega_t} x \wedge (\rho f)\, dV + \int_{\partial\Omega_t} x \wedge (\hat{t})\, dS,$$

for every fluid domain Ω_t with regular boundary surface $\partial\Omega_t$, where $\hat{t}$ is the vector with components $\hat{t}_i = p_{ij}\eta_j$. The implication of the conservation of angular momentum is the following:

Theorem 2. *Above statement of conservation of angular momentum is equivalent to the condition that the stress tensor is symmetrical, i.e., $p_{ik} = p_{ki}$ for $i,k = 1,2,3$.*

A proof of the theorem can be found, for example, in [125].

Theorem 3. *If p_{ij} is symmetrical, linear in $\partial u_k/\partial x_m$, independent of $u, \partial u/\partial t, \bar{e}_{km}$ and higher derivatives of u, and if the relation between the tensors p_{ij} and e_{km} is isotropic (i.e., invariant under rotation of the coordinate system), then*

$$p_{ij} = -(p - \mu^{'} e_{kk})\delta_{ij} + 2\mu e_{ij} = -p\delta_{ij} + \tau_{ij}, \tag{2.9}$$

where μ and $\mu^{'}$ are scalars and known, respectively, as first and second coefficients of viscosity, δ_{ij} represents the unit tensor and p is the isotropic pressure. τ_{ij} is called the viscous stress tensor.

A proof of the theorem can be found in [141] and also in other books on continuum mechanics. Fluids satisfying equation (2.9) are called Newtonian fluids, which have been considered in this study. If equation (2.9) is used in equation (2.8), the equation of motion becomes, after using the continuity equation and the Divergence theorem,

$$\rho\frac{Du_i}{Dt} = \rho f_i - \frac{\partial}{\partial x_i}\left(p - \mu^{'}\frac{\partial u_k}{\partial x_k}\right) + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]. \tag{2.10}$$

Conservation of energy
In thermodynamics, physical properties, called variables of state, such as pressure p, absolute temperature T, density ρ, entropy s per unit mass, enthalpy H per unit mass are related by the First Law of Thermodynamics:

$$dE = Tds + \frac{p}{\rho^2}d\rho.$$

This law postulates that any thermodynamic system possesses a variable of state, the total energy $E = H - p/\rho$ per unit mass, such that, in any transition from one state to another, the difference between the internal energy change and the work

done by external forces on unit mass of the system must be supplied to the system in the form of heat. For a fluid in motion, the total energy (E) is the sum of kinetic energy $|u|^2/2$ per unit mass and the internal energy e. The work done by external forces is discussed earlier. Heat can be supplied to the body of fluid by conduction and radiation, of which the later is ignored in much of the gas dynamic literature. The conduction of heat is described by heat flux vector q. Thus, for a fluid in local thermodynamic equilibrium the following equation holds

$$\frac{D}{Dt}\int_{\Omega_t}\rho E\,dV=\int_{\Omega_t}\rho f\cdot u\,dV+\int_{\partial\Omega_t}p_{ij}\eta_j u_i\,dS-\int_{\partial\Omega_t}q\cdot\eta\,dS, \tag{2.11}$$

for every fluid domain Ω_t with regular boundary $\partial\Omega_t$. The heat flux vector is given by Fourier's law,

$$q=k\,\mathrm{grad}\,T, \tag{2.12}$$

where k is the coefficient of thermal conductivity. Using this relation and applying Gauss theorem, above equation can be written as

$$\int_{\Omega_t}\left\{\frac{\partial}{\partial t}(\rho E)+\frac{\partial}{\partial x_j}(\rho uE)-\rho u_i f_i-\frac{\partial}{\partial x_j}\left(u_i p_{ij}+k\frac{\partial T}{\partial x_j}\right)\right\}dV=0. \tag{2.13}$$

Since this holds for every fluid domain Ω_t, and hence also for every open subset of such domain, the integrand must vanish identically, and hence,

$$\frac{\partial}{\partial t}(\rho E)+\frac{\partial}{\partial x_j}(\rho uE)-\rho u_i f_i-\frac{\partial}{\partial x_j}\left(u_i p_{ij}+k\frac{\partial T}{\partial x_j}\right)=0. \tag{2.14}$$

As shown above, the state variables ρ, e, T and p are connected by thermodynamic relationships (assuming local thermodynamic equilibrium). We consider the case of a simple fluid such that all its thermodynamic properties can be deduced from a single fundamental relationship which, for a compressible fluid, can be chosen of type

$$s=s(\rho,e).$$

From this relationship, the pressure p and the temperature T are obtained in terms of the basic variables ρ and e from

$$p=-\rho^2 T\left(\frac{\partial s}{\partial\rho}\right)_e,\quad T=\frac{1}{(\partial s/\partial e)_\rho}.$$

An important special case is a perfect gas with constant specific heats c_p and c_v. For such a gas the laws of state are

$$p=(\gamma-1)\rho e,\quad \gamma=\frac{c_p}{c_v},\ \text{and}\ e=c_v T.$$

The viscosity and thermal conductivity coefficients depend on local thermodynamic state; in most cases they depend only on the temperature:

$$\mu = \mu(T),\ \mu^{'} = \mu^{'}(T),\ k = k(T).$$

From the second law of thermodynamics, the dissipation function Φ can not be negative. It can be shown that this leads to the conditions that

$$3\mu^{'} + 2\mu \geq 0, \quad \mu \geq 0,$$

and, in the absence of internal relaxation phenomena which would involve departure from local thermodynamic equilibrium, the Stokes relationship

$$3\mu^{'} + 2\mu = 0$$

is generally accepted as valid approximation.

2.1.1 The Navier-Stokes Equations for Compressible Viscous Flow

Here we collect the differential equations deduced above governing the motion of compressible, viscous (Newtonian) fluid. Conservation of mass is expressed by equation (2.6)

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho u) = 0, \tag{2.15}$$

conservation of momentum is expressed by equation (2.10)

$$\rho \frac{Du_i}{Dt} = \rho f_i - \frac{\partial}{\partial x_i}\left(p - \mu^{'}\frac{\partial u_k}{\partial x_k}\right) + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right], \tag{2.16}$$

and conservation energy is expressed by equation (2.14),

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho u E) - \rho u_i f_i - \frac{\partial}{\partial x_j}\left(u_i p_{ij} + k\frac{\partial T}{\partial x_j}\right) = 0. \tag{2.17}$$

No slip boundary condition is satisfied for these equations on the solid body, that is,

$$u_i = 0 \text{ on } \partial\Omega.$$

2.1.2 The Euler Equations for Compressible Inviscid Flow

In case of uniform flow, the constitutive relation of stress tensor equation (2.9) reduces to:

$$p_{ij} = -p\delta_{ij}. \tag{2.18}$$

Then the governing equations of motion of compressible, inviscid flow take the following form. Conservation of mass is expressed by equation

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho u) = 0, \tag{2.19}$$

conservation of momentum is expressed by equation

$$\rho \frac{Du_i}{Dt} = \rho f_i - \frac{\partial p}{\partial x_i}, \tag{2.20}$$

and conservation energy is expressed by equation,

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho u E) - \rho u_i f_i - \frac{\partial}{\partial x_j}\left(k \frac{\partial T}{\partial x_j}\right) = 0. \tag{2.21}$$

The boundary condition for these equations requires that normal component of the velocity is zero on the solid body, that is,

$$u \cdot n = 0 \text{ on } \partial\Omega.$$

2.1.3 Vector Form of the Navier-Stokes Equations for Compressible Viscous Flow

For the derivations that follow, it is convenient to use Cartesian coordinates (x_1,x_2,x_3) and to adopt the convention of indicial notation where a repeated index "i" implies summation over $i = 1$ to 3. The three-dimensional Navier-Stokes equations then take the form

$$\frac{\partial w}{\partial t} + \frac{\partial f_i}{\partial x_i} = \frac{\partial f_{vi}}{\partial x_i} \quad \text{in } \Omega, \tag{2.22}$$

where the state vector w, inviscid flux vector f and viscous flux vector f_v are described respectively by

$$w = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{bmatrix}, \quad f_i = \begin{bmatrix} \rho u_i \\ \rho u_i u_1 + p\delta_{i1} \\ \rho u_i u_2 + p\delta_{i2} \\ \rho u_i u_3 + p\delta_{i3} \\ \rho u_i H \end{bmatrix}, \quad f_{vi} = \begin{bmatrix} 0 \\ \tau_{ij}\delta_{j1} \\ \tau_{ij}\delta_{j2} \\ \tau_{ij}\delta_{j3} \\ u_j \tau_{ij} + k\frac{\partial T}{\partial x_i} \end{bmatrix}. \tag{2.23}$$

The viscous stresses may be written as

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial u_k}{\partial x_k}. \tag{2.24}$$

The coefficient of thermal conductivity and the temperature are computed as

$$k = \frac{c_p \mu}{Pr}, \qquad T = \frac{p}{R\rho}, \tag{2.25}$$

where Pr is the Prandtl number, c_p is the specific heat at constant pressure, and R is the gas constant.

2.2 Non-dimensionalization

We discuss here some scaling properties of the Navier-Stokes equations in order to introduce some non-dimensional parameters measuring similarity of flow problems. For a given problem, suppose L be a *characteristic length* and U_∞ be the free-stream velocity. We introduce the following dimensionless quantities

$$x' = \frac{x}{L},\ t' = \frac{t}{L/U_\infty},\ w' = \frac{w}{U_\infty},\ \mu' = \frac{\mu}{\mu_\infty},\ \rho' = \frac{\rho}{\rho_\infty},\ p' = \frac{p}{\rho_\infty U_\infty^2},\ T' = \frac{T}{T_\infty},\ e' = \frac{e}{U_\infty^2},$$

where the quantities with subscript ∞ denote the values at free-stream conditions and the Reynolds number is defined as

$$Re = \frac{\rho_\infty U_\infty L}{\mu_\infty}.$$

If this nondimensionalizing procedure applied to the compressible Navier-Stokes equations, the following non-dimensional equations are obtained

$$\frac{\partial w'}{\partial t'} + \frac{\partial f_i'}{\partial x_i'} = \frac{\partial f_{vi}'}{\partial x_i'} \quad \text{in } \Omega, \tag{2.26}$$

where the state vector w, inviscid flux vector f and viscous flux vector f_v are described respectively by

$$w' = \begin{bmatrix} \rho' \\ \rho' u_1' \\ \rho' u_2' \\ \rho' u_3' \\ \rho' E' \end{bmatrix}, \quad f_i' = \begin{bmatrix} \rho' u_i' \\ \rho' u_i' u_1' + p' \delta_{i1} \\ \rho' u_i' u_2' + p' \delta_{i2} \\ \rho' u_i' u_3' + p' \delta_{i3} \\ \rho' u_i' H' \end{bmatrix}, \quad f_{vi}' = \begin{bmatrix} 0 \\ \tau_{ij}' \delta_{j1} \\ \tau_{ij}' \delta_{j2} \\ \tau_{ij}' \delta_{j3} \\ u_j' \tau_{ij}' + k' \frac{\partial T'}{\partial x_i'} \end{bmatrix}, \tag{2.27}$$

with

$$\tau_{ij}' = \frac{\mu'}{Re} \left[\left(\frac{\partial u_i'}{\partial x_j'} + \frac{\partial u_j'}{\partial x_i'} \right) - \frac{2}{3} \delta_{ij} \frac{\partial u_k'}{\partial x_k'} \right]. \tag{2.28}$$

2.3 Turbulence and Its Modeling

The Reynolds number (as defined above) of a flow is the measure of the relative importance of inertia forces (associated with convective effects) and viscous forces.

It is observed from experiments that at values below the so-called critical Reynolds number Re_{crit}, the flow is smooth and adjacent layers of the fluid slide past each other in an orderly fashion. This steady flow regime (unless the applied boundary conditions change with time) is called **laminar**. At values of Reynolds number above Re_{crit} a complicated series of events takes place eventually leading to a radical change of the flow character. In the final state the flow behavior is random and chaotic. This flow regime is called **turbulent**.

In visualisation of turbulent flow, appearance of eddying motions of wide range of length scales are important features. Whereas no such motion is present in laminar flow. It is reported that in a typical flow domain of 0.1 m by 0.1 m with a high Reynolds number turbulent flow might contain eddies down to 10 to 100 μm size. One would need computing meshes of 10^9 to 10^{12} points to be able to describe processes at all length scales. The fastest events take place with a frequency on the order of 10 kHz so one would need to discretize time into steps of about 100 μs.

Due to these difficulties it has only recently started to become possible to track the dynamics of eddies in very simple flows at transitional Reynolds numbers with present day computing power. The direct simulation of the time-dependent Navier-Stokes equations of fully turbulent flows at high Reynolds numbers are truly phenomenal and must await major developments in computer technology. However, scientists and engineers need some procedure which can supply adequate information about turbulent process without predicting the effect of each and every eddy in the flow. Such informations are satisfactorily described by the time-averaged properties of the flow (e.g., mean velocities, mean pressure, mean stresses etc.).

2.3.1 Turbulent Averaged Quantities

The turbulent averaging process is introduced in order to obtain the laws of motion for the 'mean', time-averaged, turbulent quantities. This time averaging is to be defined in such a way as to remove the influence of the turbulent fluctuations while not destroying the time dependence associated with other time-dependent phenomena with time scales distinct from those of turbulence. The mean $\Phi(t,x)$ of a flow property $\phi(t,\tau,x)$ is defined as

$$\Phi(t,x) = \frac{1}{\Delta\tau}\int_0^{\Delta\tau} \phi(t,\tau,x)d\tau. \tag{2.29}$$

$\Delta\tau$ is to be chosen large enough compared to the same time scale of the turbulence but still small enough compared to all other unsteady phenomena.

The flow property ϕ is time dependent and can be thought of as the sum of a steady mean component Φ and a time-varying fluctuating component ϕ' with zero mean value; hence $\phi(t,\tau,x) = \Phi(t,x) + \phi'(\tau)$. For the sake of convenience, we will write it as $\phi = \Phi + \phi'$ avoiding the time dependence explicitly. The time average of the fluctuations ϕ' is, by definition, zero:

$$\overline{\phi'} = \frac{1}{\Delta\tau}\int_0^{\Delta\tau} \phi'(\tau)d\tau = 0. \tag{2.30}$$

For compressible flows the *density-weighted* averaging is introduced, in order to avoid the explicit occurrence of products of fluctuations between density and other variables, as follows

$$\tilde{\Phi} = \frac{\overline{\rho \Phi}}{\overline{\rho}}$$

with

$$\phi = \tilde{\Phi} + \phi''$$

and

$$\overline{\rho \phi''} = 0.$$

Information regarding the fluctuating part of the flow can, for example, be obtained from the root-mean-square (rms) of the fluctuations.

2.3.2 The Reynolds Averaged Navier-Stokes Equations

Performing the averaging process mentioned above on the continuity equation will lead to

$$\frac{\partial}{\partial t}\overline{\rho} + \nabla \cdot (\overline{\rho}\tilde{u}) = 0. \tag{2.31}$$

The averaged momentum equations lead to introduction of the Reynolds stress tensor. These equations, in the absence of body forces, in averaged quantities read as

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{u}) + \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_i\tilde{u}_j) = -\frac{\partial \overline{p}}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\overline{\tau}_{ij} - \overline{\rho u_i'' u_j''}\right), \tag{2.32}$$

where, neglecting viscosity fluctuations, $\overline{\tau}_{ij}$ becomes

$$\overline{\tau}_{ij} = \mu\left[\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial \tilde{u}_k}{\partial x_k}\right] + \mu\left[\left(\frac{\partial \overline{u_i''}}{\partial x_j} + \frac{\partial \overline{u_j''}}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial \overline{u_k''}}{\partial x_k}\right] \tag{2.33}$$

and $\overline{\overline{\tau}}^R$ are the Reynolds stresses, defined by

$$\overline{\overline{\tau}}^R = -\overline{\rho u_i'' u_j''}.$$

Here, for density and pressure the time averaging is used whereas for velocities the density-weighted averaging is used.

For the averaged energy equation, one needs to make distinction between the averaged total energy $\tilde{E}$ and the total energy of the averaged flow $\hat{E}$ which differ by the kinetic energy of the turbulent fluctuations. If we define the *mean turbulent total energy* by

$$\overline{\rho \tilde{E}} = \overline{\rho E} = \overline{\rho\,(e + u^2/2)},$$

we obtain

$$\tilde{E} = \tilde{e} + \tilde{k} + k = \hat{E} + k,$$

where $\tilde{k}$ is the kinetic energy of the mean flow per unit mass,

$$\overline{\rho}\tilde{k} = \overline{\rho}\tilde{u}^2/2,$$

and k is the turbulent kinetic energy; thus

$$\overline{\rho}k = \overline{\rho u''/2} \equiv \overline{\rho k''},$$

is defined as the average of the kinetic energy k'' of the turbulent fluctuations.

Similarly, *the averaged total enthalpy* is defined by

$$\tilde{H} = \tilde{E} + \bar{p}/\bar{\rho} = \tilde{h} + \tilde{k} + k \equiv \hat{H} + k,$$

where $\hat{H}$ is the stagnation enthalpy of the averaged flow.

The fluctuating components are given by

$$H'' = h'' + u'' \cdot \tilde{u} + k'' - k,$$

and a similar relation for the fluctuating total energy E'' is

$$E'' = e'' + u'' \cdot \tilde{u} + k'' - k.$$

Using these relations, the averaged energy equation reads as

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{H}) + \frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u}_j\tilde{H} + \overline{\rho u_j''H''} - k\frac{\partial \overline{T}}{\partial x_j}\right) = -\frac{\partial \overline{p}}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\tilde{u}_i\overline{\tau}_{ij} - \overline{u_i''\tau_{ij}}\right). \quad (2.34)$$

As a consequence of such Reynolds decomposition and averaging, extra terms such as Reynolds stresses and heat flux vectors are introduced in the equations. To form a closed system, some semi-empirical relationships are necessary, leading to so-called turbulence models. Many turbulence models have been proposed ranging from zero-equation models to two-equation models. In zero-equation models, also known as algebraic models, Boussinesq's closure hypothesis is used, with an algebraic equation to relate the eddy viscosity to the primary unknowns. In our applications, we used algebraic turbulence model of Baldwin and Lomax [7]. A survey of turbulence modeling can be found in [119, 164] for details.

2.4 Analytic Aspects of the PDEs

Computational Fluid Dynamics (CFD) plays a crucial role in solving the Euler or Navier-Stokes equations in arbitrary domain. Knowledge of the numerical analysis of PDEs is indispensable in CFD. Classification of the PDEs is important for the numerical analysis. Depending on the type of PDEs, they require different boundary conditions and different numerical methods. Also, the classification corresponds to

different types of qualitative behavior of solutions, and hence to difference in the underlying physics.

In general, the PDEs can be of elliptic, hyperbolic or parabolic type in a domain of definition. The type of equation may change in the domain, for example, in transonic flow. Also, the type may not be defined for some PDEs. There the difficulty for numerical treatment is greater. In case of high-speed flows, the Navier-Stokes equations have dominating convective character. Hence, they are to be treated as hyperbolic equations similar to those of the Euler equations. Quite often, solution of these equations involve discontinuities, such as shock or contact discontinuities. Therefore, concept of weak solution is important in these cases, since strong solution does not exist. Details of numerical treatment of these equations can be found in [3, 116, 163, 88, 18, 42, 81, 130]. We will present the conservative formulation of the equations in order to give brief description of the finite volume discretization of the equations in respective chapters.

Chapter 3
PDE-Constrained Optimization Methods

We present here few definitions, theorems without proof and a brief description of the numerical methods for nonlinear optimization problems. Detailed description of the methods can be found in [132, 43, 21, 51, 160, 159, 80]. These methods are usually iterative in nature. Mainly two different types of methods are found in the literature, namely, gradient based methods and gradientless or heuristic methods (such as genetic algorithm, simulated annealing methods etc.). Gradient based methods are usually fast and obtains only a local minimizer. In contrast to this, heuristic methods are very slow, requires a large number of evaluations, but they lead to a global minimizer. Since, we are concerned with fast algorithms for practical applications, we use gradient based methods in this book.

3.1 Unconstrained Optimization Problem

An unconstrained optimization problem can be formulated mathematically as

$$\min_q f(q),$$

where $q \in \mathbb{R}^n$ is a real vector and $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a smooth function.

Definition 4. A point q^* is a global minimizer if $f(q^*) \leq f(q)$ for all $q \in \mathbb{R}^n$.

Definition 5. A point q^* is a local minimizer if there is a neighborhood N of q^* such that $f(q^*) \leq f(q)$ for all $q \in$ N.

Definition 6. A point q^* is a strict local minimizer if there is a neighborhood N of q^* such that $f(q^*) < f(q)$ for all $q \in$ N with $q \neq q^*$.

Theorem 4. *(***Taylor's Theorem***):* *Suppose that $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable and that $p \in \mathbb{R}^n$. Then we have that*

$$f(q+p) = f(q) + \nabla f(q+ep)^\top p$$

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 19–25.
springerlink.com

for some $e \in (0,1)$. *Moreover, if* f *is twice continuously differentiable, we have that*

$$\nabla f(q+p) = \nabla f(q) + \int_0^1 \nabla^2 f(q+ep)pde,$$

and that

$$f(q+p) = f(q) + \nabla f(q)^\top p + \frac{1}{2} p^\top \nabla^2 f(q+pe)p,$$

for some $e \in (0,1)$.

Theorem 5. *(***First-order necessary conditions***):* *If* q^* *is a local minimizer and* f *is continuously differentiable in an open neighborhood of* q^*, *then* $\nabla f(q^*) = 0$.

Theorem 6. *(***Second-order necessary conditions***):* *If* q^* *is a local minimizer of* f *and* $\nabla^2 f$ *is continuous in an open neighborhood of* q^*, *then* $\nabla f(q^*) = 0$ *and* $\nabla^2 f(q^*)$ *is positive semidefinite.*

Theorem 7. *(***Second-order sufficient conditions***):* *Suppose that* $\nabla^2 f$ *is continuous in an open neighborhood of* q^* *and that* $\nabla f(q^*) = 0$ *and* $\nabla^2 f(q^*)$ *is positive definite. Then* q^* *is a strict local minimizer of* f.

Theorem 8. *When* f *is convex, any local minimizer* q^* *is a global minimizer of* f. *If in addition* f *is differentiable, then any stationary point* q^* *is a global minimizer of* f.

Gradient based methods iteratively search for a minimum of f starting with an initial guess q_0. There are two fundamental strategies to move the iterate q_k to q_{k+1}, namely, line search and trust region.

Line Search: First determine a descent direction d_k such that

$$d_k^\top \nabla f(q_k) < 0.$$

Then determine a step α_k such that $q_{k+1} = q_k + \alpha d_k$ which approximately solves

$$\min_\alpha f(q_k + \alpha d_k).$$

Trust Region: Determine the direction d and control its size simultaneously by solving

$$\min_d M_k(q_k + d) \text{ s.t. } ||d|| \leq \Delta,$$

where Δ is the trust region radius. Usually the approximation function M_k is given by

$$M_k(q_k + d) = f_k + d^\top \nabla f_k + \frac{1}{2} d^\top B_k d,$$

where B_k is the Hessian of f or some approximation to it.

Based on different choices of the direction d_k, following methods are formulated:

Steepest descent method: If $d_k = -\nabla f_k$, then the method is called steepest descent. This method is linearly convergent.
Newton's method: If $d_k = -[\nabla^2 f_f]^{-1}\nabla f_k$, then the method is called Newton method. This method is quadratically convergent.
Quasi-Newton Methods: If $d_k = -B_k^{-1}\nabla f_k$, where B_k is some approximation of the Hessian matrix $\nabla^2 f_f$, then the method is called quasi Newton method. This method is super-linearly convergent.

3.2 Constrained Optimization Problem

These problems can be written mathematically as

$$\begin{aligned} &\min_{w,q} f(w,q) \\ &\text{s. t.} \quad \mathbf{c}(w,q) = 0, \end{aligned} \tag{3.1}$$

If second order derivatives of the objective function and the constraints are available, one can use Newton's method to solve the necessary optimality conditions in equations (3.3). where $w \in \mathbb{R}^{n_w}$, $q \in \mathbb{R}^{n_q}$, $f : \mathbb{R}^{(n_w+n_q)} \mapsto \mathbb{R}$ is the objective function and $\mathbf{c} : \mathbb{R}^{(n_w+n_q)} \mapsto \mathbb{R}$ is the set of nonlinear simulation equations or constraints.

The Lagrangian functional is defined as

$$L(w,q,\lambda) = f(w,q) - \lambda^\top \mathbf{c}(w,q), \tag{3.2}$$

where $\lambda \in \mathbb{R}^{n_w}$ is the vector of Lagrange multipliers or the adjoint variables. The necessary optimality conditions will lead to the system of equations:

$$\frac{\partial L}{\partial w} = \frac{\partial f}{\partial w} - \lambda^\top \frac{\partial \mathbf{c}}{\partial w} = 0, \tag{3.3a}$$

$$\frac{\partial L}{\partial q} = \frac{\partial f}{\partial q} - \lambda^\top \frac{\partial \mathbf{c}}{\partial q} = 0, \tag{3.3b}$$

$$\mathbf{c}(w,q) = 0. \tag{3.3c}$$

3.2.1 Nested Analysis and Design (NAND)

This approach is also known as so-called 'black-box' approach. In this approach the state variables are considered as implicit function of the design variables, i.e., $w = w(q)$ and for a given set of design variables the constraints $\mathbf{c}(w(q),q) = 0$ can be solved for the states w. These are then substituted in the objective function so that the constrained problem reduces to the unconstrained one

$$\min_q \hat{f}(q) = f(w(q),q). \tag{3.4}$$

The above mentioned algorithms for unconstrained problem can be used to solve this problem. This approach is also known as 'nested analysis and design' (NAND) method in the literature. The disadvantage of this method is that in each optimization iteration the constraint equations $\mathbf{c}(w(q),q) = 0$ are to be solved quite accurately in order to get the state variables $w(q)$. This leads to huge computational cost of the method.

In most of the practical applications, $\mathbf{c}(w(q),q) = 0$ is a system of nonlinear partial differential equations together with the initial and/or the boundary conditions. Usually they are solved using a numerical (discretization) method coded in the PDE-solver. The numerical method is an approximation or linearization of the nonlinear equations which can be presented mathematically using Taylor expansion around (w_0,q_0) as

$$\mathbf{c}(w,q) = \mathbf{c}(w_0,q_0) + \frac{\partial \mathbf{c}}{\partial w}\delta w + \frac{\partial \mathbf{c}}{\partial q}\delta q + \circ(||\delta w||^2) + \circ(||\delta q||^2),$$

where $\frac{\partial \mathbf{c}}{\partial w}$ is a square $(n_w \times n_w)$ Jacobian matrix evaluated at (w_0,q_0) and $\frac{\partial \mathbf{c}}{\partial q}$ is a rectangular $(n_w \times n_q)$ matrix evaluated at (w_0,q_0). If the residual do not change (i.e., $\mathbf{c}(w,q) = \mathbf{c}(w_0,q_0)$) then for sufficiently small δw and δq, ignoring the higher order terms, one gets

$$\frac{\partial \mathbf{c}}{\partial w}\delta w + \frac{\partial \mathbf{c}}{\partial q}\delta q = 0.$$

If the Jacobian $\mathbf{J} = \dfrac{\partial \mathbf{c}}{\partial w}$ is nonsingular then this gives the increments of the state variables as

$$\delta w = -(\frac{\partial \mathbf{c}}{\partial w})^{-1}\frac{\partial \mathbf{c}}{\partial q}\delta q.$$

The matrix

$$\frac{\partial w}{\partial q} = -(\frac{\partial \mathbf{c}}{\partial w})^{-1}\frac{\partial \mathbf{c}}{\partial q} \tag{3.5}$$

represents the sensitivity of w with respect to q and is known as *direct sensitivity matrix*.

Hence, the PDE-solver gives the state variables and nonlinear elimination of these variables from the objective function reduces the problem (3.1) to the problem (3.4) in which $\hat{f}(q) = f(w(q),q)$ is known as *reduced objective function*. If one uses the aforementioned gradient methods to solve this unconstrained problem, then one needs the *reduced gradient*, $\dfrac{\partial \hat{f}}{\partial q}$, of the reduced objective function with respect to the parameters or design variables q.

Computation of the reduced gradients: There are various methods to compute the reduced gradient. The simplest and straightforward method is the finite-difference method. In this method a small perturbation is introduced in, say, i-th parameter whose effect in the constraint as well as in the objective function is determined and used to find the variation as

$$\left(\frac{\partial \hat{f}}{\partial q}\right)_i \approx \frac{f(w(q_i+\varepsilon e_i), q_i+\varepsilon e_i) - f(w(q_i), q_i)}{\varepsilon},\ i = 1, 2, \cdots, n_q,$$

where n_q is the dimension of q. As the formula shows, this method needs $(n_q + 1)$ number of PDE-solves per optimization iteration for computation of gradients. Another difficulty is the choice of the perturbation parameter ε which affects the accuracy of the computed reduced gradient.

An alternative to this is to use exact differentiation (chain rule) of the actual objective function

$$\frac{\partial \hat{f}}{\partial q} = \frac{\partial f}{\partial w}\frac{\partial w}{\partial q} + \frac{\partial f}{\partial q}.$$

Using the formula of direct sensitivity matrix from equation (3.5), one gets

$$\frac{\partial \hat{f}}{\partial q} = -\frac{\partial f}{\partial w}\left(\frac{\partial \mathbf{c}}{\partial w}\right)^{-1}\frac{\partial \mathbf{c}}{\partial q} + \frac{\partial f}{\partial q}. \tag{3.6}$$

The first term in the right hand side of equation (3.6) can be computed following two approaches namely *direct sensitivity approach* or *adjoint sensitivity approach*. In the direct sensitivity approach the direct sensitivity matrix $\frac{\partial w}{\partial q} = -\left(\frac{\partial \mathbf{c}}{\partial w}\right)^{-1}\frac{\partial \mathbf{c}}{\partial q}$ is computed first and then the product $\frac{\partial f}{\partial w}\frac{\partial w}{\partial q}$ is computed. This is advantageous specially for those problems in which the analysis or PDE-solver uses Newton-type methods. Since the Jacobian $\frac{\partial \mathbf{c}}{\partial w}$ is already setup to solve the linear system in Newton-type methods. The sensitivity matrix $\frac{\partial w}{\partial q}$ is formed by n_q number of linear system solve with the Jacobian matrix $\frac{\partial \mathbf{c}}{\partial w}$ and each column of $\frac{\partial \mathbf{c}}{\partial q}$ in the right-hand side. This is generally a great improvement over the finite-difference methods as n_q linear system solve is much cheaper than a full simulation to evaluate $w(q)$ and the resulting reduced gradient is much more accurate. This approach is used for computation of reduced gradients in our applications mentioned in Chapter 5.

In the second approach, adjoint sensitivity approach, of computing the first term in the right hand side of equation (3.6) the adjoint variables (also known as Lagrange multipliers)

$$\lambda = \left(\frac{\partial \mathbf{c}}{\partial w}\right)^{-\top}\frac{\partial f}{\partial w}^{\top}, \tag{3.7}$$

are computed first and then the product $\lambda^{\top}\frac{\partial \mathbf{c}}{\partial q}$. In PDE-constrained optimization, the system of equations (3.7) is usually a linear system of equations for λ involving transposed Jacobian matrix $\frac{\partial \mathbf{c}}{\partial w}^{\top}$ and only a singe solve of this equation required to compute the exact reduced gradient. This approach is specially advantageous for the problems in which the analysis code does not use Newton-type methods.

This approach is used for computation of reduced gradients in our applications to aerodynamic shape optimization problems in Chapters 6-12.

Algorithm 1: *Outline of NAND Algorithms for Unconstrained Optimization*
(0) Set $k := 0$*; start at some initial guess* q_0*.*
(1) Compute the reduced gradient $\frac{\partial \hat{f}}{\partial q}$ *at* $w = w(q_k)$*.*
(2) Compute the increment δq *such that* $\frac{\partial \hat{f}}{\partial q}\delta q < 0$ *.*
(3) Find a step length, say, α *that ensures improvement in the solution.*
(4) Compute $q^{k+1} = q^k + \alpha\delta q$*.*
(5) $k := k+1$*; go to (1) until convergence.*

3.2.2 Simultaneous Analysis and Design (SAND)

The other approach, known as 'simultaneous analysis and design' (SAND), uses simultaneous iteration for the constraints (simulation or analysis problem) and the objective function (design problem). These methods also require a PDE-solver as well as the reduced gradients. The PDE-solver does not require to deliver full converged solution of the constraints in each optimization iteration, only a partial convergence upto 'acceptable' tolerance is required. This leads to possible reduction of the computational cost of the method. The reduced gradients can be computed using any of the methods discussed in the previous section.

Algorithm 2: *Outline of SAND Algorithms for constrained Optimization*
(0) Set $k := 0$*; start at some initial guess* w_0, q_0*.*
(1) Compute the reduced gradient $\frac{\partial \hat{f}}{\partial q}$ *and the residual of* $\mathbf{c}(w_k, q_k)$*.*
(2) Compute the increment $\delta w = \left(\frac{\partial \mathbf{c}}{\partial w}\right)^{-1}\mathbf{c}$ *and* δq *such that* $\frac{\partial \hat{f}}{\partial q}\delta q < 0$ *.*
(3) Find a step length, say, α *that ensures improvement in the solution.*
(4) Compute $w^{k+1} = w^k + \alpha(\delta w_k + \frac{\partial w}{\partial q}\delta q)$ *and* $q^{k+1} = q^k + \alpha\delta q$*.*
(5) $k := k+1$*; go to (1) until convergence.*

3.2.3 Full Newton SAND

If second order derivatives of the objective function and the constraints are available, one can use Newton's method to solve the necessary optimality conditions in equations (3.3). This will lead to solving the following linear system (known as KKT system):

$$\begin{pmatrix} \frac{\partial^2 L}{\partial w^2} & \frac{\partial^2 L}{\partial w \partial q}^\top & \frac{\partial \mathbf{c}}{\partial w}^\top \\ \frac{\partial^2 L}{\partial w \partial q} & \frac{\partial^2 L}{\partial q^2} & \left(\frac{\partial \mathbf{c}}{\partial q}\right)^\top \\ \frac{\partial \mathbf{c}}{\partial w} & \frac{\partial \mathbf{c}}{\partial q} & 0 \end{pmatrix} \begin{pmatrix} \delta w \\ \delta q \\ \delta \lambda \end{pmatrix} = \begin{pmatrix} -\frac{\partial L}{\partial w}^\top \\ -\frac{\partial L}{\partial q}^\top \\ -\mathbf{c} \end{pmatrix}. \tag{3.8}$$

Algorithm 3: *Outline of Full Newton SAND Algorithms for constrained Optimization*
(0) Set $k := 0$*; start at some initial guess* w_0, q_0, λ_0.
(1) Compute the first and second order sensitivities and the residual of $\mathbf{c}(w_k, q_k)$.
(2) Compute the increment $\delta w, \delta q, \delta \lambda$ *solving the KKT system.*
(3) Find a step length, say, α *that ensures improvement in the solution.*
(4) Compute $w^{k+1} = w^k + \alpha \delta w_k$, $q^{k+1} = q^k + \alpha \delta q$ *and* $\lambda^{k+1} = \lambda^k + \alpha \delta \lambda$.
(5) $k := k+1$*; go to (1) until convergence.*

In case the second order sensitivities are expansive to compute, some update formulas are used as an approximation and this leads to different variants of Newton's method.

In our applications of aerodynamic shape optimization problems, the governing PDEs are steady state. However, the solution approach does not use matrix-based iterative method, but it uses pseudo-time-stepping. Hence, we apply a method which is based on simultaneous approach and motivated by the rSQP methods. We will discuss that in detail in Chapter 6.

Chapter 4
Mathematical Model of Multiphase Flow through Porous Media

4.1 Introduction

In general, multiphase flows concern the flow of two or more immiscible fluids (phases) in a porous media. These are of immense practical relevance to subsurface contamination and remediation techniques. A distinguishing feature of multiphase flow, in comparison to single phase flow, is the existence of interfaces between fluids. At the microscopic (pore) scale, these interfaces are known to influence the system behavior by supporting non-zero stresses such that the pressures at the adjacent phases are not equal. Thus, to make a reliable mathematical model, it is necessary to identify and understand physical processes at microscopic scale and to describe their manifestation at the macroscopic (core) or field scale. The connection of the flow physics between these two scales can be understood by so-called upscalling. In the following the mathematical model is presented most part of which can also be found in [73, 72, 74].[1]

The mathematical formulation of multiphase flow and transport processes in a porous medium require a system of equations that is capable to describe the relevant physical processes appropriately. Depending on the problem, as a preliminary step, one has to build up a conceptual model which must reproduce the essential characteristic properties of the system behavior. A major distinction is made between multiphase systems and multiphase multicomponent systems. In case of a multiphase multicomponent system, the phases are composed of several components and the components may exchange from one phase into another. Such mass transfer processes are, for example, evaporation, condensation, dissolution, and degassing. These are coupled with an exchange of thermal energy between the phases. Hence, an energy balance is necessary in order to take that into account. As described in Chapter 2, the mathematical model describing a flow phenomena is given by balance equations for mass, momentum and energy together with system dependent

[1] Reprinted with kind permissions of Springer-Verlag and Springer Science and Business Media.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 29–41.
springerlink.com

equations of state. Most flow and transport processes in porous media are very slow processes [13] with very small Reynolds number (< 1) and with sufficiently large Knudsen numbers (> 10) such that the momentum equations lead to generalized form of Darcy's law. Thus, current description of macroscopic multiphase flow behavior is based on empirical extension of Darcy's law supplemented with capillary pressure-saturation-relative permeability relationships.

In the following, we will first present the general form of the multiphase flow differential equations. On the basis of that, we describe the equations and properties of an isothermal two-phase system and afterwards extend this to a nonisothermal two-phase two-component water–gas model concept. Then, we introduce the constitutive relationships and closure relations.

4.2 General form of the Multiphase Flow Equations

In the Eulerian approach, the continuity equation, presented in Chapter 2, for multiphase flow in porous media for a phase α is given by:

$$\int_G \left[\frac{\partial(\phi S_\alpha \rho_\alpha)}{\partial t} + \nabla \cdot (\rho_\alpha \mathbf{v}_\alpha) \right] dG = 0. \tag{4.1}$$

Here, ϕ stands for the porosity, S_α for the saturation, and ρ_α for the mass density, where the index α identifies the respective fluid phase. $\mathbf{v}_\alpha$ is the flow velocity of phase α averaged over the cross section of the porous medium. Note that this is not the mean velocity $\mathbf{v}_{a,\alpha}$ of the water molecules, since the latter is related to the Darcy velocity by

$$\mathbf{v}_\alpha = \mathbf{v}_{a,\alpha} \phi \ . \tag{4.2}$$

Darcy's Law for single phase flow is

$$\mathbf{v} = -\frac{\mathbf{K}}{\mu} \cdot (\nabla p - \rho \mathbf{g}) , \tag{4.3}$$

where p is the phase pressure, $\mathbf{K}$ the intrinsic permeability tensor of the porous medium, μ the dynamic viscosity of the fluid, and $\mathbf{g}$ the vector of gravitational acceleration. For a multiphase system, the Darcy's Law (Eq. (4.3)) can be extended by considering the relative permeability $k_{r\alpha}$ of the phases, see, for example, in [75], such that the Darcy velocity for a phase α is obtained from

$$\mathbf{v}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} \mathbf{K} \cdot (\nabla p_\alpha - \rho_\alpha \mathbf{g}). \tag{4.4}$$

The term $\frac{k_{r\alpha}}{\mu_\alpha}$ is commonly called the mobility λ_α of phase α.

By using Darcy's Law as a reduced form of the momentum equation in porous media, it is possible to decouple the calculation of the phase velocities from the continuity equation. Inserting Eq. (4.4) into Eq. (4.1), adding a source/sink term q_α for

phase α, and writing in differential form yields the general form of the multiphase flow equation:

$$\frac{\partial(\phi S_\alpha \rho_\alpha)}{\partial t} - \nabla \cdot (\rho_\alpha \lambda_\alpha \mathbf{K} \cdot (\nabla p_\alpha - \rho_\alpha \mathbf{g})) - \rho_\alpha q_\alpha = 0. \tag{4.5}$$

4.2.1 Isothermal Water-Gas System (Two-Phase Flow)

Now, we consider a two-phase system. The fluid phase, which has the higher affinity to the porous medium, is the wetting phase w, the second phase is the non-wetting phase n. We always have water as the wetting and gas as the non-wetting phase. Then, Eq. (4.5) represents a system of two coupled differential equations, which is completed by the relation

$$\sum_\alpha S_\alpha = 1 \,, \tag{4.6}$$

the relation of the phase pressures via the capillary pressure p_c

$$p_c = p_n - p_w = f(S) \,, \tag{4.7}$$

and additional state relations for the density $\rho(p)$, for example Ideal Gas Law, viscosity $\mu(p)$ and relative permeability $k_r(S)$. The system exhibits a high degree of nonlinearity, mainly caused by the nonlinear dependence of the capillary pressure and the relative permeability on the saturation. This is reinforced by a strong variation of these constitutive relationships due to heterogeneities.

Pressure–saturation formulation

The choice of the primary variables can be made in different ways. The formulation we use here is the pressure–saturation formulation. The two unknowns are the pressure of the wetting phase, p_w, and the saturation of the non-wetting phase, S_n, respectively or vice versa. The following reformulations should be made for the following terms in Eq. (4.5):

$$\nabla p_n = \nabla(p_w + p_c) \tag{4.8}$$

$$\frac{\partial S_w}{\partial t} = \frac{\partial}{\partial t}(1 - S_n) = -\frac{\partial S_n}{\partial t}. \tag{4.9}$$

Then, we get for the wetting phase (water)

$$-\phi \rho_w \frac{\partial S_n}{\partial t} - \nabla \cdot (\rho_w \lambda_w \mathbf{K} \cdot (\nabla p_w - \rho_w \mathbf{g})) - \rho_w q_w = 0, \tag{4.10}$$

and for the non-wetting phase (gas or NAPL)

$$\phi \frac{\partial(\rho_n S_n)}{\partial t} - \nabla \cdot (\rho_n \lambda_n \mathbf{K} \cdot (\nabla p_w + \nabla p_c - \rho_n \mathbf{g})) - \rho_n q_n = 0 \,. \tag{4.11}$$

Note that we take the porosity ϕ out of the time derivative term since we assume it to be constant. We do the same with the wetting phase density ρ_w due to the assumption that water being incompressible. If the non-wetting phase is a gaseous phase, we have to consider a density varying with pressure.

Some alternatives exist to the pressure–saturation formulation, for an overview see e.g. [75]. One of them is the pressure formulation with both phase pressures as unknowns. This formulation takes advantage of the monotonic behavior of the capillary pressure as a function of saturation (Eq. (4.7)), which is the necessary condition for existence of an inverse function $S = g(p_c)$. However, the disadvantage of the pressure formulation appears when the gradient of the capillary pressure becomes small, i.e., $\frac{dp_c}{dS} \approx 0$. This case normally occurs for high water saturations. For two incompressible fluids, it is also possible to use the fractional flow formulation [75].

4.2.2 Nonisothermal Water-Gas Systems (Two-Phase Two-Component Flow)

In the following we extend the isothermal two-phase flow model to a nonisothermal water-gas system containing the phases water and gas (phase $\alpha \in \{w, g\}$)[2] as well as the components water and air[3] (denoted by the superscripts *wa* and *ai* respectively, component $K \in \{wa, ai\}$). A more detailed presentation of the nonisothermal model concept implemented in the program system MUFTE_UG [77] is given by Class et al. (2002) [30] and Class (2001) [29].

The system of equations include two mass balances, one for each component, and a single energy balance. Note that we assume local thermal equilibrium. Chemical or biological effects are not considered. The pressure and temperature ranges for which the model concept is designed are $\approx 1 \cdots 5$ bar and $\approx 0 \cdots 200$ °C respectively.

We formulate the balance equations for each mass component by multiplying the terms in Eq. (4.5) with the corresponding mole fractions of the components in the phases and then summing up over the phases. We additionally consider a diffusive flux term in the gas phase. Furthermore, the balance equations are molar, which is why we distinguish between the molar density $\rho_{mol,\alpha}$ and the mass density $\rho_{mass,\alpha}$ of a phase α. Hence we get

Mass balance

$$\phi \frac{\partial (\sum_\alpha \rho_{mol,\alpha} x_\alpha^K S_\alpha)}{\partial t} - \sum_\alpha \nabla \cdot \left\{ \frac{k_{r\alpha}}{\mu_\alpha} \rho_{mol,\alpha} x_\alpha^K \mathbf{K} (\nabla p_\alpha - \rho_{mass,\alpha} \mathbf{g}) \right\}$$

$$- \nabla \cdot \left\{ D_{pm} \rho_{mol,g} \nabla x_g^K \right\} - q^K = 0, \qquad K \in \{wa, ai\},\ \alpha \in \{w, g\}. \tag{4.12}$$

[2] We present by subscript w both the meanings *water* and *wetting phase*, as in our context the wetting phase is always water.

[3] We are well aware that air consists of several components. However, we neglect this for the sake of simplicity.

The diffusion coefficient D^K_{pm} is obtained by

$$D_{pm} = \tau\phi S_g D^{aw}_g , \tag{4.13}$$

where τ is the tortuosity of the porous medium and D^{aw}_g is the binary diffusion coefficient of air/steam.

Thermal energy balance

$$\begin{aligned}
&\phi \frac{\partial \left(\sum_\alpha \rho_{mass,\alpha} u_\alpha S_\alpha\right)}{\partial t} + (1-\phi)\frac{\partial \rho_s c_s T}{\partial t} - \nabla \cdot (\lambda_{pm} \nabla T) \\
&- \sum_\alpha \nabla \cdot \left\{ \frac{k_{r\alpha}}{\mu_\alpha} \rho_{mass,\alpha} h_\alpha \mathbf{K} (\nabla p_\alpha - \rho_{mass,\alpha} \mathbf{g}) \right\} - \sum_K \nabla \cdot \left\{ D_{pm} \rho_{mol,g} h^K_g M^K \nabla x^K_g \right\} \\
&- q^h = 0, \qquad K \in \{wa, ai\},\ \alpha \in \{w, g\},
\end{aligned} \tag{4.14}$$

where c_s is the specific heat capacity of the soil grains. u_α and h_α denote the specific internal energy and enthalpy of the phases respectively. λ_{pm} represents the heat conductivity averaged over the whole fluid-filled porous medium.

Mass/energy transfer and local phase state

The nonisothermal systems that we typically investigate are characterized by the possibility of mass transfer and phase appearance/disappearance due to mass transfer processes such as evaporation, condensation, dissolution, and degassing.

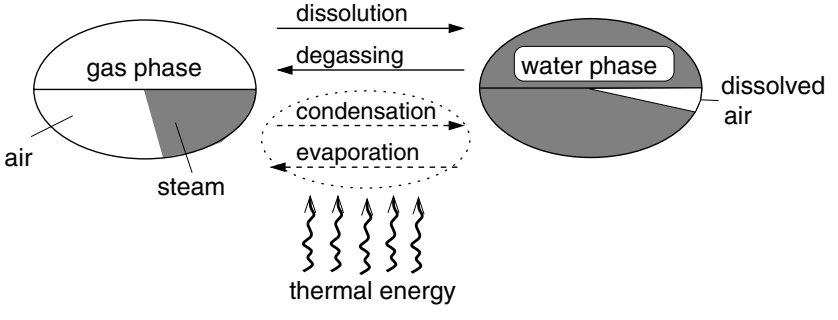


Fig. 4.1 Phases, components, and transfer processes of mass and energy between the fluid phases (modified according to [29])

This is sketched in Figure 4.1. In particular, condensation and evaporation are coupled with a strong exchange of thermal energy. When a phase appears or disappears locally, the number and the combination of the fluid phases being present at that point change. Therefore, we introduce the term phase state. In a water–gas system, there are three possible phase states (see Tab. 4.1). The ratios of the components K in the fluid phases α are expressed by mole fractions x^K_α. The mole fraction

Table 4.1 Phase states and corresponding set of primary variables

phase state	primary variables
both phases	S_w, p_g, T
gas phase	x_g^{wa}, p_g, T
water phase	x_w^{ai}, p_g, T

of dissolved air in the water phase is very small and can be described by Henry's Law. The mole fraction of steam in the gas phase is determined by the saturation vapor pressure, which is a function of temperature. Note that this works so long as water is also present as a liquid phase. If the water phase is absent or has just disappeared, for example due to evaporation, the mole fraction of steam in the gas phase x_g^{wa} is an independent variable. In such a case, we choose x_g^{wa} to be one of the primary variables.

We can see that the set of three primary variables for the three equations is not constant and depends on the local phase state. Tab. 4.1 lists the possible phase states and the corresponding primary variables. Note, that we use p_g as a primary variable for all phase states, also for the state 'water phase', although p_g is not a physically defined parameter here. We can do so by interpreting p_g as the total pressure of the system which is coupled to p_w via the capillary pressure–saturation function. Changes of the local phase states, i.e., the appearance or disappearance of fluid phases, must be recognized by the model. This requires the formulation of an algorithm providing criteria for the indication of a phase state change. Phase disappearance is simply indicated by negative values of the corresponding saturations. An appearance of the phases requires a distinction in the algorithm between the case when liquid water appears and the case when the gas phase appears. Water appears when the partial pressure of steam in the gas phase exceeds the saturation vapor pressure

$$p_g^{wa} = x_g^{wa} p_g > p_{sat}^{wa}(T) \,. \tag{4.15}$$

Gas appears as a phase when the sum of the (hypothetical) vapor pressure exceeds the total pressure given by p_g, which in this case is also a hypothetical gas phase pressure

$$H_w^{ai} x_w^{ai} + p_{sat}^{wa} > p_g \,, \tag{4.16}$$

where H_w^{ai} is Henry's constant for the dissolution of air in water.

4.2.3 Constitutive Relationships

For the closure of the multiphase flow equations, a set of constitutive relationships is required in order to describe the secondary variables which dependent on the primary variables. We can distinguish between constitutive relations which describe the fluid properties, and those which quantify the interaction between the phases and the porous medium.

For the density, we can formulate the total differential as

$$d\rho = \rho\beta_p\, dp + \rho\beta_T\, dT \tag{4.17}$$

with the isothermal compressibility coefficient $\beta_p = \frac{1}{\rho}\frac{\partial\rho}{\partial p}$ and the isobaric volume expansion coefficient $\beta_T = \frac{1}{\rho}\frac{\partial\rho}{\partial T}$. Further, the water ($w$) phase is assumed to be incompressible, such that $\beta_{p_w} = 0$. For a gaseous phase (g) like air, we can calculate the density by assuming the validity of the Ideal Gas Law:

$$\rho_g = \frac{p_g}{R_g T}, \tag{4.18}$$

where R_g is the individual gas constant obtained from

$$R_g = \frac{R_u}{M_g}, \tag{4.19}$$

with $R_u = 8.314$ J/(mole K) being the universal gas constant and M_g the molecular weight. For air, the gas constant is $R_{air} \approx 287$ J/(kg K).

The viscosity for all phases is mainly dependent on temperature. Thus, we use constant viscosities for the isothermal case. Several approaches can be found in the literature which consider temperature dependence. We use the relations given by [1].

For nonisothermal systems, we have to determine caloric state variables for the energy balance. The specific internal energy u_α represents the total energy of the molecules of phase α per unit mass. The specific enthalpy h_α is related to u_α by

$$h_\alpha = u_\alpha + \frac{p_\alpha}{\rho_{mass,\alpha}}\,. \tag{4.20}$$

For the water phase, the term $p_w/\rho_{mass,w}$ can be neglected compared to u_w and we approximate $u_w \approx h_w$. However, this term must be considered for the gas phase due its lower density. Values of the specific enthalpy and internal energy depend both on the pressure and the temperature and can be taken, for example, from the International Formulation Committee (1967) [1].

Up to now, we have discussed some properties of the fluid phases. Numerous values and functions describing them rather accurately can be found in the literature. The correct description of the interaction between fluid phases and the porous medium plays a key role in the description of the relationships for the capillary pressure and the relative permeabilities dependent on the phase saturations.

In recent years, a number of approaches have been developed for the description of the capillary pressure–saturation behavior of two fluid phases in a porous medium. Among the most well known approaches are those of Brooks & Corey (1964) [25] and of van Genuchten (1980) [161]. Both use parameterized functionals, which, however, differ characteristically if the wetting phase saturation approaches one ($S_w \to 1$). The Brooks–Corey (BC) approach is formulated as

$$p_c = p_d S_e^{-1/\lambda} \tag{4.21}$$

and the van Genuchten (VG) approach as

$$p_c = \frac{1}{\alpha}\left(S_e^{-1/m} - 1\right)^{1/n} \tag{4.22}$$

with

$$S_e = \frac{S_w - S_{w,r}}{1 - S_{w,r}} \tag{4.23}$$

and

$$m = 1 - \frac{1}{n}\,. \tag{4.24}$$

$S_{w,r}$ is the residual wetting phase saturation and S_e the effective saturation of the wetting phase. p_d, λ, α, and n are parameters, which can be determined by curve fitting to the experimental data. Lenhard et al. (1989) [114] give a correlation between the BC (p_d, λ) and the VG (α, m, n) parameters.

There exist also numerous functions for the description of the relative permeability–saturation behavior. Again, among the most well-known are the Brooks–Corey and the van Genuchten approach, which can be derived from the corresponding capillary pressure functions by using pore network and capillary tube models according to Burdine (1953) [26] and Mualem (1976) [127]. The BC functions for the wetting and the non-wetting phases yield

$$k_{r,w} = S_e^{\frac{2+3\lambda}{\lambda}}, \tag{4.25}$$

$$k_{r,n} = (1 - S_e)^2\left(1 - S_e^{\frac{2+\lambda}{\lambda}}\right), \tag{4.26}$$

and the VG functions

$$k_{r,w} = \sqrt{S_e}\left[1 - \left(1 - S_e^{1/m}\right)^m\right]^2, \tag{4.27}$$

$$k_{r,n} = (1 - S_e)^{\frac{1}{3}}\left[1 - S_e^{1/m}\right]^{2m}. \tag{4.28}$$

4.3 The Forward Simulation Problem

4.3.1 Governing Equations

In this section we present the governing equations for non-isothermal two-phase two-component system which are described by, following the discussion in the previous section, two mass balance equations and one energy balance equation. Corresponding equations for isothermal two-phase system can be found in [73]. The mass balance equations for component κ ($\kappa = w$(water),a(air)) read as follows [75]

$$\phi \frac{\partial(\rho_g X_g^\kappa (1-S_w)}{\partial t} + \phi \frac{\partial(\rho_w X_w^\kappa S_w))}{\partial t} + \nabla.\left\{\rho_g X_g^\kappa \mathbf{U}_g\right\} + \nabla.\left\{\rho_w X_w^\kappa \mathbf{U}_w\right\} - \nabla.\left\{D_{pm}\rho_g \nabla X_g^\kappa\right\} - q_m^\kappa = 0, \tag{4.29}$$

where the *Darcy's* velocities are given by

$$\mathbf{U}_g = -\lambda_g \mathbf{v}_g, \qquad \mathbf{v}_g = \mathbf{K}(\nabla p_g - \rho_g \mathbf{g}), \tag{4.30}$$

$$\mathbf{U}_w = -\lambda_w \mathbf{v}_w, \qquad \mathbf{v}_w = \mathbf{K}(\nabla p_g - \nabla p_c - \rho_w \mathbf{g}), \tag{4.31}$$

with the mobility terms

$$\lambda_g = \frac{k_{rg}}{\mu_g} \quad \text{and} \quad \lambda_w = \frac{k_{rw}}{\mu_w}. \tag{4.32}$$

Here ϕ is the porosity, S_α is the saturation of phase α, X_α^κ is the mass fraction of component κ in phase α, ρ_α is the phase density, $k_{r\alpha}$ is the relative permeability of phase α, μ_α is the dynamic viscosity of phase α, $\mathbf{K}$ is the absolute permeability tensor of the porous medium, D_{pm} is the diffusion coefficient, $\mathbf{g}$ is the vector for gravitational force and q_m^κ is the source/sink term.

The energy balance equation reads as

$$(1-\phi)\rho_s C_s \frac{\partial T}{\partial t} + \phi \frac{\partial(\rho_g u_g (1-S_w))}{\partial t} + \phi \frac{\partial(\rho_w u_w S_w)}{\partial t} + \nabla.\left\{\rho_g h_g \mathbf{U}_g\right\} + \nabla.\left\{\rho_w h_w \mathbf{U}_w\right\} - \nabla.\left\{\lambda_{pm}\nabla T\right\} - \nabla.\left\{D_{pm}\rho_g h_g^a \nabla X_g^a\right\} - \nabla.\left\{D_{pm}\rho_g h_g^w \nabla X_g^w\right\} - q_e = 0, \tag{4.33}$$

where T is the temperature, ρ_s is the soil grain density, C_s is the heat capacity of the soil grains, u_α is the internal energy of phase α, h_α represents the specific enthalpy of phase α, λ_{pm} is the heat conductivity of the fluid filled porous medium and q_e is the heat source/sink.

The initial and boundary conditions are

$$S_w(x,0) = S_{w0}(x), \quad p_g(x,0) = p_{g0}(x), \quad T(x,0) = T_0(x), \quad \text{for } x \in \Omega, \tag{4.34}$$

$$S_w(x,t) = S_{wd}(x,t), \quad p_g(x,t) = p_{gd}(x,t), \quad T(x,t) = T_d(x,t), \text{ on } \Gamma_{\alpha d}, \tag{4.35}$$

$$\rho_\alpha \mathbf{U}_\alpha \cdot n = F_\alpha(x,t), \ \nabla T \cdot n = F_1(x,t), \text{ on } \Gamma_{\alpha n}, \tag{4.36}$$

where $\Gamma_{\alpha d}$ is the *Dirichlet* part of the boundary and $\Gamma_{\alpha n}$ is the *Neumann* part of the boundary of the domain Ω.

In a two-phase two-component system of water/steam/air, if both the water phase - containing water and dissolved air - and the gas phase - containing air and steam - are present, they are assumed to be in thermodynamic equilibrium. We neglect chemical reactions and biological decompositions. Air is fully saturated with steam. The three primary variables are pressure, temperature and an unknown saturation of

one of the phases. If one of the phases disappears, the primary variable saturation has to be substituted by the air mass fraction in the remaining phase.

The constitutive relationships used in our computations for capillary pressure and relative permeabilities dependent on the phase saturations and are due to *van Genuchten* [161] as described earlier.

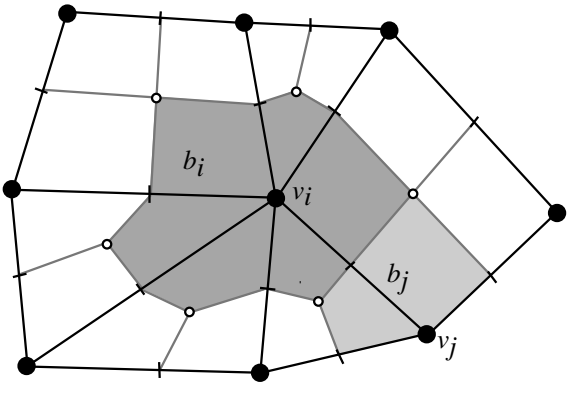


Fig. 4.2 Control volume

4.4 Discretization

The forward problem is solved by using the numerical simulator *MUFTE-UG*, where a vertex centered finite volume element method with fully implicit time discretization on unstructured meshes has been implemented (as described in [8, 9, 36]). In this method, the polyhedral domain Ω is divided into meshes $E_h = \{e_1, e_2, \cdots, e_k\}$ consisting of elements e_i with mesh width h. The set of vertices is denoted by $V = \{v_1, v_2, \cdots, v_n\}$, the location of vertex v_i is $\mathbf{x}_i$ and the barycentre of element e_k is $\mathbf{x}^k$. Furthermore, $V(k)$ denotes the set of all indices i where v_i is a corner of the element e_k and conversely $E(i)$ is the set of all indices k such that $i \in V(k)$. The secondary or dual mesh is constructed on the basis of E_h by connecting the element barycentres to the edge midpoints (Fig. 4.2). The secondary mesh $B_h = \{b_1, b_2, \cdots, b_n\}$ consists of polyhedral regions b_i called *boxes* or *control volumes*. V_h is the space of lowest order conforming finite element functions or 'hat'-functions $\{\xi_i\}$ associated with E_h and W_h is the space of test functions which are the characteristic functions $\{\chi_i\}$ of the control volumes B_h. Thus for any $u_h \in V_h$ and $w_h \in W_h$ one has $u_h = \sum_{i \in E(i)} u_i \xi_i(\mathbf{x})$ and $w_h = \sum_{i \in E(i)} w_i \chi_i(\mathbf{x})$ with $u_i = u_h(\mathbf{x}_i)$ and $w_i = w_h(\mathbf{x}_i)$. Every finite element function $u_h \in V_h$ is identified with a vector $\mathbf{u} \in \mathbb{R}^N$ by a mapping $I_h : \mathbb{R}^N \to V_h$ in the usual way: $I_h(\mathbf{u}) = u_h$.

The semi-discretization of equations (4.29) and (4.33) implies that the corresponding weak form of the equations are valid in each of the control volumes b_i ($i \in I = \{1, 2, \cdots, n\}$) (fig.1) which are given by ($\kappa = w, a$)

$$\frac{d}{dt}\left[\int_{b_i}\phi_h\rho_{gh}X^{\kappa}_{gh}(1-S_{wh})\,dV+\int_{b_i}\phi_h\rho_{wh}X^{\kappa}_{wh}S_{wh}\,dV\right]-\int_{\partial b_i}\rho_{gh}X^{\kappa}_{gh}\lambda_{gh}\mathbf{v}_{gh}\cdot n\,ds$$
$$-\int_{\partial b_i}\rho_{wh}X^{\kappa}_{wh}\lambda_{wh}\mathbf{v}_{wh}\cdot n\,ds-\int_{\partial b_i}D_{pmh}\rho_{gh}\nabla X^{\kappa}_{gh}\cdot n\,ds-\int_{b_i}q^{\kappa}_{mh}\,dV=0. \tag{4.37}$$

$$\frac{d}{dt}\left[\int_{b_i}(1-\phi_h)\rho_{sh}C_{sh}T_h\,dV+\int_{b_i}\phi_h\rho_{gh}u_{gh}(1-S_{wh})\,dV+\int_{b_i}\phi_h\rho_{wh}u_{wh}S_{wh}\,dV\right]$$
$$-\int_{\partial b_i}\rho_{gh}h_{gh}\lambda_{gh}\mathbf{v}_{gh}\cdot n\,ds-\int_{\partial b_i}\rho_{wh}h_{wh}\lambda_{wh}\mathbf{v}_{wh}\cdot n\,ds-\int_{\partial b_i}\lambda_{pmh}\nabla T_h\cdot n\,ds$$
$$-\int_{\partial b_i}D_{pmh}\rho_{gh}h^{a}_{gh}\nabla X^{a}_{gh}\cdot n\,ds-\int_{\partial b_i}D_{pmh}\rho_{gh}h^{w}_{ah}\nabla X^{w}_{gh}\cdot n\,ds-\int_{b_i}q_{eh}\,dV=0. \tag{4.38}$$

Using the basis function representation, we get from (4.37) (for $\kappa=w,a$)

$$\frac{d}{dt}\left[\sum_{n\in E(i)}(\phi_i\rho_{g,i}X^{\kappa}_{g,i}(1-S_{w,i}).\mathrm{meas}(b^n_i)+\sum_{n\in E(i)}(\phi_i\rho_{w,i}X^{\kappa}_{w,i}S_{w,i}.\mathrm{meas}(b^n_i)\right]$$
$$-\sum_{n,j}\rho_{gh}(\mathbf{x}^n_{ij})X^{\kappa}_{gh}(\mathbf{x}^n_{ij})\left[\lambda_{gh}\right]^n_{ij}\mathbf{v}_{gh}(\mathbf{x}^n_{ij})\cdot n^n_{ij}\mathrm{meas}(\gamma^n_{ij})-\sum_{n,j}\rho_{wh}(\mathbf{x}^n_{ij})X^{\kappa}_{wh}(\mathbf{x}^n_{ij})$$
$$[\lambda_{wh}]^n_{ij}\,\mathbf{v}_{wh}(\mathbf{x}^n_{ij})\cdot n^n_{ij}\mathrm{meas}(\gamma^n_{ij})-\sum_{n,j}D_{pmh}(\mathbf{x}^n_{ij})\rho_{gh}(\mathbf{x}^n_{ij})\tilde{X}^{\kappa}_{gh}(\mathbf{x}^n_{ij})\cdot n^n_{ij}\mathrm{meas}(\gamma^n_{ij})$$
$$-\sum_{n\in E(i)}q^{\kappa}_{m,i}\mathrm{meas}(b^n_i)=0, \tag{4.39}$$

where $\mathbf{x}^n_{ij}$ is the barycentre of a sub-control volume face associated with the element e_n. The fluxes are evaluated as follows

$$\mathbf{v}_{gh}(\mathbf{x}^n_{ij})=-\mathbf{K}(\mathbf{x}^n)\left[\sum_{m\in I}\left(p_{g,m}\nabla\xi_m(\mathbf{x}^n_{ij})-\rho_{g,m}\xi_m(\mathbf{x}^n_{ij})\mathbf{g}\right)\right], \tag{4.40}$$

and the mobilities are evaluated using upwinding as follows

$$\left[\lambda_{gh}\right]^n_{ij}=(1-\varepsilon)\lambda_{gh}(\mathbf{x}^n_{ij})+\varepsilon\begin{cases}\lambda_{g,i}\ \text{if} & \mathbf{v}_{gh}(\mathbf{x}^n_{ij})\cdot n^n_{ij}\geq 0\\ \lambda_{g,i} & \text{else,}\end{cases} \tag{4.41}$$

where ε lies in $[0,1]$. Similarly equation (4.38) can be written as

$$\frac{d}{dt}\left[\sum_{n\in E(i)}(1-\phi_i)\rho_{s,i}C_{s,i}T_i\text{meas}(b_i^n)+\sum_{n\in E(i)}\phi_i\rho_{g,i}u_{g,i}(1-S_{w,i})\text{meas}(b_i^n)+\right.$$
$$\left.\sum_{n\in E(i)}\phi_i\rho_{w,i}u_{w,i}S_{w,i}\text{meas}(b_i^n)\right]-\sum_{n,j}\rho_{gh}(\mathbf{x}_{ij}^n)h_{gh}(\mathbf{x}_{ij}^n)\left[\lambda_{gh}\right]_{ij}^n\mathbf{v}_{gh}(\mathbf{x}_{ij}^n)\cdot n_{ij}^n\text{meas}(\gamma_{ij}^n)$$
$$-\sum_{n,j}\rho_{wh}(\mathbf{x}_{ij}^n)h_{wh}(\mathbf{x}_{ij}^n)\left[\lambda_{wh}\right]_{ij}^n\mathbf{v}_{wh}(\mathbf{x}_{ij}^n)\cdot n_{ij}^n meas(\gamma_{ij}^n)-\sum_{n,j}\lambda_{pmh}(\mathbf{x}_{ij}^n)\tilde{T}_h(\mathbf{x}_{ij}^n)$$
$$\cdot n_{ij}^n\text{meas}(\gamma_{ij}^n)-\sum_{n,j}D_{pmh}(\mathbf{x}_{ij}^n)\rho_{gh}(\mathbf{x}_{ij}^n)h_{gh}^a(\mathbf{x}_{ij}^n)\tilde{X}_{gh}^a(\mathbf{x}_{ij}^n)\cdot n_{ij}^n\text{meas}(\gamma_{ij}^n)$$
$$-\sum_{n,j}D_{pmh}(\mathbf{x}_{ij}^n)\rho_{gh}(\mathbf{x}_{ij}^n)h_{ah}^w(\mathbf{x}_{ij}^n)\tilde{X}_{gh}^w(\mathbf{x}_{ij}^n)\cdot n(\mathbf{x}_{ij}^n)\text{meas}(\gamma_{ij}^n)$$
$$-\sum_{n\in E(i)}q_{e,i}^{\kappa}\text{meas}(b_i^n)=0, \tag{4.42}$$

where $\tilde{T}_h(\mathbf{x}_{ij}^n)$, $\tilde{X}_{gh}^a(\mathbf{x}_{ij}^n)$ and $\tilde{X}_{gh}^w(\mathbf{x}_{ij}^n)$ are evaluated according to (4.40).

This semidiscrete formulation will lead to a system of ODEs. That is, for $0<t<T^*$ one has to find the vectors $\mathbf{p_g}(\mathbf{t})$, $\mathbf{S_w}(\mathbf{t})$, $\mathbf{T}(\mathbf{t})$ such that for $\alpha=g,w,e$:

$$\frac{d}{dt}\mathbf{M}_\alpha(\mathbf{p_g},\mathbf{S_w},\mathbf{T})+\mathbf{A}_\alpha(\mathbf{p_g},\mathbf{S_w},\mathbf{T})+\mathbf{Q}_\alpha(\mathbf{t},\mathbf{p_g},\mathbf{S_w},\mathbf{T})=\mathbf{0}. \tag{4.43}$$

The vector $\mathbf{M}_\alpha$ represents the accumulation term, $\mathbf{A}_\alpha$ the flux term and $\mathbf{Q}_\alpha$ the source/sink and boundary flux terms. This system can be formally rewritten as

$$\begin{pmatrix}M_{gg} & M_{gw} & M_{ge}\\ M_{wg} & M_{ww} & M_{we}\\ M_{eg} & M_{ew} & M_{ee}\end{pmatrix}\begin{pmatrix}\frac{\partial\mathbf{p_g}}{\partial t}\\ \frac{\partial\mathbf{S_w}}{\partial t}\\ \frac{\partial\mathbf{T}}{\partial t}\end{pmatrix}+\begin{pmatrix}\mathbf{A_g}(\mathbf{p_g},\mathbf{S_w},\mathbf{T})+\mathbf{Q_g}(\mathbf{t},\mathbf{p_g},\mathbf{S_w},\mathbf{T})\\ \mathbf{A_w}(\mathbf{p_g},\mathbf{S_w},\mathbf{T})+\mathbf{Q_w}(\mathbf{t},\mathbf{p_g},\mathbf{S_w},\mathbf{T})\\ \mathbf{A_e}(\mathbf{p_g},\mathbf{S_w},\mathbf{T})+\mathbf{Q_e}(\mathbf{t},\mathbf{p_g},\mathbf{S_w},\mathbf{T})\end{pmatrix}$$
$$=0, \tag{4.44}$$

with the (solution-dependent) submatrices given by

$$(M_{\alpha g})_{ij}=\frac{\partial\mathbf{M}_{\alpha\mathbf{g},\mathbf{i}}}{\partial\mathbf{p}_{\mathbf{g},\mathbf{j}}},\qquad(M_{\alpha w})_{ij}=\frac{\partial\mathbf{M}_{\alpha\mathbf{w},\mathbf{i}}}{\partial\mathbf{S}_{\mathbf{w},\mathbf{j}}},\qquad(M_{\alpha e})_{ij}=\frac{\partial\mathbf{M}_{\alpha\mathbf{e},\mathbf{i}}}{\partial\mathbf{T}_{\mathbf{j}}}.$$

In the case of isothermal two-phase flow, the variable T (temperature) will not appear and the submatrices will have dimension 2×2 as discussed in [73].

4.4.1 Implicit Time Discretization

For the time discretization, we use an implicit scheme. For notational ease, the evaluation of any quantity at time level t^n is denoted by a superscript n e.g., $p_{wh}(t^n)=p_{wh}^n$, $S_n(t^n)=S_n^n$ etc. The notation for a time step is

$$\Delta t^n = t^{n+1} - t^n.$$

The one step θ-scheme [75] applied to the semi-discrete system (4.44) yields p_w^n, S_n^n such that for $\alpha = w, n$

$$\mathbf{M}_\alpha^{\mathbf{n+1}} - \mathbf{M}_\alpha^{\mathbf{n}} + \Delta t^n \theta \left(\mathbf{A}_\alpha^{\mathbf{n+1}} + \mathbf{Q}_\alpha^{\mathbf{n+1}} \right) + \Delta t^n (1-\theta)(\mathbf{A}_\alpha^{\mathbf{n}} + \mathbf{Q}_\alpha^{\mathbf{n}}) = 0, \quad (4.45)$$

with $\mathbf{M}_\alpha^{\mathbf{n}} = \mathbf{M}_\alpha(\mathbf{p}_\mathbf{w}^{\mathbf{n}}, \mathbf{S}_\mathbf{n}^{\mathbf{n}})$, etc. For $\theta = 1$, we obtain the first order accurate backward Euler scheme and for $\theta = 1/2$ the Crank-Nicolson scheme which is second order accurate in time. Here, we use the backward Euler scheme since Crank-Nicolson has only weak damping properties which may cause stability problems as the equations are of mixed parabolic and hyperbolic types. For details of the step selection process, see [9].

4.5 The Software System MUFTE_UG

The discretization techniques discussed above and fast solvers for the simulation of multiphase–multicomponent flow in porous and highly heterogeneous media are being developed by interdisciplinary teams at the Institute of Hydraulic Research (IWS), University of Stuttgart, and at the Technical Simulation Group of the Interdisciplinary Center for Scientific Computing, University of Heidelberg (IWR). The platform for this work is the numerical simulation program MUFTE_UG, which combines the physical procedure and the discretization techniques of the program system MUFTE (Multiphase Flow, Transport and Energy Model, IWS) with the solvers and multigrid techniques of the program system UG (Unstructured Grids, IWR). In MUFTE_UG [77], all modules are made available in such a way that they can be combined easily. A good overview of the available isothermal and nonisothermal MUFTE modules in combination with the solution and discretization techniques offered in UG can be found, for example, in [9] and [30].

Chapter 5
Parameter Identification in Multiphase Flow through Porous Media

5.1 Introduction

As presented in the previous chapter, the governing partial differential equations or the constitutive relationships involve parameters representing the properties of the fluids, the media and/or their interactions. In practical situations these parameters cannot be measured directly. Rather, they are to be determined from a set of observation data. Two types of methods have been reported, namely, direct and indirect methods (cf., e.g., [106]). In the direct methods, the parameters are determined by inverting the governing equations with simplified initial and boundary conditions using analytical or semi-analytical methods. These methods have various limitations and cannot be applied to field-scale models. Indirect methods, on the other hand, are quite flexible and can be applied to practical problems. Our parameter identification technique is one of the indirect methods. In this technique, the direct problem is posed for prescribed but arbitrary initial and boundary conditions which can be solved by any appropriate analytical or numerical technique. The constitutive relationships intended to be applied are parameterized based on a-priori knowledge, and coefficients are determined by means of an optimization algorithm that extremizes some objective function. The drawback of this method is that it cannot determine the specific form of the constitutive relationships and one has to presume some formulation of these relationships which holds to a sufficient degree of approximation. Many inverse problems are ill-posed which is characterized by non-uniqueness and instability [168], and this causes uncertainty in the determined parameters. This method also has the advantage that it is possible to obtain information concerning the parameter uncertainty from the estimation analysis.

The basic methodology explained here can be applied to a general parameter identification in non-stationary multiphase models and can also be found in [73, 72, 74].[1] The current inverse modeling methodology is dominated by approaches which can be characterized by treating the multiphase simulation solver

[1] Reprinted with kind permissions of Springer-Verlag and Springer Science and Business Media.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 43–62.
springerlink.com

routine in the form of a black box, which just matches the unknown parameters (to be estimated) via a nonlinear process to an output least squares functional. This is the case, e.g., in ITOUGH/ITOUGH2 [41] and also in [27], [104]. From the point of view of boundary value problems for non-stationary processes, this can be seen as a single shooting approach to the parameter identification problem, which, on the other hand, shares more properties with boundary value problems than pure initial value problems. As it is known that single shooting reveals instabilities for boundary value problems in ODE, a similar behavior has been observed with these black box approaches. Here, we use a multiple shooting approach similar to [148]. The multiple shooting by itself leads to a more robust solution behavior than a single shooting approach. The overall multiphase system solution technology is taken from the code MUFTE-UG (described in the previous chapter), which is enhanced by a multiple-shooting framework and computation of necessary derivatives.

5.2 Least-Squares Formulation

In order to perform a maximum likelihood estimation with respect to the output errors in measured data Z_{ij} of functions ψ_{ij} of the variables S_w, p_g and T, we formulate a pointwise weighted least squares function to be minimized [73, 72, 74]:

$$\min \frac{1}{2} \sum_{i,j} \left(\psi_{ij}(p_g, S_w, T, \beta) - Z_{ij} \right)^2 / \sigma_{ij}^2. \tag{5.1}$$

Here, Z_{ij} are measurements of the saturation of water taken at the j-th measurement in time ($\hat{t}_j$) and the i-th measurement position in space ($\hat{x}_i$). The measurement errors are assumed to be independently normally distributed with expectation 0 and standard deviation $\sigma_{i,j}$. This objective functional is subject to the condition that the ODE (4.44) together with the suitable initial and boundary conditions is satisfied over the time horizon $[0, D] \ni \{\hat{t}_j\}_j$. The vector β collects the unknown parameters to be estimated.

5.3 The Multiple Shooting Parameter Estimation Approach

We subdivide the time interval under consideration, $(0, D)$ into subintervals with the grid points $0 = \tau_0 < \tau_1 < \tau_2 < \cdots < \tau_m = D$, where in general the nodes $\{\tau_j\}$ are independent from the measurement points in time, but they typically include a subset of the measurement points. For ease of presentation, however, we let the measurement time-grid coincide with the multiple shooting time-grid, since the necessary generalizations are obvious. At these nodes the initial values of the state or differential variables S_j, p_j and T_j are introduced as unknowns in addition to the parameter vector β. In a standard multiple shooting formulation, these additional degrees of freedom are constrained by explicitly formulating continuity equations for the state variables. Thus we arrive at the (time-) discretized least-squares problem

$$\min_{\{p_j,S_j,T_j\}_j,\beta} \frac{1}{2}\sum_{i,j}\left(\psi_{ij}(p_j,S_j,T_j,\beta)-Z_{ij}\right)^2/\sigma_{ij}^2, \tag{5.2}$$

subject to the continuity conditions

$$\begin{aligned} p_{j+1}-p_g(\tau_{j+1};p_j,S_j,T_j,\beta)&=0,\\ S_{j+1}-S_w(\tau_{j+1};p_j,S_j,T_j,\beta)&=0,\\ T_{j+1}-T(\tau_{j+1};p_j,S_j,T_j,\beta)&=0, \end{aligned} \tag{5.3}$$

for $j=0,1,\cdots,(m--1)$ and where $p_g(\tau_{j+1};p_j,S_j,T,\beta)$, $S_w(\tau_{j+1};p_j,S_j,T,\beta)$, $T(\tau_{j+1};p_j,S_j,T,\beta)$ denote the solution at time τ_{j+1} of the multiphase ODE (4.44) with its initial and boundary conditions together with the additional initial conditions $S_w(\tau_j)=S_j$, $p_g(\tau_j)=p_j$ and $T(\tau_j)=T_j$.

In case of isothermal two phase flow, we only have the continuity condition for the variable saturation. The semidiscretization of equations (4.10) and (4.11) leads to a semi-explicit DAE where S_w is the differential variable and p_n is the algebraic variable. Otherwise the treatment remains the same. Details of the method for isothermal problem can be found in [73].

5.4 A Reduced Generalized Gauss-Newton Method

An efficient numerical solution technique for the discretized parameter identification problem described in the previous section is the application of generalized Gauss-Newton methods as introduced in [20]. Increments to be added in each iteration are computed by solving the linearized constrained least squares problem

$$\begin{aligned} \min_{\{\Delta p_j,\Delta S_j,\Delta T_j\}_j,\Delta\beta} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=0}^{m}\Bigg\{ &\psi_{ij}(p_j,S_j,T_j,\beta)-Z_{ij}+\\ &\left(\frac{\partial\psi_{ij}}{\partial p_j}\quad\frac{\partial\psi_{ij}}{\partial S_j}\quad\frac{\partial\psi_{ij}}{\partial T_j}\quad\frac{\partial\psi_{ij}}{\partial\beta}\right)\begin{pmatrix}\Delta p_j\\ \Delta S_j\\ \Delta T_j\\ \Delta\beta\end{pmatrix}\Bigg\}^2/\sigma_{ij}^2, \end{aligned} \tag{5.4}$$

subject to

$$\begin{aligned} G_j^{pp}\Delta p_j+G_j^{ps}\Delta S_j+G_j^{pT}\Delta T_j-\Delta p_{j+1}+G_j^{p\beta}\Delta\beta&=d_{j+1}^p,\\ G_j^{sp}\Delta p_j+G_j^{ss}\Delta S_j+G_j^{sT}\Delta T_j-\Delta S_{j+1}+G_j^{s\beta}\Delta\beta&=d_{j+1}^s,\\ G_j^{Tp}\Delta p_j+G_j^{Ts}\Delta S_j+G_j^{TT}\Delta T_j-\Delta T_{j+1}+G_j^{T\beta}\Delta\beta&=d_{j+1}^T, \end{aligned} \tag{5.5}$$

for $(j=0,1,2,\ldots,m)$, where m is the number of shooting intervals, N is the number of measurements, d_j are the defects or mismatches due to linearization, G_j^{pp}, G_j^{ps} etc. are the Wronskians which, for example, are given by

$$G_j^{sp} = \frac{\partial S_w\left(\tau_{j+1}; p_j, S_j, T_j, \beta\right)}{\partial p_j}, \quad G_j^{ss} = \frac{\partial S_w\left(\tau_{j+1}; p_j, S_j, T_j, \beta\right)}{\partial S_j},$$
$$G_j^{sT} = \frac{\partial S_w\left(\tau_{j+1}; p_j, S_j, T_j, \beta\right)}{\partial T_j}, \quad G_j^{s\beta} = \frac{\partial S_w\left(\tau_{j+1}; p_j, S_j, T_j, \beta\right)}{\partial \beta}, \quad \text{etc.}$$

The equation (5.5) results from (5.3) due to the fact that the continuity conditions are not satisfied exactly at each of the shooting nodes during the iteration procedure. The Wronskians $G_j^{k,l}$ and $G_j^{k\beta}$ $(k,l = \{s,p,T\})$ cannot be computed practically or even stored in the case of time dependent PDEs. In order to avoid that, we apply a reduction technique, first proposed in [145] and used in [137] for general nonlinear optimal control problem, in [35, 34] for parameter estimation application for flow in porous media. For the applications in our case, we rewrite (5.5) as

$$\begin{pmatrix} G_j & G_j^{\beta} \end{pmatrix} \begin{pmatrix} \Delta_j \\ \Delta\beta \end{pmatrix} - \Delta_{j+1} = d_{j+1}, \tag{5.6}$$

where the block matrices and the vectors are given as

$$G_j = \begin{pmatrix} G_j^{pp} & G_j^{pT} & G_j^{pT} \\ G_j^{sp} & G_j^{ss} & G_j^{sT} \\ G_j^{Tp} & G_j^{Ts} & G_j^{TT} \end{pmatrix}, G_j^{\beta} = \begin{pmatrix} G_j^{p\beta} \\ G_j^{s\beta} \\ G_j^{T\beta} \end{pmatrix}, \Delta_j = \begin{pmatrix} \Delta p_j \\ \Delta S_j \\ \Delta T_j \end{pmatrix}, d_j = \begin{pmatrix} d_j^p \\ d_j^s \\ d_j^T \end{pmatrix}.$$

Then we can solve (5.6) for Δ_{j+1} in recursion as

$$\Delta_{j+1} = -\sum_{l=1}^{j+1} \left(\prod_{i=l}^{j} G_i \right) d_l + \left\{ \sum_{l=1}^{j+1} \left(\prod_{i=l}^{j} G_i \right) G_{l-1}^{\beta} \right\} \Delta\beta + \prod_{i=0}^{j} G_i \Delta_0. \tag{5.7}$$

Since we assume to have full information on the initial data, we know $\Delta_0 = 0$ and it can be neglected in what follows. Now the linear quadratic problem can be reformulated as an unconstrained quadratic problem,

$$\min_{\Delta\beta} \frac{1}{2} \left[\sum_{i=1}^{N} \sum_{j=0}^{m} \left\{ \psi_{ij}\left(p_j, S_j, T_j, \beta\right) - Z_{ij} - \begin{pmatrix} \frac{\partial \psi_{ij}}{\partial p_j} & \frac{\partial \psi_{ij}}{\partial S_j} & \frac{\partial \psi_{ij}}{\partial T_j} \end{pmatrix} g_j^s \right.\right.$$
$$\left.\left. + \left(\begin{pmatrix} \frac{\partial \psi_{ij}}{\partial p_j} & \frac{\partial \psi_{ij}}{\partial S_j} & \frac{\partial \psi_{ij}}{\partial T_j} \end{pmatrix} g_j^{\beta} + \frac{\partial \psi_{ij}}{\partial \beta} \right) \Delta\beta \right\}^2 \right], \tag{5.8}$$

where

$$g_j^s = \sum_{l=1}^{j} \left(\prod_{k=l}^{j} G_k \right) d_l \tag{5.9}$$

$$g_j^{\beta} = \sum_{l=1}^{j} \left(\prod_{k=l}^{j} G_k \right) G_{l-1}^{\beta}. \tag{5.10}$$

The vector g_j^s and the matrix g_j^β can be computed in parallel to the solution of the forward multiple shooting sweep in each nonlinear iteration. This QP is solved for the parameter vector increment $\Delta\beta$. The Levenberg-Marquardt technique ([115], [123]), for regularization by diagonal terms on the Hessian, has been used in solving the QP. Afterwards, the increments can be obtained from the recursion

$$\Delta_{j+1} = G_j \Delta_j + G_j^\beta \Delta\beta - d_{j+1}.$$

These increments are then scaled by a line-search parameter and added to the current iterate.

5.5 Computation of (Inexact) Derivatives

For the solution of the linear quadratic subproblems of the previous section, we need the matrix-vector products with the Wronskians G_j, G_j^β. These can be carried out "on the fly" (*Internal Numerical Differentiation* (IND) [20, 148]) by solving linear systems of equations with the same linear solver, which is used for the integration of the ODE. The differentiation of the ODE (4.44) with respect to p_g, S_w and T leads to the same matrix, which is used in the formulation of linear systems resulting from the application of a Newton method to the implicit equation defined by, e.g., an implicit Euler method. Therefore, the necessary computations to be done in each integration step for the computation of G_j, G_j^β are $\dim(\beta)$ additional solutions of linear systems after each completed nonlinear Newton solve with the same matrix as used in the last Newton step and with the same linear solver.

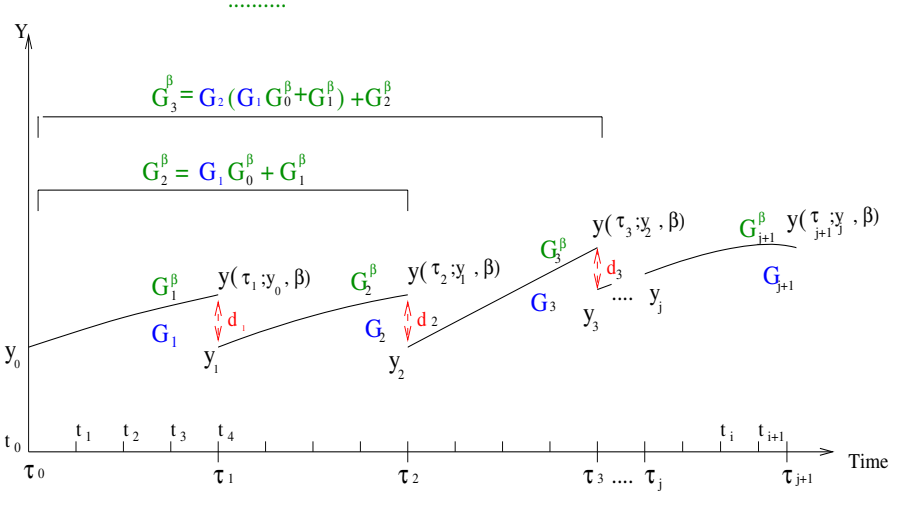


Fig. 5.1 Multiple shooting intervals

In order to clarify this, we consider equation (4.44), which we write in a more abstract and therefore in a more simple form

$$\mathbf{M}(\mathbf{y},\beta)\frac{d\mathbf{y}}{dt} = f(\mathbf{y},\beta), \tag{5.11}$$

within the shooting interval $[\tau_j, \tau_{j+1}]$, where we have the initial condition $\mathbf{y}(\tau_j) = \mathbf{y}_j$ with $\mathbf{y}$ representing the vector of primary variables. Differentiating (5.11) with respect to $\mathbf{y}_j$ and β will result respectively in the ODEs

$$\mathbf{M}(\mathbf{y},\beta)\frac{d}{dt}\left(\frac{\partial \mathbf{y}}{\partial \mathbf{y}_j}\right) = \left(f_{\mathbf{y}} - \mathbf{M}_{\mathbf{y}}\frac{d\mathbf{y}}{dt}\right)\frac{\partial \mathbf{y}}{\partial \mathbf{y}_j}, \tag{5.12}$$

and

$$\mathbf{M}(\mathbf{y},\beta)\frac{d}{dt}\left(\frac{\partial \mathbf{y}}{\partial \beta}\right) = \left(f_{\mathbf{y}} - \mathbf{M}_{\mathbf{y}}\frac{d\mathbf{y}}{dt}\right)\frac{\partial \mathbf{y}}{\partial \beta} - \mathbf{M}_{\beta}\frac{d\mathbf{y}}{dt} + f_{\beta}. \tag{5.13}$$

Let us define

$$G_j(t) := \frac{\partial \bar{\mathbf{y}}(t)}{\partial \mathbf{y}_j} \qquad \text{and} \qquad H_j^{\beta}(t) := \frac{\partial \bar{\mathbf{y}}(t)}{\partial \mathbf{y}_j}\cdot\frac{\partial \mathbf{y}_j}{\partial \beta} + \frac{\partial \bar{\mathbf{y}}(t)}{\partial \beta},$$

where $\bar{\mathbf{y}} = \bar{\mathbf{y}}(\mathbf{y}_j, \beta, t)$ is the solution of the ODE (5.11) at time t with initial values $\mathbf{y}_j$ and $t \in [\tau_j, \tau_{j+1}]$.

Remarks:

- *$G_j(t)$ and $H_j^{\beta}(t)$ satisfy the variational ODE (5.12) and variational ODE (5.13) respectively.*
- *We do not use the variational ODEs because of IND.*

An implicit Euler step computing $\mathbf{y}_j^{i+1}$ $(= \mathbf{y}(t_{i+1}, \mathbf{y}_j^i, \beta))$ as an approximation of $\mathbf{y}(t_{i+1})$ from $\mathbf{y}_j^i = \mathbf{y}(t_i)$ with $t_i, t_{i+1} \in (\tau_j, \tau_{j+1})$ is of the form

$$\mathbf{M}(\mathbf{y}_j^{i+1},\beta)\frac{\mathbf{y}_j^{i+1} - \mathbf{y}_j^i}{h_{i+1}} = f(\mathbf{y}_j^{i+1},\beta), \tag{5.14}$$

with appropriate time-step size h_{i+1}, where the non-linear system (5.14) is solved by Newton's method with each linear system is solved using a bi-conjugate gradient stabilized solver. The principle of internal numerical differentiation is based on a computation of the exact derivative of the approximating discretization scheme (in contrast to computing an approximation of an exact derivative of the nondiscretized solution). Therefore we obtain by differentiating w.r.t. $\mathbf{y}_j$ the recursion

$$\left[\mathbf{M}(\mathbf{y}_j^{i+1},\beta) + h_{i+1}\left(\mathbf{M}_{\mathbf{y}}\cdot\frac{\mathbf{y}_j^{i+1} - \mathbf{y}_j^i}{h_{i+1}} - f_{\mathbf{y}}(\mathbf{y}_j^{i+1},\beta)\right)\right]\frac{\partial \mathbf{y}_j^{i+1}}{\partial \mathbf{y}_j} = \mathbf{M}(\mathbf{y}_j^{i+1},\beta)\frac{\partial \mathbf{y}_j^i}{\partial \mathbf{y}_j},$$

for

$$G_j^{i+1} := \frac{\partial \mathbf{y}\left(t_{i+1}; \mathbf{y}_j^i, \beta\right)}{\partial \mathbf{y}_j},$$

where

$$G_j = G_j^v, \quad \text{and} \quad G_j^0 = I,$$

if v implicit Euler steps are performed in interval $[\tau_j, \tau_{j+1}]$. Thus we obtain the following lemma:

Lemma 3. *The matrix-vector product $G_j d$ is the result of v recursion steps for $\pi^i := G_j^i d$.*

$$\left[\mathbf{M}(\mathbf{y}_j^{i+1}, \beta) + h_{i+1}\left(\mathbf{M}_{\mathbf{y}} \cdot \tfrac{\mathbf{y}_j^{i+1} - \mathbf{y}_j^i}{h_{i+1}} - f_{\mathbf{y}}(\mathbf{y}_j^{i+1}, \beta)\right)\right]\pi^{i+1} = \mathbf{M}(\mathbf{y}_j^{i+1}, \beta)\pi^i,$$

$$\text{with} \quad \pi^0 := (d).$$

Hence, $G_j d = (\pi^v)$. □

Remarks:

- *All matrices mentioned are already assembled in the last Newton step of each implicit Euler step for the nominal trajectory* $\mathbf{y}$.
- *According to IND, the same linear solver, as used for the nominal trajectory, is used for the matrix on the left hand side.*
- *Termination criterion for nominal trajectory is residual less than some given tolerance and for variational trajectory the same number of iterations as that of evaluating nominal trajectory.*

Similarly, we obtain by differentiating (5.14) w.r.t. β the recursion

$$\left[\mathbf{M}(\mathbf{y}_j^{i+1}, \beta) + h_{i+1}\left(\mathbf{M}_{\mathbf{y}} \cdot \frac{\mathbf{y}_j^{i+1} - \mathbf{y}_j^i}{h_{i+1}} - f_{\mathbf{y}}(\mathbf{y}_j^{i+1}, \beta)\right)\right]\frac{\partial \mathbf{y}_j^{i+1}}{\partial \beta} = \mathbf{M}(\mathbf{y}_j^{i+1}, \beta)\frac{\partial \mathbf{y}_j^i}{\partial \beta}$$

$$+h_{i+1}(f_\beta - \mathbf{M}_\beta \cdot \frac{\mathbf{y}_j^{i+1} - \mathbf{y}_j^i}{h_{i+1}}),$$

for

$$G_j^{\beta,i+1} := \frac{\partial \mathbf{y}\left(t_{i+1}; \mathbf{y}_j^i, \beta\right)}{\partial \beta},$$

where

$$G_j^\beta = G_j^{\beta,v}, \quad \text{and} \quad G_j^{\beta,0} = 0,$$

if v implicit Euler steps are performed in interval $[\tau_j, \tau_{j+1}]$. Thus we obtain the following lemma:

Lemma 4. *The matrix-vector product $G_j G_j^\beta$ is the result of v recursion steps for $\pi_\beta^i := G_j G_j^{\beta,i}$.*

$$\left[\mathbf{M}(\mathbf{y}_j^{i+1},\beta)+h_{i+1}\left(\mathbf{M}_{\mathbf{y}}\cdot\frac{\mathbf{y}_j^{i+1}-\mathbf{y}_j^i}{h_{i+1}}-f_{\mathbf{y}}(\mathbf{y}_j^{i+1},\beta)\right)\right]\pi_\beta^{i+1}=\mathbf{M}(\mathbf{y}_j^{i+1},\beta)\pi^i$$
$$+h_{i+1}(f_\beta-\mathbf{M}_\beta\cdot\frac{\mathbf{y}_j^{i+1}-\mathbf{y}_j^i}{h_{i+1}}),\qquad with\quad \pi_\beta^0:=0.$$

Hence, $G_jG_j^\beta=\pi_\beta^v$. □

One should note that the system matrices in this recursion are identical to the ones above - with obvious consequences for the computer implementation. In complete analogy, more complicated products, as in (5.9,5.10), are carried out with identical recursions but different starting data π^0 and π_β^0 for different multiple shooting intervals (Figure 5.1).

5.6 Numerical Results and Discussion

5.6.1 Isothermal Case (Two-Phase flow)

We consider the **McWhorter Problem** (cf.[75], page-258) in the domain $\Omega=[0,2.6]\times[0,1.0]$ and the time interval $(0,1000[s])$ as a test case for verification of our algorithm. This problem deals with computation of instationary displacement process of oil by water, taking into account the capillary effects in a one-dimensional horizontal system (Fig. 5.2). The fluid and solid matrix properties, constitutive relationships and simulation parameters are given as follows.

Boundary conditions:
water saturation S_w = 1.0 [-], oil pressure $p_n = 2.10^5$ [Pa] at x=0

$$\rho_\alpha\mathbf{v}_\alpha\cdot\eta=0\quad\text{at}\quad y=0[m],\quad y=1.0[m]\quad\text{and}\quad x=2.6[m]$$

Initial Condition:
water saturation $S_w(x,0)$ = 0.01 [-] for $x\in\Omega$

We identify the parameter λ, in the Brooks-Corey relationship for capillary pressure and relative permeabilities, as discussed in Chapter 4, and a, the scaling factor

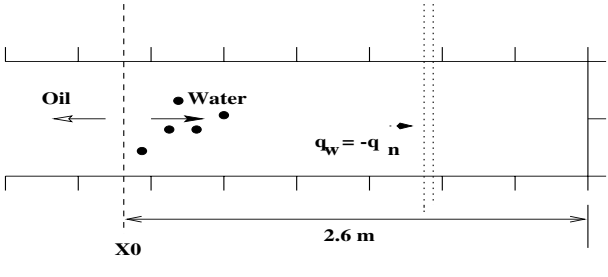


Fig. 5.2 McWhorter Problem (cf. [75])

Table 5.1 Fluid and solid matrix properties and constitutive relationships

	water	oil
(1) fluid properties		
density	1000 [kg/m^3]	1000 [kg/m^3]
dyn. viscosity	0.001 [kg/(ms)]	0.001 [kg/(ms)]
(2) solid matrix properties and constitutive relationships		
abs. permeability k	[m^2]	$a * 10^{-10}$ *a:to be estimated*
porosity ϕ	[-]	0.30
pore size distr. index λ	[-]	*to be estimated*
entry pressure p_d	[Pa]	5000
residual saturation $s_{\alpha r}$	0.00	0.00
rel. permeability $k_r(S_w)$	[-]	Brooks-Corey model
capillary pressure $pc(S_w)$	[Pa]	Brooks-Corey model

in the absolute permeability. Since we do not have actual experimental or measurement data for this type of problem, we have used the 'artificial' data. That is, the capillary pressure values obtained by the numerical computation using $\lambda = 2$ and $a = 1$ have been used as measurement values for this case. These values of parameters were used in the numerical computations in [75] for a comparison with quasi-analytic solutions. Five such measurement points (marked in black in Figure 5.2) and two shooting intervals are used for the computation at times $3.75[s]$ and $31.15[s]$. All the measurement points are taken within the region between the boundary at $x = 0$ and the 'free boundary' (which moves with time) at time 31.15[s]. We use the MUFTE-UG [77] software tool for solving the above mentioned set of partial differential equations on a grid with 1305 grid points. The least-squares problem is solved using reduced Gauss-Newton technique, which is incorporated in MUFTE-UG. Each iteration of the multiple-shooting takes about 2 seconds of CPU time for the above grid size on an SGI machine. The iterations are stopped when $\|(\Delta\lambda, \Delta a)\|_2 < 10^{-3}$. This criterion is reasonable as the regularization parameter becomes smaller with the increments of the parameters.

The solution is independent of the initial guess of the saturation. Table 5.2 presents the results of the computation by using the actual measurements and measurements with a random error of 5% and of 10% with starting values of $\lambda = 1.6$ and $a = 0.5$. As we see, the change in the final value of the parameters (λ, a) is approximately $(0.7\%, 2.7\%)$ and $(1.05\%, 2.1\%)$, respectively. Figure 5.3 presents the results of the saturations of two shooting intervals in different iterations. The defects in the computations are initially large and are reduced in the subsequent iterations, as expected. As we have already mentioned, it is possible to determine the 'measure of goodness' of the parameters or the parameter uncertainty by this method. Since the term

$$\left(\frac{\partial \psi_{ij}}{\partial S_j} g_j^{\beta} + \frac{\partial \psi_{ij}}{\partial \beta}\right)$$

in (5.8) is computed in each iteration, all information necessary for the computation of linearized variances and covariances for the parameters are available if the parameter identification algorithm is converged (and therefore $d_j = 0, \forall j$). If a parameter P_i lies in the interval $P_i \in \left(\hat{P}_i - \delta_i, \hat{P}_i + \delta_i\right)$, then δ_i is determined by using the formula (for details, see [20])

$$\delta_i = ||F_1||_2 \left(\sigma_{ii} \frac{l_1}{l_2} F_{1-\alpha}(l_1, l_2)\right)^{1/2}, \tag{5.15}$$

where $||F_1||_2$ is the 2-norm of the linearized objective function, l_1 is the number of parameters, $l_2 = \dim F_1 - l_1$, σ_{ii} is the variance of the parameter P_i, and $F_{1-\alpha}$ is the $(1-\alpha)$-quantile of the $F-distribution$. Based on this, we compute 95% confidence intervals and display them in Table 5.2, as well.

Table 5.2 Stability of solution for the estimation of λ and a

data set	# iter.	value of λ	value of a
actual data	7	2.000 ± 0	0.999 ± 0
data with 5% error	7	1.987 ± 0.014	0.973 ± 0.030
data with 10% error	7	1.979 ± 0.027	0.979 ± 0.055

5.6.2 Non-isothermal Case (Two-Phase Two-Component Flow)

5.6.2.1 Experimental Setup

For the parameter identification in highly coupled flow and transport processes linked with heat and mass transfer between the phases, the one-dimensional experiments are most practical. We have applied our method to one of these experiments for the simulation and identification. The experiment was carried out in a vertically positioned, sand filled column in the VEGAS research facility at the University of Stuttgart, Germany. The motivation for this experiment was to carry out an experimental program in order to find criteria for the optimization of thermally enhanced soil vapor extraction as an efficient technology for the remediation of NAPL-contaminated unsaturated soils. Small-scale laboratory experiments represent an important part of the experimental program for the investigation of the thermodynamical and hydraulic processes and for the quantification and identification of the dominating processes. The one-dimensional setup of the column experiment facilitates the understanding of the complex coupled processes. In the one-dimensional case, the heat flow is less sensitive to heterogeneities on the microscale, while relative permeability and capillary pressure have a stronger influence on the overall flow. The phase saturation of the water phase was measured by the principle of gamma absorption. The detailed description of the experiments (shown in Figure 5.4) are given in [37].

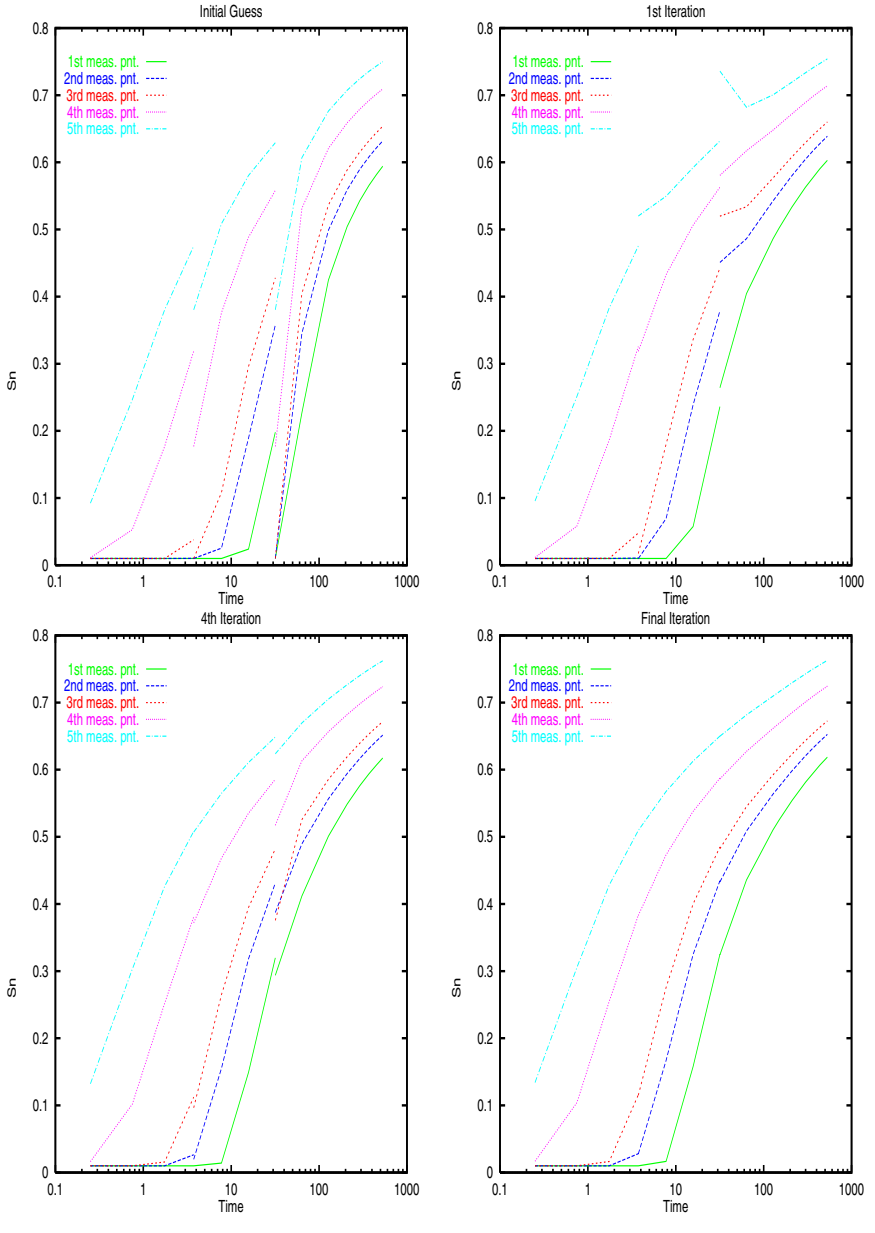


Fig. 5.3 Saturation of non-wetting phase at different iterations

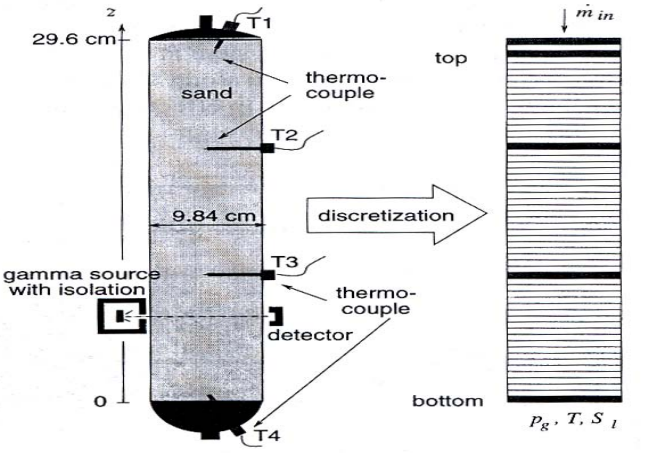


Fig. 5.4 Experimental setup (according to [29])

Table 5.3 Fluid and solid matrix properties and constitutive relationships

(1) fluid properties (water)	
density	1000 [kg/m^3]
dyn. viscosity	0.001 [kg/(ms)]
(2) solid matrix properties (sand)	
Density ρ_s	2650[kg/m^3]
abs. permeability k	$a * 10^{-10}$ m^2
	a:to be estimated
porosity ϕ	0.3678
residual saturation water S_{wr}	0.1
residual saturation air S_{gr}	0.05
heat capacity C_s	840 [J/(kg K)]
heat conductivity $\lambda_{pm}^{S_w=0}$	0.35 [J/(s m K)]
heat conductivity $\lambda_{pm}^{S_w=1}$	1.8 [J/(s m K)]
van Genuchten parameter $1/\alpha$	663.13 [/Pa]
van Genuchten parameter n	*to be estimated*

5.6.2.2 Constitutive Relationships and Initial/Boundary Conditions

The mathematical relationship between capillary pressure and saturation in the sand is used which is according to *van Genuchten* [161], as discussed in Chapter 4, since this produces a good fit to the capillary pressure-saturation measurement data for the coarse sand (cf. [76]). The fluid and solid matrix properties are used as given in Table 5.3. The initial temperature for the experiment was 20^oC. The steam flow was injected at $z = 296mm$ with a *Neumann*-boundary condition. Temperature, pressure,

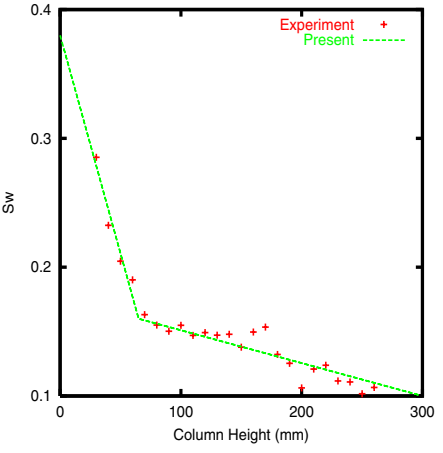


Fig. 5.5 Initial saturation of water

Table 5.4 Stability of the solution for the estimation of n and a

data set	# iter.	value of n	value of a
actual data	17	4.000 ± 0	1.000 ± 0
data with 5% error	17	3.9747 ± 0.0007	1.0293 ± 0.0008
data with 10% error	17	3.9648 ± 0.0014	1.0414 ± 0.0016

Table 5.5 Stability of solution for the estimation of n and a

numer. exp.	parameter	initial guess	estimated value
1	n	4.0	3.264817 ± 0.005329
	a	0.5	0.544405 ± 0.005059
2	n	3.5	3.264943 ± 0.005329
	a	0.75	0.544321 ± 0.005060
3	n	3.0	3.263355 ± 0.005336
	a	1.0	0.545413 ± 0.005044
4	n	2.5	3.263370 ± 0.005336
	a	0.65	0.545404 ± 0.005044
5	n	3.0	3.262310 ± 0.005341
	a	0.25	0.546120 ± 0.005034

and saturation of the gas phase were fixed to a constant value (*Dirichlet*-boundary condition).

We have identified the parameter n in the *van Genuchten* relationship for capillary pressure and relative permeabilities, and a, the *scaling factor* in the absolute permeability. The iterations are stopped when $\|\text{gradient}\|_2 < 10^{-4}$.

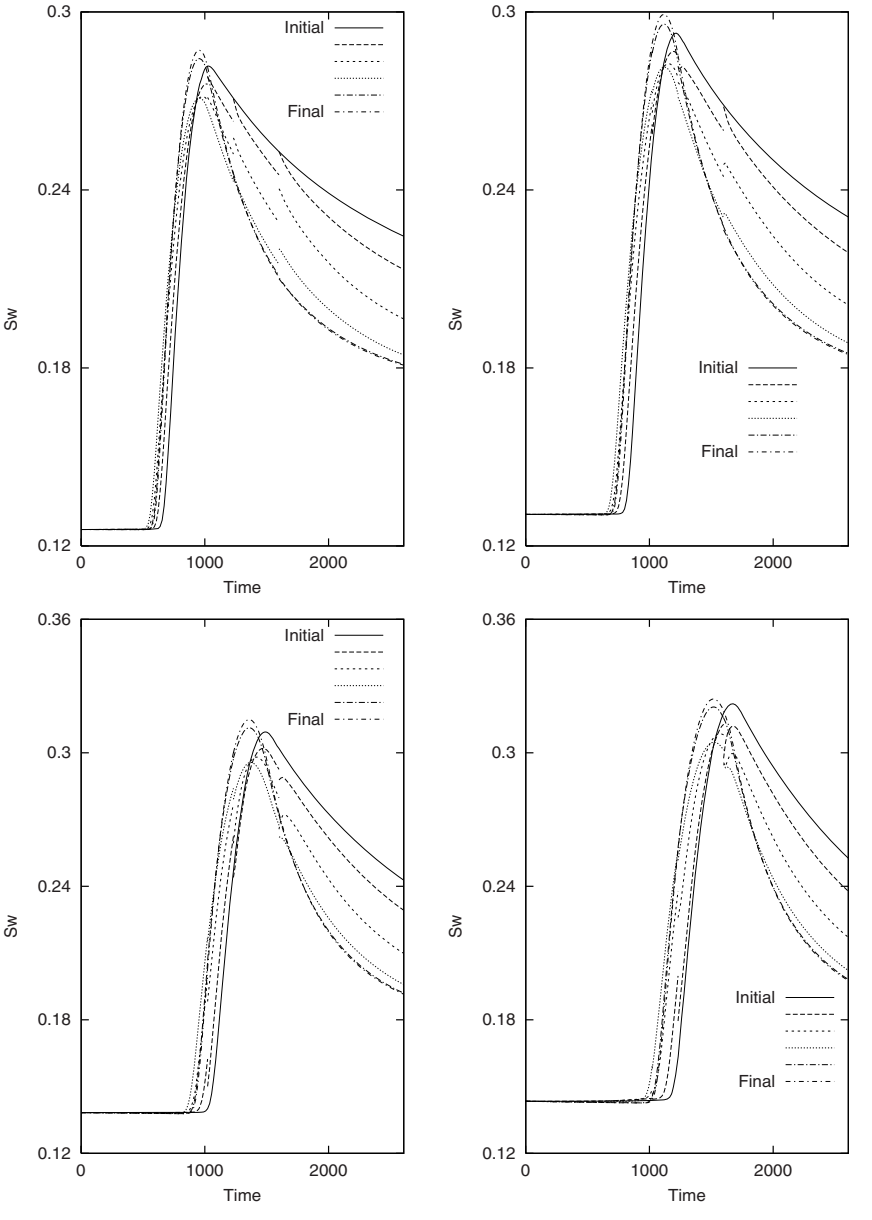


Fig. 5.6 Convergence history at the points Z=200 mm (top left), 180 mm (top right), 150 mm (bottom left) and 130 mm (bottom right)

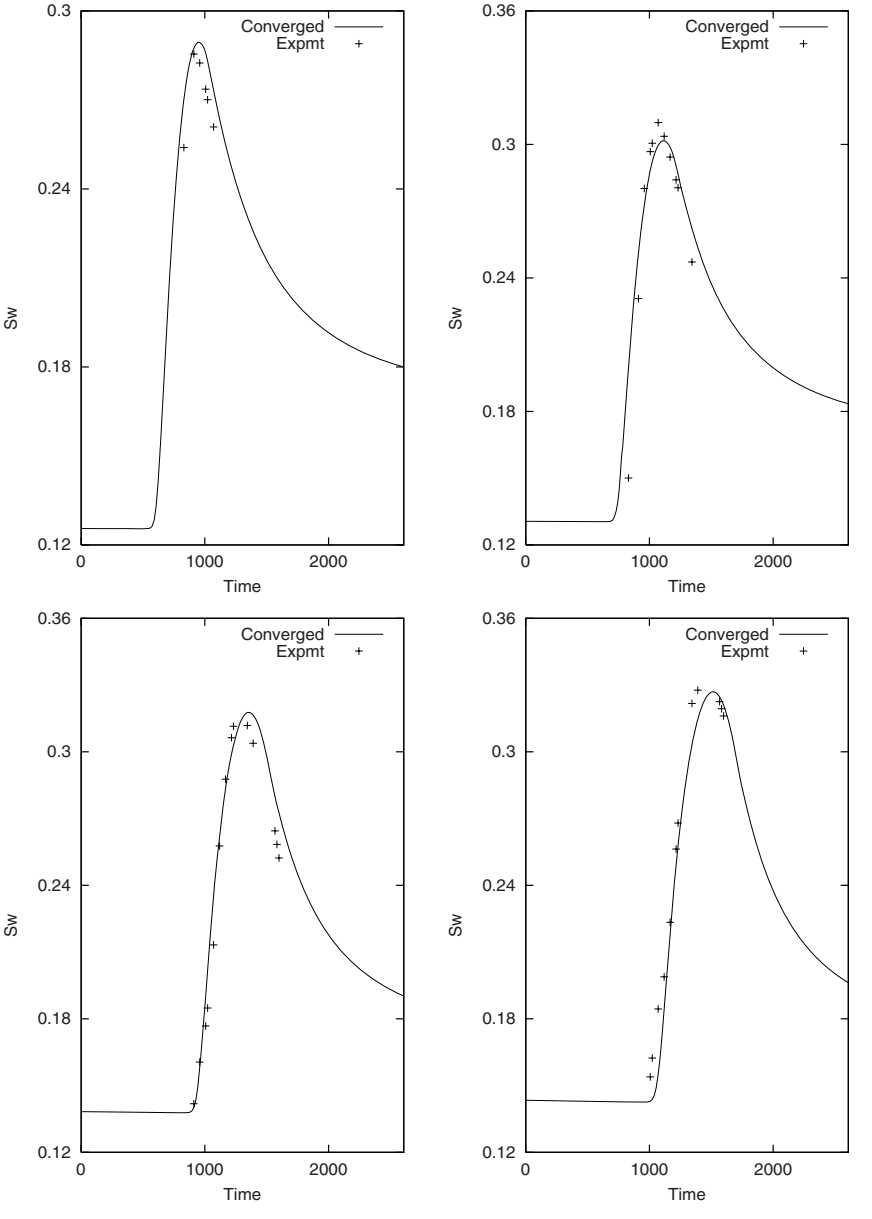


Fig. 5.7 Comparison of computed and used experimental water saturation in the column at Z=200 mm (top left), 180 mm (top right), 150 mm (bottom left) and 130 mm (bottom right)

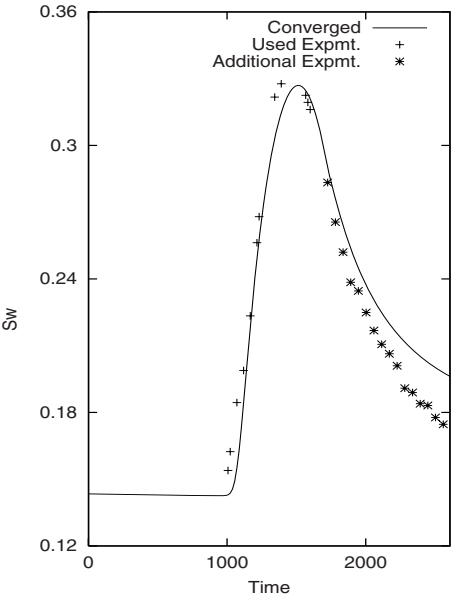


Fig. 5.8 Comparison of computed and experimental water saturation in the column at Z=130 mm

5.6.2.3 Use of Synthetic Data

In order to avoid the ambiguity due to uncertainty in the experimental data that might affect the numerical model, we use synthetic data generated by using the fixed values of the parameters $n = 4.0$ and $a = 1.0$ for the verification of the model. Fifty-four of such data from three shooting intervals at times $\tau = \{766, 1534, 2238\}$ are used. Afterwards, a random error of 5% and of 10% are added to these data and used in the subsequent runs. The results for the initial guess of the parameters $(n = 3.75, a = 0.75)$ are displayed in table 5.4. The change in the final parameter (n, a) values is $(0.63\%, 2.93\%)$ and $(0.88\%, 4.14\%)$, respectively.

5.6.2.4 Use of Experiments: Case 1 (Steam Injection into Wet Coarse Sand from the Top)

This experiment has a constant mass flow injection of 0.18 kg/h steam with a quality of $\approx 90\%$. Forty-three measurements of water saturation from four positions in space, e.g., $z = \{200, 180, 150, 130\}$ mm and at different times $t \in [830, 1600]$ seconds have been used in the three shooting intervals at times $\tau = \{1023, 1231, 1599\}$ seconds. The initial distribution of saturation used in the experiment and in the numerical computation is shown in Figure 5.5. The saturation at the upper part of the column indicates a nearly residual saturation, while there is still a storage capacity of approximately 40% to 70% in the bottom region ($z = 30$mm).

Table 5.6 Result of the Estimation of n and δ

Parameter	Initial Guess	Estimated Value
n	4.0	3.748032 ± 0.0010
δ	1.0	0.901795 ± 0.0016

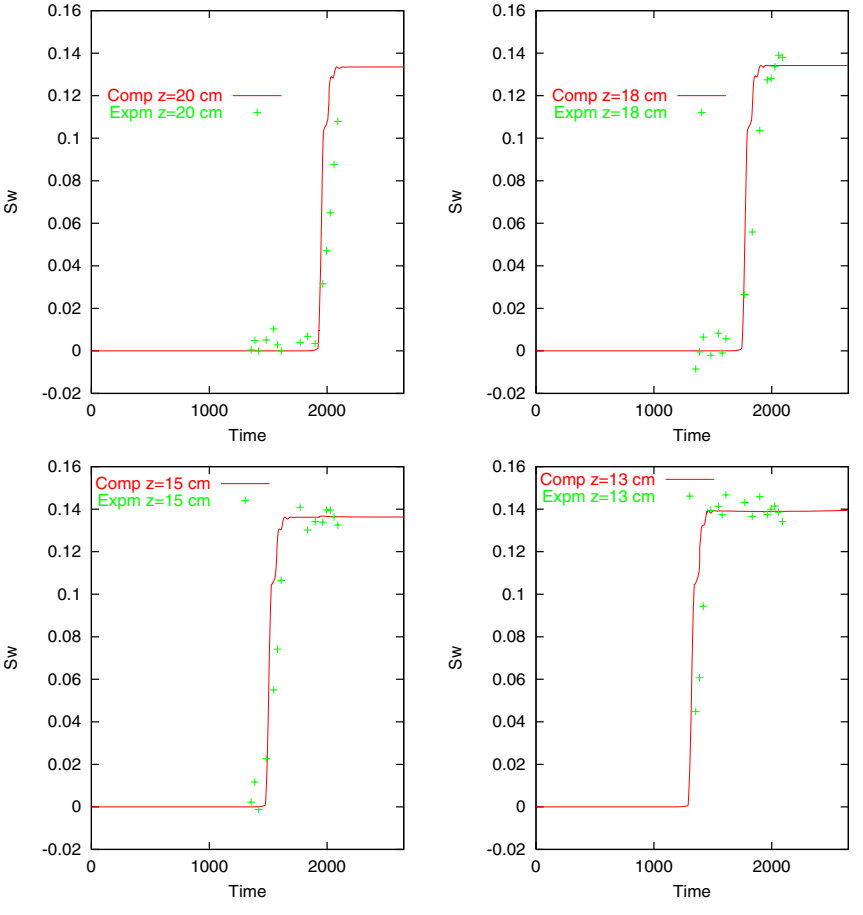


Fig. 5.9 Comparison of computed and experimental water saturation in the column

We have computed the solutions for five different sets of the initial guesses, in order to show that the converged solution does not depend on the initial guess (some kind of stability of the solution, as the theoretical analysis for this nonlinear case is beyond consideration). A 95% confidence interval of the estimated parameters is also computed by using (5.15) based upon the linearized variance and covariance matrix. In all the five cases this is less than (± 0.0054) for the parameter n and

(± 0.0051) for the parameter a. From the results of the computations displayed in Table 5.5 we can say that we have achieved relatively stable estimates of the parameters. The cost of computations is between 30 to 60 direct runs in all the five cases where each direct run requires about 45 seconds of CPU time on an Intel(R) Xeon(TM) 1700MHz machine. Figure 5.6 presents the results of different iterations (for $n = 2.5$ and $a = 0.65$) where the water saturations have been computed at different times. The discontinuities of the solutions at the shooting interval represent the values of the defects (d_i), which are larger during the beginning of the iterations and become zero when we get to the converged solutions of the parameter. Figure 5.7 presents a comparison of the converged numerical solution with the *used* experimental data. As we can see, there is excellent agreement of the solution. Figure 5.8 presents a comparison of the converged solution with the additional experimental data in the downstream (*) at the column height $z = 130$mm. As we see there, the numerical solution does not match very well towards the downstream (approximately, after a time of 2000 seconds). This means that the gravity driven drainage process in the experiment does not match with that of the numerical computation. Several reasons are possible for this mismatch. First, we should keep in mind that the mathematical model uses the same capillary pressure-saturation relationship for both imbibition and drainage, which is not the case in reality, an effect that is called hysteresis. This hypothesis is also offered by similar experiments and inverse computations described in [29]. Another reason can be that the injection rate is not included in the set of estimated parameters here. As shown in [29], the rate of mass flow injected into the sand strongly affects the propagation of the steam/condensation front since it is directly correlated with the amount of thermal energy required for the heating of the sand. [76] describes the numerical simulation of the same experiment with a different set of data without using inverse modeling, but applying a trial-and-error method instead in order to obtain the best fit between the measurements and the simulated data. When using an inverse model with an automatic minimization of the objective function, one must be aware that the minimization algorithm always focuses on regions with steep fronts (peaks), because already small deviations between measurements and simulated data will cause large residuals. This is the reason why the results obtained in this case are difficult to compare with those in [76]. Nevertheless, we strongly recommend to include hysteresis effects into the forward flow model. Following the arguments given above and considering the results of [29] which indicate the effects of hysteresis, it is likely that the results will improve if hysteresis effects are incorporated in the model.

5.6.2.5 Use of Experiments: Case 2 (Steam Injection into Dry Coarse Sand from Bottom)

In this experiment the steam of quality approximately 90% is injected at a constant mass flow rate of 0.18 kg/h into the dry column from bottom. For the numerical experiment, we have taken 47 points of measurements of water saturations from four different positions in space at $z = \{200, 180, 150, 130\}cm$ and at different times in three shooting intervals at $\tau = \{1451, 1931, 2663\}$ seconds. The initial guess of the

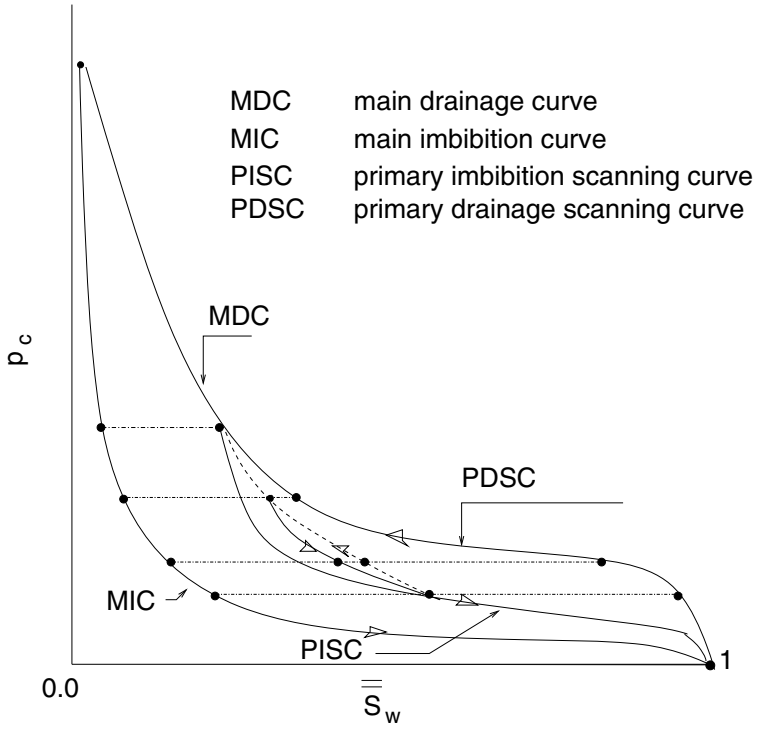


Fig. 5.10 Hysteresis of the capillary pressure–saturation relationship according to [155]

parameters were taken as $n = 4.0$ and $\delta = 1.0$. The solution converged after 14 direct flow solution runs. The final parameter values together with the 95% confidence interval using linearized variance-covariance matrix (as in [20]) are presented in Table 5.6. Figure 5.9 presents a comparison of water saturation of the converged numerical solution with the experiment.

5.7 Conclusions

An algorithm has been developed for parameter identification in multiphase flow through porous media. It employs the reduced Gauss-Newton method to an output least squares minimization problem in an efficient implementation. Special care has been taken concerning the proper formulation of continuity conditions and the computation of derivatives. The numerical studies show that the method is comparatively stable (small changes in experimental data result in similar changes in the solution).

The numerical results indicate that the modeling concept applied within this work lacks certain effects which are due to hysteresis. The methodology built up so far gives reasonable results. It, however, can be improved by incorporating this additional effect. Here, some comments on hysteresis are added.

In multiphase systems one can distinguish between two kinds of displacement processes. A drainage process is given when a non-wetting fluid displaces a wetting fluid. In the case of imbibition, the wetting fluid displaces the non-wetting fluid. The curves of the capillary pressure–saturation and relative permeability–saturation relationships differ between a drainage and an imbibition. This phenomenon is called hysteresis.

Different effects on the microscale are responsible for the hysteretic behavior in porous media multiphase flow, e.g., contact angle, pore geometry (ink-bottle effect), fluid entrapment, see Sheta (1999) [155]. In order to consider hysteresis effects on the macroscale, it is necessary to define the curves shown in Figure 5.10. For example, the main drainage curve (MDC) is valid if the porous medium is initially filled with the wetting phase; thus, the capillary pressure is $p_c = 0$. The non-wetting phase displaces the wetting phase and the capillary pressure increases with a decreasing wetting-phase saturation along the MDC. Accordingly, the other curves must be defined and the history of the drainage/imbibition processes must be monitored. Approaches like that of Parker and Lenhard (1987) [136] often use a semi-empirical scaling of the capillary pressure curves. They introduce additional parameters which must be determined a priori or may be determined by inverse modeling. For future, we plan to investigate if inverse modeling can be used in order to determine such effects quantitatively.

Chapter 6
Simultaneous Pseudo-Time-Stepping for PDE-Model Based Optimization Problems

6.1 Introduction

Time-stepping and pseudo-time-stepping methods are among the oldest methods for the iterative solution of partial differential equations (PDEs). In particular, stationary PDEs are considered as stationary states of a dynamical system past its transient phase. This point of view is most prominent in computation of stationary flow fields, where also the transient phase has a physical interpretation. Furthermore, this approach has the flexibility of producing interpretable results, even if the stationary equation does not possess a solution.

In general, for finite dimensional Banach spaces X, Y and a mapping $F : X \to Y$, the equation

$$F(x) = 0, \tag{6.1}$$

is transformed into the differential equation

$$\frac{d}{dt}\, x = -F(x), \tag{6.2}$$

with suitable initial condition. If $(-F)$ is damping, i.e. $Re\,(\sigma(-\frac{\partial F(x)}{\partial x})) < 0$, where the symbol σ means the spectrum of the Jacobian, then the differential equation reaches a stationary point x_∞, at which

$$0 = \frac{d}{dt}\bigg|_\infty x = -F(x_\infty),$$

so that x_∞ is a solution to equation (6.1).[1]

The differential equation (6.2) usually is solved numerically by employing an explicit "time"-integration. In the simplest example (i.e. explicit Euler) this leads to the iteration

$$x^{m+1} = x^m - \Delta t\, F(x^m),$$

[1] Materials presented in this chapter can also be found in [67], reprinted with kind permission of Springer Science and Business Media.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 65–80.
springerlink.com

which is the Richardson-iteration for linear operators F. This observation, that there is a correspondence between general iterative methods and time-stepping schemes, holds in most cases, which is the reason why (pseudo-) time-stepping is called a superfluous concept in (Hackbusch [54]). In this line of thought, preconditioning operators P for speeding up the convergence as ,

$$x^{m+1} = x^m - \Delta t\, P F(x^m),$$

are interpreted as a change of the differential equation

$$\dot{x} = -PF(x),$$

resulting in the same stationary limit. Pseudo-time-stepping for PDE using implicit schemes is discussed in [102].

Also for optimization problems, use of pseudo-time-stepping is quite long. It dates back to the works of Hadamard [55] and Courant [32] where they call it as method of gradients. The idea arose in the study of variational partial differential equations. Each of these equations has a function $f : X \to \mathbb{R}$ (also called a functional) such that a solution of the equation is a minimizer of f. The method of gradients starts with an initial point $x_0 \in X$ and seeks to find a minimizer of f by following a curve ϕ defined by the ordinary differential equation

$$\begin{aligned} \dot{\phi}(t) &= -\nabla f(\phi(t)), \\ \phi(0) &= x_0, \end{aligned}$$

where ∇f is the gradient of f. This method for optimization problem is called gradient flow method, see for example [14] and the references therein. A second order equation can also be considered for such problems, see for example in [126].

Although, the interpretation of iterative techniques in the form of differential equations seems artificial from a mathematical point of view, it is very common in the engineering literature. Therefore, often in collaborative optimization efforts one is forced to formulate optimization algorithms within the framework of (pseudo-) time-stepping methods for the purpose of a consistent overall implementation. This technique, also called pseudo-transient continuation in [33], uses the instationary flow equations and ODE-integrators applied to it as a paradigm to derive semi-iterative technique as shown in [54]. The distinction to real instationary computations lies in the fact that the time-step selection is not geared towards most accurate approximation of a time history but rather towards leading to a stationary point as fast as possible, i.e., stability is more important than consistency. Although, the term 'pseudo-time-stepping' is, in this respect, somewhat misleading, we will use this paradigm in the sequel, because on the one hand it is understood quite well among scientists working closely with engineering applications, and on the other hand, the more correct substitute term 'semi-iterative method' refers to a much wider class of methods than we envision.

6.2 The Optimization Problem and Pseudo-unsteady Formulation of the KKT Conditions

The optimization problem that we are dealing with in this study is of the class of optimal control problems including also the sub-class of shape design problems. Pioneering theoretical works on the methodology for solving such problems have been presented in [120, 139, 138, 140]. These problems can be written in abstract form as

$$\begin{aligned} &\min I(w,q) \\ &\text{s. t.} \quad \mathbf{c}(w,q) = 0, \end{aligned} \tag{6.3}$$

where $(w,q) \in W \times Q$ (W, Q are appropriate Hilbert spaces), $I : W \times Q \to \mathbb{R}$ and $c : W \times Q \to Y$ are twice Frechet-differentiable (with Y an appropriate Hilbert space). The Jacobian, $\mathbf{J} = \dfrac{\partial \mathbf{c}}{\partial w}$, is assumed to be invertible. The equation $\mathbf{c}(w,q) = 0$ represents a differential model equation together with its boundary conditions, w is the vector of dependent variables and q is the vector of design variables. In what follows, we denote by the notation $()^\top$ the transpose of the vector or the operator. In an optimal control setting, adjoint variables are from the dual space and the notation should reflect this. However, since ultimately we are dealing with discretized magnitudes, we do not distinguish between finite and infinite dimensional adjoints and use always the same notation.

The necessary optimality conditions can be formulated using the Lagrangian functional

$$L(w,q,\lambda) = I(w,q) - \lambda^\top \mathbf{c}(w,q), \tag{6.4}$$

where λ is the Lagrange multiplier or the adjoint variable from the dual Hilbert space and the second term in the right hand side is a duality pairing. If $\hat{z} = (\hat{w}, \hat{q})$ is a minimizer, then there exists a $\hat{\lambda}$ such that

$$\nabla_z L(\hat{z}, \hat{\lambda}) = \nabla_z I(\hat{z}) - \hat{\lambda}^\top \nabla_z \mathbf{c}(\hat{z}) = 0. \tag{6.5}$$

Hence, the necessary optimality conditions, known as the Karush-Kuhn-Tucker (KKT) conditions, are

$$\mathbf{c}(w,q) = 0, \qquad (\textit{State equation}) \tag{6.6a}$$

$$\nabla_w L(w,q,\lambda) = 0, \qquad (\textit{Costate equation}) \tag{6.6b}$$

$$\nabla_q L(w,q,\lambda) = 0. \qquad (\textit{Design equation}) \tag{6.6c}$$

It is to be noted that the statement of the optimality conditions is formal for the target problems, both in the function space setting as well as for the discretized problems in this problem class. In general, derivatives like $\dfrac{\partial \mathbf{c}}{\partial w}$ cannot be guaranteed to exist, specially for problems in aerodynamic applications. However, the total derivatives of the objective function with respect to the design variables exist typically and is all what is necessary for computations. Also, for the derivation of the costate or adjoint equation formulation of the Lagrangian is not necessary. As shown in

Chapter 12, one can derive the adjoint equations using 'direct' or 'engineering' approach as well.

Gradient methods, which are widely used in many practical applications, involve the solution of the state and the costate equations at each update of the design variables. These methods act only in the design space and assume that the state and the adjoint equations are solved exactly. Thus, they can be viewed as an explicit Euler approximation to the following evolution differential algebraic equation

$$\mathbf{c}(w,q) = 0, \tag{6.7a}$$

$$\nabla_w L(w,q,\lambda) = 0, \tag{6.7b}$$

$$\frac{dq}{dt} + \nabla_q L(w,q,\lambda) = 0. \tag{6.7c}$$

The disadvantage of this method is its high computational cost because state and costate equations have to be solved quite accurately in each iteration step. This approach with less accurate state and costate solution has been performed in Iollo et. al. [86].

In [156], S. Ta'asan proposed another approach in which pseudo-time embedding is suggested for the state and the costate equations and the design equation is solved as an additional boundary condition, specially for boundary control problems. This means, to find a steady state solution of the system

$$\frac{dw}{dt} + \mathbf{c}(w,q) = 0, \tag{6.8a}$$

$$\frac{d\lambda}{dt} + \nabla_w L(w,q,\lambda) = 0, \tag{6.8b}$$

$$\nabla_q L(w,q,\lambda) = 0. \tag{6.8c}$$

This is still a system of differential algebraic equations, where one has to provide some means to solve the design equation alone. We supersede that formulation by constructing a system of only differential equations. An added advantage of the new strategy is that it automatically incorporates globalization in the control or design space.

We propose a new method, simultaneous pseudo-time-stepping, for solving the above problem (6.6). As explained above, there is a strong correlation between iterative methods and pseudo-time stepping which is now going to be exploited for the construction of a time-stepping method in the spirit of reduced SQP-methods. That is, we are looking for the steady state solution of the pseudo-time embedded evolution equations

$$\frac{dw}{dt} + \mathbf{c}(w,q) = 0, \tag{6.9a}$$

$$\frac{d\lambda}{dt} + \nabla_w L(w,q,\lambda) = 0, \tag{6.9b}$$

$$\frac{dq}{dt} + \nabla_q L(w,q,\lambda) = 0. \tag{6.9c}$$

However, this pseudo-time embedded system, after semi-discretization, results in usually a stiff system of ODEs. Explicit time-stepping schemes (which are used in most of the applications in this problem class)may converge very slowly or may even diverge. In order to accelerate convergence, this system needs preconditioning. The preconditioner that we consider stems from the reduced SQP methods (see next section), whose mathematical background is well studied. We have implemented this proposed method for a boundary control problem in this chapter. The number of iterations required for the full optimization problem is almost double the analysis problem. This means a drastic reduction of the computational cost compared to the 'black-box' gradient methods. In subsequent chapters, we discuss the realistic design tasks in flow problems.

6.3 Reduced SQP Methods

For the solution of problem (6.3), we recall a straight forward reduced SQP (rSQP) method. A detailed discussion of this approach can be found in [21, 150, 151]. Industrial applications of the rSQP concepts are discussed in [19, 122, 153, 154]. Here we sketch only the idea.

Reduced SQP methods are most advantageous in cases where the number of degrees of freedom (here the parameters) is small compared to the number of state variables (cf. [51]). The variable steps in each iteration can be considered as linear combinations of steps towards optimality and steps towards feasibility of the constraints. The constraints are linearized by a Taylor expansion up to first order terms, so that all steps towards optimality lie in the tangent space of $\mathbf{c}$ of the current approximation (w,q):

$$\mathbf{c}(w,q) + J(w,q)\,\Delta w + \frac{\partial \mathbf{c}}{\partial q}(w,q)\,\Delta q = 0.$$

The optimization problem is projected to this tangent space and approximated by a quadratic programming problem with the projected Hessian of the Lagrangian given by (6.4). In this formulation the algorithm reads as:

Algorithm 1: *The rSQP method.*
(0) Set $k := 0$*; start at some initial guess* w_0*,* q_0*.*
(1) Compute the adjoint variables from the linear system
$J^\top(w_k,q_k)\,\lambda_{k+1} := \nabla_w I(w_k,q_k)$*;*
compute the reduced gradient
$$\gamma_k := \nabla_q I(w_k,q_k) - \left(\frac{\partial \mathbf{c}}{\partial q}(w_k,q_k)\right)^\top \lambda_{k+1};$$
determine some approximation B_k *of the projected Hessian of the Lagrangian.*
(2) solve $B_k \Delta q_k = -\gamma_k$*.*

(3) compute step on w from the linear system

$$J(w_k,q_k)\,\Delta w_k := -\frac{\partial \mathbf{c}}{\partial q}(w_k,q_k)\,\Delta q_k - c(w_k,q_k).$$

(4) Set $w_{k+1} := w_k + \Delta w_k$, $q_{k+1} := q_k + \Delta q_k$.
(5) Set $k := k+1$; go to (1) until convergence.

The computationally expensive operation of computing the exact projected Hessian typically is avoided by using appropriate update formulas. It can be proven that under mild conditions the reduced SQP method described by Algorithm 1 with the update formulas for the reduced Hessian shows super-linear local convergence properties (s. [150]). This has also been shown in a Hilbert space setting in [112].

In Algorithm 1, it is necessary to invert the Jacobian J of the constraints. In many cases, that is not a viable approach. Therefore, we employ an inexact reduced SQP method by substituting J with an approximate operator A as considered in [151]. Although it is not inverted, the Jacobian is still used for computation of the correct adjoint variables and the correct state increments in the sense of defect correcting iterations:

Algorithm 2: *The rSQP method with an approximate Jacobian.*
(0) Set $k := 0$; start at some initial guess w_0, q_0.
(1) Compute the increment of the adjoint variables from the linear system

$$A^\top(w_k,q_k)\,\Delta\lambda_k := \nabla_w I(w_k,q_k) - J^\top(w_k,q_k)\,\lambda_k;$$

compute the reduced gradient

$$\gamma_k := \nabla_q I(w_k,q_k) - \left(\frac{\partial \mathbf{c}}{\partial q}(w_k,q_k)\right)^\top (\lambda_k + \Delta\lambda_k);$$

determine some approximation B_k of the projected Hessian of the Lagrangian.
(2) solve $B_k \Delta q_k = -\gamma_k$.
(3) compute step on w from the linear system

$$A(w_k,q_k)\,\Delta w_k := -\frac{\partial \mathbf{c}}{\partial q}(w_k,q_k)\,\Delta q_k - \mathbf{c}(w_k,q_k).$$

(4) Set $w_{k+1} := w_k + \Delta w_k$, $q_{k+1} := q_k + \Delta q_k$ and $\lambda_{k+1} = \lambda_k + \Delta\lambda_k$.
(5) Set $k := k+1$; go to (1) until convergence.

A step of this method can also be interpreted as an approximate Newton step for the necessary conditions of extremum for problem (6.3), since the updates of the variables are computed according to a linear system

$$\begin{pmatrix} 0 & 0 & A^\top \\ 0 & B & \left(\frac{\partial \mathbf{c}}{\partial q}\right)^\top \\ A & \frac{\partial \mathbf{c}}{\partial q} & 0 \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta q \\ \Delta\lambda \end{pmatrix} = \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -\mathbf{c} \end{pmatrix}. \tag{6.10}$$

This is the basic formulation of inexact reduced SQP methods that we are using in the subsequent sections.

6.4 Pseudo-Time-Stepping for Optimization Problems

Corresponding to the presentation in section 6.1, the necessary conditions (6.5) for optimization can be considered as an overall nonlinear equation to be solved, i.e.

$$F(w,\lambda,q) = \begin{pmatrix} \nabla_w L \\ \nabla_q L \\ \mathbf{c} \end{pmatrix} (w,\lambda,q) = 0.$$

In this way, it would be possible to perform an optimization strategy in a consistent way with the time stepping method, which is often used for the solution of the design equation alone.

In order to accelerate convergence, this system needs some preconditioning. An ideal preconditioner would be the inverse of the full approximate KKT-matrix, i.e.

$$P_{\text{ideal}} = \begin{bmatrix} H_{ww}(w,q) & H_{wq}(w,q) & A^\top(w,q) \\ H_{qw}(w,q) & H_{qq}(w,q) & \left(\dfrac{\partial \mathbf{c}(w,q)}{\partial q}\right)^\top \\ A(w,q) & \dfrac{\partial \mathbf{c}(w,q)}{\partial q} & 0 \end{bmatrix}^{-1}.$$

However, the solution of linear systems with the full KKT-matrix typically requires an iterative process by its own and is not feasible from an implementation point of view, imagining that one starts out with a time-stepping scheme for the state equations.

In our implementations, we use the inverse of the matrix in equation (6.10) as a preconditioner P for the time-stepping process. The pseudo-time embedded system of ODE that we consider reads as, after reordering of variables,

$$\begin{pmatrix} \dot\lambda(t) \\ \dot q(t) \\ \dot w(t) \end{pmatrix} = \begin{bmatrix} A^\top(w(t),q(t)) & 0 & 0 \\ \left(\dfrac{\partial \mathbf{c}(w(t),q(t))}{\partial q}\right)^\top & B(w(t),q(t)) & 0 \\ 0 & \dfrac{\partial \mathbf{c}(w(t),q(t))}{\partial q} & A(w(t),q(t)) \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -\mathbf{c} \end{pmatrix}.$$

This seems natural since equation (6.10) can be considered as an explicit Euler discretization for the corresponding time-stepping that we envision. The preconditioner employed is similar to the preconditioners for KKT-systems discussed in [12, 11] in the context of Krylov subspace methods and in [17] in the context of Lagrange-Newton-Krylov-Schur methods. Convergence of iterations with KKT-preconditioners are discussed in [12, 87]. Within the inexact reduced SQP-preconditioner, we have to look for an appropriate approximation of the reduced Hessian. In particular, when dealing with partial differential equations constituting the state equations, often the reduced Hessian can be expressed as a pseudo-differential operator, the symbol of which can be computed and exploited for preconditioning purposes. This is done for the considered test problem in the next

section. In [5, 6] use of the symbol of the Hessian as preconditioner is discussed in different form.

6.5 Application to a Model Problem

Here, we investigate a model problem of academic interest. We apply the new method to the following optimal control problem (Neumann to Dirichlet map) defined by ([120], Chap.II, Sec.5.2)

$$\min \int_{y=1} \left(\frac{\partial \phi}{\partial \eta} - g(x) \right)^2 dx + \sigma \int_{y=1} \left(q^2(x) + (\frac{dq}{dx})^2 \right) dx \tag{6.11}$$

$$\begin{aligned} \text{s.t.} \qquad \Delta \phi &= 0 \qquad \text{in} \quad \Omega, \\ \phi &= q(x) \qquad \text{on} \quad y = 1, \\ \phi &= \phi_0 \qquad \text{on} \quad y = 0, \end{aligned} \tag{6.12}$$

where σ is a fixed nonnegative parameter, $g(x)$ is a given function, η denotes the outer normal to the boundary and $\Omega = \{0 < x < 1; 0 < y < 1\}$ (Figure 6.1). In contrast to [120], we choose $q \in H^1([0,1])$, otherwise the norm in the tracking part of the objective should be the H^{-1}-norm, which is numerically less amenable. Periodicity is assumed in the x direction. q is the control function defined on the boundary $\partial\Omega = \{0 \le x \le 1; y = 1\}$. The necessary conditions (6.5) lead to the definition of costate variables which satisfy the costate equations given by

$$\begin{aligned} \Delta \lambda &= 0 \qquad \text{in} \quad \Omega, \\ \lambda + 2 \left(\frac{\partial \phi}{\partial \eta} - g(x) \right) &= 0 \qquad \text{on} \quad y = 1, \\ \lambda &= 0 \qquad \text{on} \quad y = 0. \end{aligned} \tag{6.13}$$

They also satisfy the periodic boundary conditions in the x direction. The control equation is given by

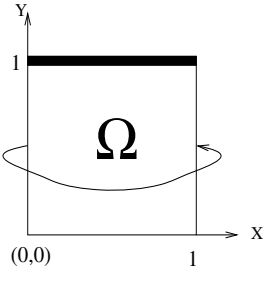


Fig. 6.1 Domain of the model problem

$$2\sigma\left(q-\frac{d^2q}{dx^2}\right)-\frac{\partial\lambda}{\partial\eta}=0 \qquad \text{on} \quad y=1. \tag{6.14}$$

As explained in the previous section, instead of solving the above set of equations iteratively, we are looking for the stationary solution of the following time evolution equations resulting in from the pseudo-time embedding

$$\begin{aligned}
\frac{d}{dt}\phi-\Delta\phi &= 0 && \text{in} \quad \Omega,\\
\frac{d}{dt}\phi+\phi-q(x) &= 0 && \text{on} \quad y=1,\\
\frac{d}{dt}\phi+\phi-\phi_0 &= 0 && \text{on} \quad y=0,\\
\frac{d}{dt}\lambda-\Delta\lambda &= 0 && \text{in} \quad \Omega,\\
\frac{d}{dt}\lambda+\lambda+2\left(\frac{\partial\phi}{\partial\eta}-g(x)\right) &= 0 && \text{on} \quad y=1,\\
\frac{d}{dt}\lambda+\lambda &= 0 && \text{on} \quad y=0,\\
\frac{d}{dt}q+2\sigma\left(q-\frac{d^2q}{dx^2}\right)-\frac{\partial\lambda}{\partial\eta} &= 0 && \text{on} \quad y=1.
\end{aligned} \tag{6.15}$$

These equations are to be solved in time with suitable initial conditions. The PDEs in (6.15) are semi-discretized in space by finite-differences. The Laplacians are approximated using second order symmetric difference formulas. The semi-discretization results in a system of ODEs. The preconditioned ODEs are solved using a Runge-Kutta-Fehlberg (fourth-fifth order) time-stepping scheme. We construct the reduced Hessian in the preconditioner as explained in the next sections.

6.6 Analysis of the Hessian

We use the concept of pseudo-differential operators and their symbol which are generalizations of differential operators and developed from the study of singular integral operators used for inverting differential operators. For detailed theoretical investigations on pseudo-differential operators, we refer to [84, 105]. The action of a pseudo-differential operator defined by its symbol on functions in some domain can be defined via Fourier analysis (see, for example, [5, 6]). Such analysis is carried out to determine the symbol of the Hessian of the objective function which is defined on the boundary.

In the following, we study the effect of some perturbation in the control variable at the boundary $y = 1$ to the solution in the half plane $y < 1$. For that, we define the perturbed variables $\bar{\phi}, \bar{\lambda}$ and $\bar{q}$ corresponding to the state, costate and design variables. Due to linearity of the interior equations in (6.12) and (6.13), the perturbed

variables also satisfy the same equations as the unperturbed variables in the interior. The conditions on the boundary $y = 1$ are

$$\bar{\phi} = \bar{q} \quad \text{on} \quad y = 1 \tag{6.16}$$

$$\bar{\lambda} + 2\frac{\partial \bar{\phi}}{\partial \eta} = 0 \quad \text{on} \quad y = 1 \tag{6.17}$$

$$2\sigma\left(\bar{q} - \frac{d^2\bar{q}}{dx^2}\right) - \frac{\partial \bar{\lambda}}{\partial \eta} = 0 \quad \text{on} \quad y = 1 \tag{6.18}$$

The action of the reduced Hessian on the perturbation $\bar{q}$ in the form of a Fourier mode $\bar{q} = e^{i\omega x}$ is now represented as

$$B : e^{i\omega x} \longmapsto \Sigma(\omega, t)e^{i\omega x},$$

where $\Sigma(\omega, t)$ denotes the symbol of the operator B. If this representation is valid and $\Sigma(\omega, t)$ is a polynomial in the frequency ω, B can be represented by a differential operator. This is shown in the following lemma.

Lemma 5. *The symbol of the reduced Hessian of (6.11) subject to (6.12), (6.13) and (6.14) is* $\Sigma(\omega) = 2(\sigma + (\sigma + 1)|\omega|^2)$.

Proof. Suppose $\bar{q} = e^{i\omega x}$ is a Fourier component. Then $\bar{\phi}$ and $\bar{\lambda}$ can be expressed in terms of Fourier modes as

$$\bar{\phi} = \xi e^{i\omega x} e^{r(\omega)y} \qquad \text{and} \qquad \bar{\lambda} = \zeta e^{i\omega x} e^{r(\omega)y}.$$

Substituting these expressions at the interior (Laplace) equations results in $\omega^2 = r^2(\omega)$. Since we are looking for a stable solution as $y \to -\infty$, we take $r(\omega) = |\omega|$. Substitution in (6.16) yields

$$\xi e^{i\omega x} e^{|\omega|} = e^{i\omega x},$$

i.e. $\xi = e^{-|\omega|}$. Similarly, substitution in (6.17) yields

$$\zeta e^{i\omega x} e^{|\omega|} + 2|\omega| e^{-|\omega|} e^{i\omega x} e^{|\omega|} = 0,$$

i.e. $\zeta = -2|\omega| e^{-|\omega|}$. Finally, substitution in (6.18) will result

$$2\sigma(1 + |\omega|^2)e^{i\omega x} + 2|\omega|^2 e^{-|\omega|} e^{i\omega x} e^{|\omega|} = 0.$$

From this we obtain

$$2\left(\sigma + (\sigma + 1)|\omega|^2\right) = 0. \tag{6.19}$$

Hence, the symbol of the Hessian is $\Sigma(\omega) = 2\left(\sigma + (\sigma + 1)|\omega|^2\right)$.

The symbol deduced above represents the differential operator

$$B = 2(\sigma I - (\sigma+1)\frac{\partial^2}{\partial x^2}). \tag{6.20}$$

Since we need the inverse of the operator B in the actual computation, we use $(B)^{-1} = 1/[2(\sigma + \frac{4(\sigma+1)}{\delta x^2})]$, where δx is the mesh size, as an approximation. For the numerical implementation, also the matrix A of equation (6.10) has to be specified. There we use only the diagonal part of the discretized Laplacian.

6.7 Numerical Implementation

For problem (6.15), we can write the preconditioned system separating the interior (denoted by subscript i) and the boundary (denoted by subscript b) variables (due to ease of implementation) in the form as

$$\begin{pmatrix} \dot{\lambda}_i \\ \dot{\lambda}_b \\ \dot{q} \\ \dot{\phi}_b \\ \dot{\phi}_i \end{pmatrix} = \begin{bmatrix} A_{ii}^\top & A_{ib}^\top & 0 & 0 & 0 \\ A_{bi}^\top & A_{bb}^\top & 0 & 0 & 0 \\ \left(\frac{\partial c}{\partial q}\right)_i^\top & \left(\frac{\partial c}{\partial q}\right)_b^\top & B & 0 & 0 \\ 0 & 0 & \left(\frac{\partial c}{\partial q}\right)_b & A_{bb} & A_{bi} \\ 0 & 0 & \left(\frac{\partial c}{\partial q}\right)_i & A_{ib} & A_{ii} \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_{\phi_i} L \\ -\nabla_{\phi_b} L \\ -\nabla_q L \\ -c_b \\ -c_i \end{pmatrix}.$$

We use a further approximation of the preconditioner as follows. We replace the block matrices

$$\begin{bmatrix} A_{ii} & A_{ib} \\ A_{bi} & A_{bb} \end{bmatrix}$$

by

$$\begin{bmatrix} D_i & 0 \\ 0 & D_b \end{bmatrix}$$

where D_i, D_b are only the diagonal part of A_{ii} and A_{bb}. Because of the nature of the Dirichlet boundary conditions at $y = 1$, we have $\left(\frac{\partial c}{\partial q}\right)_i = 0$ and $\left(\frac{\partial c}{\partial q}\right)_b = -I$ where I means the identity matrix. Then the final preconditioned system reads as

$$\begin{pmatrix} \dot{\lambda}_i \\ \dot{\lambda}_b \\ \dot{q} \\ \dot{\phi}_b \\ \dot{\phi}_i \end{pmatrix} = \begin{bmatrix} D_i & 0 & 0 & 0 & 0 \\ 0 & D_b & 0 & 0 & 0 \\ 0 & -I & B & 0 & 0 \\ 0 & 0 & -I & D_b & 0 \\ 0 & 0 & 0 & 0 & D_i \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_{\phi_i} L \\ -\nabla_{\phi_b} L \\ -\nabla_q L \\ -c_b \\ -c_i \end{pmatrix}.$$

Lemma 6. *The preconditioner is invertible and the inverse is given by*

$$\begin{bmatrix} D_i^{-1} & 0 & 0 & 0 & 0 \\ 0 & D_b^{-1} & 0 & 0 & 0 \\ 0 & B^{-1}D_b^{-1} & B^{-1} & 0 & 0 \\ 0 & D_b^{-1}B^{-1}D_b^{-1} & D_b^{-1}B^{-1} & D_b^{-1} & 0 \\ 0 & 0 & 0 & 0 & D_i^{-1} \end{bmatrix}.$$

Proof. The preconditioner is a lower block-diagonal matrix. Each of the blocks are diagonal matrices except for the block B, whose approximate inverse is given in the preceeding section. Hence the preconditioner is invertible. For the inverse, multiplying both from left and right will yield the result.

Hence the final preconditioned system reads as

$$\begin{pmatrix} \dot{\lambda}_i \\ \dot{\lambda}_b \\ \dot{q} \\ \dot{\phi}_b \\ \dot{\phi}_i \end{pmatrix} = K \begin{pmatrix} -\nabla_{\phi_i} L \\ -\nabla_{\phi_b} L \\ -\nabla_q L \\ -c_b \\ -c_i \end{pmatrix}, \tag{6.21}$$

where

$$K = \begin{bmatrix} D_i^{-1} & 0 & 0 & 0 & 0 \\ 0 & D_b^{-1} & 0 & 0 & 0 \\ 0 & B^{-1}D_b^{-1} & B^{-1} & 0 & 0 \\ 0 & D_b^{-1}B^{-1}D_b^{-1} & D_b^{-1}B^{-1} & D_b^{-1} & 0 \\ 0 & 0 & 0 & 0 & D_i^{-1} \end{bmatrix}.$$

6.8 Results and Discussion

Numerical computations are carried out on a (20×20) grid in space for this model problem. For the time integration, we use a standard Runge-Kutta-Fehlberg integration scheme with local stepsize tolerance of TOL=10E-8. The initial values of ϕ, λ and q are taken constantly as 1.0, 1.0 and 2.0 respectively. The value of σ is taken to be 0.001. As stopping criterion we use the *maximum* norm of the state variable residual to be $< 10^{-2}$. Figure 6.2 presents the residual of state, costate and control variables for the computations with the preconditioner P. The convergence criterion is satisfied for the preconditioned system within 160 Runge-Kutta-Fehlberg steps.

We also have carried out the numerical computation of the original pseudo-time embedded system (without the preconditioner) using the same numerical method. The convergence criterion is not satisfied in this case even after five thousand Runge-Kutta-Fehlberg iterations. This is due to the fact that the stability criterion restricts the code to use very small time steps. Figure 6.3 presents the comparison of the adaptive time steps used by the code for the original and the

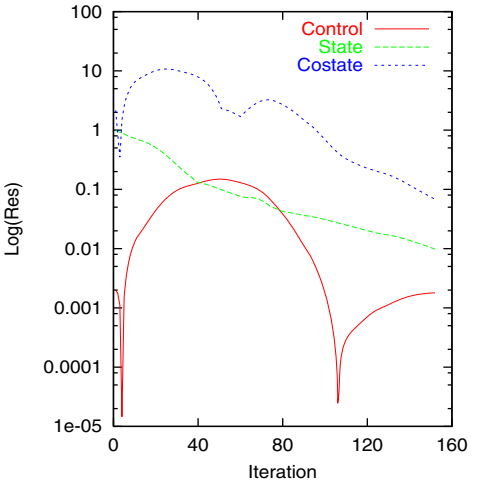


Fig. 6.2 Convergence history of pseudo-time method with preconditioner

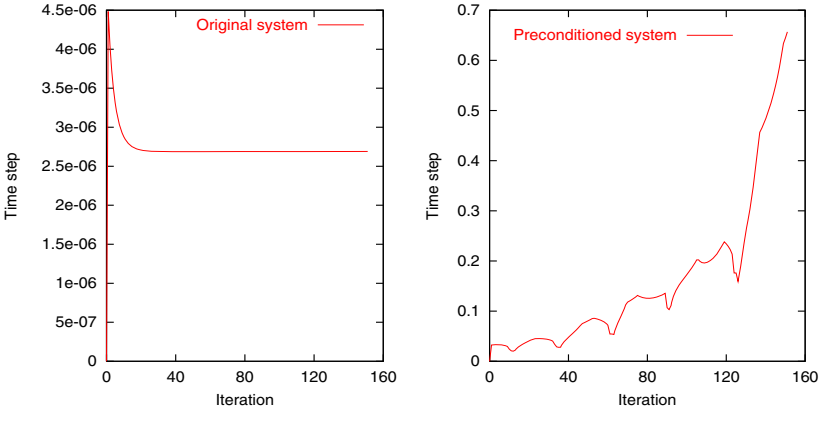


Fig. 6.3 Adaptive time steps of Runge-Kutta-Fehlberg method

preconditioned system. Since the above mentioned preconditioner contains the diagonal sub-matrices for the interior grids, the additional computational cost is just one multiplication per grid point. The one-dimensional Laplacian in the preconditioner is discretized by a second order central difference formula. This discrete formula is also multiplied by a diagonal matrix and operated on the boundary terms only. Therefore, it also involves an almost negligible amount of additional cost of computation.

Figure 6.4 presents the convergence history of the full optimization problem and the analysis problem alone, when the same pseudo-time-stepping method is used to solve the state equation. As we see, the full optimization problem takes almost double the iterations as that of the state equation alone. The CPU time required for

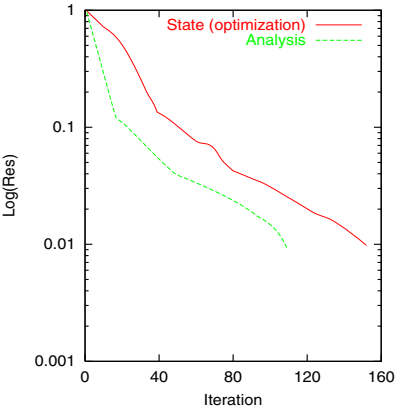


Fig. 6.4 Comparison of residuals of optimization and the analysis problem

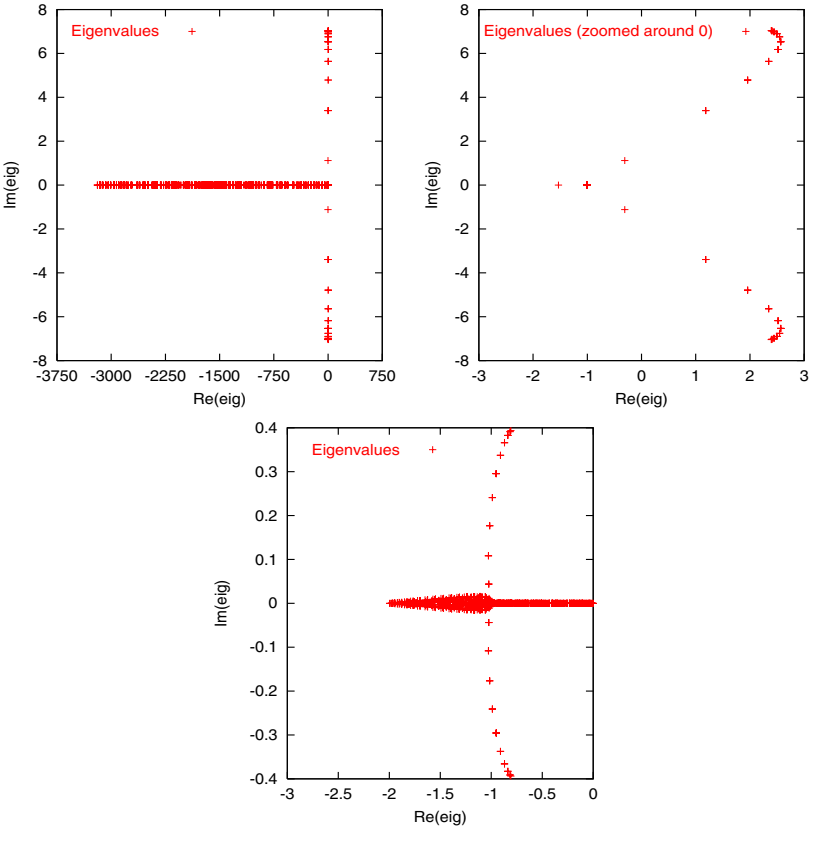


Fig. 6.5 Spectrum of the original system (top row) and preconditioned system (bottom)

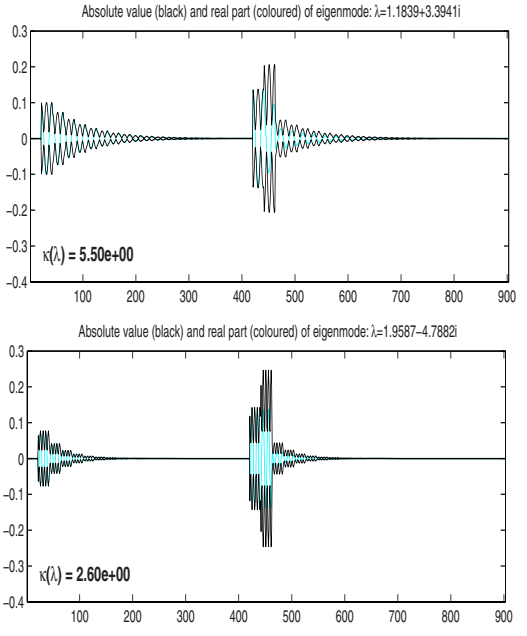


Fig. 6.6 Eigenmodes of the eigenvalues (of original system) with positive real part

both problems are 0.810 sec. (full optimization run) and 0.260 sec. (state equation only) respectively on an Intel(R) Xeon(TM) CPU 1700MHz machine. This means, the optimization run requires 3.2 times the effort (in terms of CPU time) of a pure simulation run, where the same pseudo-time-stepping methods are used.

From an analytical point of view, the contractivity of the un-preconditioned optimization pseudo-time-stepping method (6.15) and of the preconditioned pseudo-time-stepping method (6.21) is in principle in question. Theoretical investigations into that will be the subject of a forthcoming research. Here we provide eigenvalue distribution of the un-preconditioned (Figure 6.5, top row) and preconditioned (Figure 6.5, bottom) systems. In the un-preconditioned case we observe unstable modes. Investigating the corresponding eigenvectors (computed using EigTool [165]), we see strong oscillations in the control. In Figures 6.6 eigenvectors corresponding to two unstable modes are plotted (where $\kappa(\lambda)$ indicates the condition number of the eigenvalue λ). Since the solution is smooth and we start with a smooth initial guess, we did not observe divergence of the un-preconditioned case. On the other hand a comparison of Figure 6.5 top and bottom rows shows that the rSQP preconditioner mainly compresses the eigenvalue distribution to the left-half plane. The consequences of this observation and the fact of a strong improvement of the convergence will be investigated later.

6.9 Conclusions

A new method is developed for the optimization problems governed by PDEs. It is based on simultaneous pseudo-time-stepping for evolution equations. The main advantage of this is to be consistent with the preferred globalization techniques for solving the forward problem (specially in problems that involve discontinuities). The method can be viewed as a continuous rSQP method for solving such problems. The symbol of the Hessian is used as its approximation in the preconditioner. This method applied to a model optimal control problem is only a factor of 3.2-times expensive as that of the analysis problem.

Chapter 7
Aerodynamic Shape Optimization Using Simultaneous Pseudo-Time-Stepping

7.1 Introduction

Computational Fluid Dynamics (CFD) has made considerable progress so that it is being used extensively for analysis of any prescribed aerodynamic shape. Recently, a combination of CFD and numerical optimization method is being used to determine an optimum aerodynamic shape automatically for intended applications. Due to the fact that the cost of computation is much cheaper than the cost of wind tunnel experiment, this has proved to be extremely valuable in practice since it provides with multiple alternatives to the designer. The best designs identified by numerical computations are confirmed for a decision by the wind tunnel experiment. Mathematically such shape design problems can be formulated as a control problem for systems governed by PDEs [90].

In control theory based design methods, the problem is regarded as optimal control of the flow equations by changing the shape of the boundary to achieve the goal. Gradient based optimization methods to solve such problems proved to be most effective. In this method, a direction of descent is determined using the gradient informations of the cost function with respect to the design parameters. Once the direction is found, a step, determined by some means (which is known as line search or globalization strategy), towards this direction is taken in each optimization iteration. This process is continued until convergence.[1]

For computation of gradients or sensitivity derivatives one can use, for example, finite-difference method in which a small variation is introduced in each design parameter and the flow is recalculated to obtain the variation in the objective function, as discussed in Chapter 3. This is repeated for all the design parameters. The disadvantage of this method is that the number of flow calculations needed to estimate the gradient is proportional to the number of design parameters [78]. Continuous adjoint method, (see, for example, in [90]) has advantage over the finite-difference

[1] Some of the materials appearing in this chapter can also be found in [59, 71], reprinted with kind permissions from American Institute of Aeronautics and Astronautics, Inc. and from Elsevier.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 81–104.
springerlink.com

method. In this method, another set of partial differential equations, known as adjoint or costate equations, are to be solved. The cost of solving these equations is comparable to the cost of solving the state (flow) equations. In each iteration of a traditional gradient method, e.g., steepest descent or conjugate gradient or quasi-Newton method, one has to solve the state and costate equations with sufficient accuracy. This has to be repeated several times until the convergence of the optimization algorithm. Therefore, inspite of using high fidelity CFD, the overall cost of computation is quite high to get an optimal solution. Computational results based on these methodologies have been presented, among others, in [91, 92, 142, 143, 45, 47] on structured grids. An application of this method on unstructured grids has been presented in [4].

In [67] we proposed a new method for solving such problems using simultaneous pseudo-time stepping, as explained in Chapter 6. This formulation is advantageous since the steady-state flow (as well as adjoint) solution is obtained by integrating the pseudo-unsteady Euler (or the Navier-Stokes) equations in this problem class. Therefore, one can use the same time-stepping philosophy for the whole set of equations and a reduced Sequential Quadratic Programming (SQP) methods based preconditioner can be used to accelerate the convergence. In [59, 58, 71] we have implemented that method for shape design examples using the Euler equations. The method has been used for determining Pareto curve for multi-objective problem in [146]. The method is 'one-shot' since we perform one time-step for each design update. It differs with the 'one-shot' method of [158] in which the design variables are updated in a hierarchical manner. In [133, 134, 135] investigation has been made to improve the computational efficiency and storage requirements for such optimization problems using the simultaneous approach where sparsity of the Jacobian and of the Hessian matrices are examined and utilized. Simultaneous approach is also proposed in [85] which is based on successive linear programming involving solution of a linear adjoint system. In [40] all-at-once approach is proposed in the SQP scheme where simplification is introduced in the reduced Hessian so that the method requires no adjoint solution. Both the methods of [85, 40] require line-search and applied to 1D problems.

The optimization problem can be written in abstract form (6.3) ([67, 71]). Here, the equation $\mathbf{c}(w,q) = 0$ represents the steady-state flow equations (in the present chapter the Euler equations) together with the boundary conditions, w is the vector of dependent variables and q is the vector of design variables. The objective $I(w,q)$ is the drag of an airfoil or wing for the purposes of this chapter. Typically, there arise inequality constraints of the form

$$h(w,q) \geq 0,$$

which, in practical applications, often pose severe restrictions on the validity region of the model or for the design construction. In the present chapter we are discussing the problem without such additional constraints, and the addition of constraints is addressed in the next chapters.

7.2 Pseudo-Time-Stepping for Optimization Problems

Corresponding to the presentation in Chapter 6, the pseudo-time embedded system of differential equations that we consider, for the optimization problem, is

$$\begin{pmatrix} \dot{w} \\ \dot{q} \\ \dot{\lambda} \end{pmatrix} = K \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -\mathbf{c} \end{pmatrix}, \tag{7.1}$$

where

$$K = \begin{bmatrix} 0 & 0 & \mathbf{A}^\top \\ 0 & B & \left(\frac{\partial \mathbf{c}}{\partial q}\right)^\top \\ \mathbf{A} & \frac{\partial \mathbf{c}}{\partial q} & 0 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}\frac{\partial \mathbf{c}}{\partial q}B^{-1}\left(\frac{\partial \mathbf{c}}{\partial q}\right)^\top \mathbf{A}^{-\top} & -\mathbf{A}^{-1}\frac{\partial \mathbf{c}}{\partial q}B^{-1} & \mathbf{A}^{-1} \\ -B^{-1}\left(\frac{\partial \mathbf{c}}{\partial q}\right)^\top \mathbf{A}^{-\top} & B^{-1} & 0 \\ \mathbf{A}^{-\top} & 0 & 0 \end{bmatrix}.$$

This formulation is advantageous since the steady-state flow is obtained by integrating the pseudo-unsteady Euler equations (or the Navier-Stokes equations, as discussed in Chapter 12) in this problem class. Therefore, one can use the same time-stepping philosophy for the whole set of equations and preconditioners can be used to accelerate the convergence (as discussed in the previous chapter). In the present chapter, we implement this method for the shape design example using the Euler equations. The number of iterations required for the full optimization problem is about 2 times that of the forward simulation (only) problem. This means a drastic reduction of the computational cost compared to the 'traditional' or 'black-box' gradient methods.

Within the inexact reduced SQP-preconditioner, one has to look for an appropriate approximation of the reduced Hessian, B. In particular, when dealing with partial differential equations constituting the state equations, the reduced Hessian can often be expressed as a pseudo-differential operator. Pseudo-differential operators [84, 105] are characterized by their, so-called, symbol in terms of Fourier analysis. This can be exploited for preconditioning purposes as in [67] and presented in Chapter 6. For the applications in this chapter, the reduced Hessian update is discussed in Section 7.5. In the following we discuss in detail the governing equations, their discretization, gradient computation, and grid perturbation strategies.

7.3 Detailed Equations of the Aerodynamic Shape Optimization Problem in 2D

In this section we explain briefly the state, costate and design equations represented in equations (6.6) for the aerodynamic shape optimization problem. The governing state equations are presented in differential form in Chapter 2. Here we present the corresponding conservative form which is necessary for finite-volume discretization.

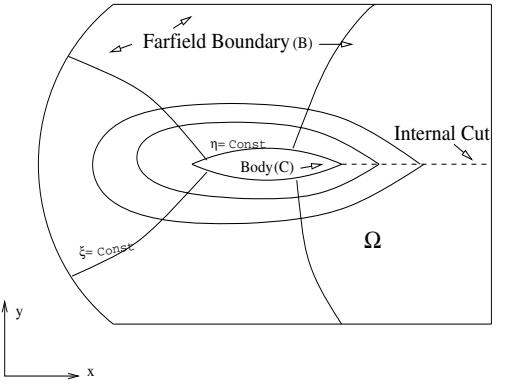


Fig. 7.1 Physical Domain of the problem

State equations: Since we are interested in the steady flow, a proper approach for numerical modeling is to integrate the unsteady Euler equations in time until a steady state is reached. These equations in Cartesian coordinates (x,y) for two-dimensional flow can be written in integral (or conservative) form for the region Ω with boundaries $\partial\Omega$ $(= B \cup C)$ (see Figure (7.1)) as

$$\frac{\partial}{\partial t}\int_{\Omega} w\, d\Omega + \int_{\partial\Omega} F \cdot \mathbf{n}\, ds = 0, \tag{7.2}$$

where $\mathbf{n}$ denotes the unit outward normal to $\partial\Omega$ and

$$w := \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{bmatrix}, \qquad F := [f_1, f_2], \qquad f_1 := \begin{bmatrix} \rho u \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ \rho u_1 H \end{bmatrix} \quad \text{and} \quad f_2 := \begin{bmatrix} \rho v \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ \rho u_2 H \end{bmatrix}.$$

For a perfect gas the pressure and total enthalpy are given by

$$p = (\gamma - 1)\rho \left\{ E - \frac{1}{2}(u_1^2 + u_2^2) \right\}, \quad H = E + \frac{p}{\rho},$$

respectively. The boundary conditions used to solve these equations are the zero normal velocity on the solid wall C, and the farfield boundary B is treated by considering the incoming and outgoing characteristics based on the one dimensional Riemann invariants [100].

The cost function that we choose in the present optimization problem is drag reduction (with the geometric constraint of constant thickness of the airfoil). Hence, the cost function reads as

$$I(w,q) := C_D = \frac{1}{C_{ref}} \int_C C_p \,(n_x \cos\alpha + n_y \sin\alpha)\, ds, \tag{7.3}$$

where the surface pressure coefficient is defined by

$$C_p := \frac{2(p - p_\infty)}{\gamma M_\infty^2 p_\infty}. \tag{7.4}$$

The other force coefficients, namely lift and pitching moment, are defined as

$$C_L = \frac{1}{C_{ref}} \int_C C_p \left(n_y \cos\alpha - n_x \sin\alpha\right) ds, \tag{7.5}$$

and

$$C_M = \frac{1}{C_{ref}^2} \int_C C_p \left(n_y (x - x_m) - n_x (y - y_m)\right) ds, \tag{7.6}$$

where (x_m, y_m) is the pitching moment's reference point.

Costate equations: The costate or adjoint Euler equations are given by (see, for example, [90])

$$\frac{\partial}{\partial t} \int_\Omega \lambda \, d\Omega + \int_{\partial\Omega} \bar{F} \cdot \mathbf{n} \, ds = 0, \tag{7.7}$$

where the vector λ contains the components of the adjoint variable and $\bar{F}$ is the matrix of adjoint flux density, defined as

$$\lambda := \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix}, \qquad \bar{F} := \left[\left(\frac{\partial f_1}{\partial w}\right)^T \lambda \, , \left(\frac{\partial f_2}{\partial w}\right)^T \lambda \right].$$

The boundary conditions for the adjoint Euler equations on the solid body are, for the above mentioned cost function, given by

$$n_x \lambda_2 + n_y \lambda_3 = \text{RHS}, \quad \text{on } C. \tag{7.8}$$

For the above mentioned cost function the right hand side of (7.8) is

$$\text{RHS} = -\frac{2}{\gamma M_\infty^2 p_\infty C_{\text{ref}}} (n_x \cos\alpha + n_y \sin\alpha), \quad \text{on } C. \tag{7.9}$$

The farfield boundary conditions are based upon incoming and outgoing characteristics and free-stream conditions apply there as well. It is important to note that the adjoint Euler equations are linear in λ and the wall boundary conditions depend on the cost function.

Design or Sensitivity equation: For the design equation (6.6c), we need an expression for the derivative of the Lagrangian with respect to the geometry of the airfoil.

All the computations are carried out in a Generalized coordinate system. Therefore, a transformation is used to transform the physical (x,y)-domain to the computational (ξ,η)-domain. In the computational domain, the components of the gradient $\frac{\partial L}{\partial q}$ can be determined by integrating the adjoint solutions multiplied by the metric sensitivities as follows

$$\left(\frac{\partial L(q+\varepsilon\tilde{q})}{\partial q}\right)\Big|_{\varepsilon=0} = -\int_C p(-\lambda_2\tilde{q}^y_\xi + \lambda_3\tilde{q}^x_\xi)ds + T1 - \int_\Omega \left(\lambda^T_\xi\left(\tilde{q}^y_\eta f_1 - \tilde{q}^x_\eta f_2\right) + \lambda^T_\eta\left(-\tilde{q}^y_\xi f_1 + \tilde{q}^x_\xi f_2\right)\right)d\Omega, \quad (7.10)$$

where $\tilde{q}$ is the variation in the geometry of the airfoil and $\tilde{q}^x$, $\tilde{q}^y$ are its x- and y-components, $\left((\tilde{q}^\perp)^x,(\tilde{q}^\perp)^y\right)$ are the components of the unit normal to $\tilde{q}$. For the sensitivities of the cost function the term $T1$ is given by

$$T1 = \frac{1}{C_{ref}}\int_C C_p\left((\tilde{q}^\perp)^x\cos\alpha + (\tilde{q}^\perp)^y\sin\alpha\right)ds, \quad (7.11)$$

where $\left((\tilde{q}^\perp)^x,(\tilde{q}^\perp)^y\right)$ are the components of the unit normal to $\tilde{q}$.

7.4 Discretization

State equations: The governing equations are discretized following the method of lines. Space discretization of the compressible Euler equations is carried out using a cell centered finite volume scheme. The physical domain is subdivided into a large number of quadrilateral cells $(\Omega_{i,j})$ as shown in Figure 7.2. Since the conservation

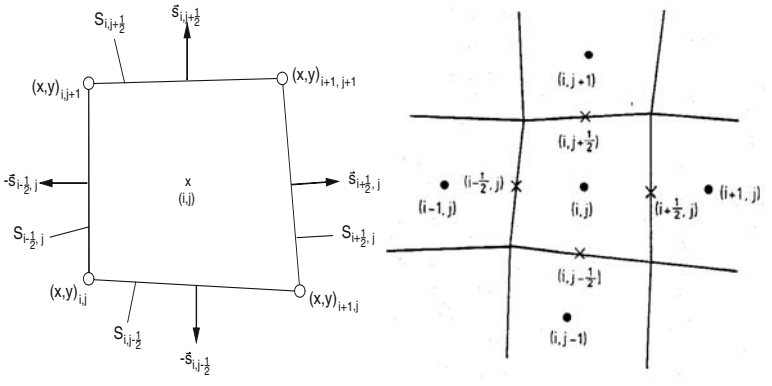


Fig. 7.2 Quadrilateral cell (i,j) (left) and location of dependent variables (•) and flux values (×) (right)

laws, Eq.(7.2), are valid for any arbitrary control volume, they also hold locally for each cell $(\Omega_{i,j})$. Hence,

$$\frac{d}{dt}\int_{\Omega i,j} w\,d\Omega + \int_{\partial\Omega_{i,j}} F\cdot\mathbf{n}\,ds = 0, \tag{7.12}$$

where the boundary $\partial\Omega_{i,j}$ consists of the four sides of the quadrilateral, and $\mathbf{n}$ is the unit outward normal to the surface. The flow quantities w are taken to be volume averaged at the center (i,j) of the cell $\Omega_{i,j}$ (see Figure 7.2), that is,

$$w_{i,j} := \frac{1}{V_{i,j}}\int\limits_{\Omega_{i,j}} w\,d\Omega, \tag{7.13}$$

where $V_{i,j}$ is the volume of the cell $\Omega_{i,j}$. If the mesh is time independent and the second integral of (7.12) is approximated using the mid-point rule, the discrete analog of Eq.(7.12) is written as

$$V_{i,j}\left(\frac{d}{dt}w_{i,j}\right) + Q_{i,j} = 0, \tag{7.14}$$

where $Q_{i,j}$ represents the net flux out of a cell (i,j) and is given by

$$Q_{i,j} := F_{i,j+\frac{1}{2}}\mathbf{S}_{i,j+\frac{1}{2}} - F_{i,j-\frac{1}{2}}\mathbf{S}_{i,j-\frac{1}{2}} + F_{i+\frac{1}{2},j}\mathbf{S}_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j}\mathbf{S}_{i-\frac{1}{2},j}, \tag{7.15}$$

with $\mathbf{S}_{i,j-\frac{1}{2}}$ being the normal to the side $S_{i,j-\frac{1}{2}}$ and $F_{i,j-\frac{1}{2}}$ is calculated using the average of w at the cell centers (i,j) and $(i,j-1)$. The spatial positions of the the flow quantities and the flux quantities are different in the finite volume discretization and are shown in Figure 7.2. For example, the flux through the boundary $S_{i,j-\frac{1}{2}}$ is given by

$$F_{i,j-\frac{1}{2}}.\mathbf{S}_{i,j-\frac{1}{2}} = \begin{bmatrix} \rho_{i,j-\frac{1}{2}}\left(\mathbf{q}_{i,j-\frac{1}{2}}.\mathbf{S}_{i,j-\frac{1}{2}}\right) \\ (\rho u)_{i,j-\frac{1}{2}}\left(\mathbf{q}_{i,j-\frac{1}{2}}.\mathbf{S}_{i,j-\frac{1}{2}}\right) + P_{i,j-\frac{1}{2}}S^{(1)}_{i,j-\frac{1}{2}} \\ (\rho v)_{i,j-\frac{1}{2}}\left(\mathbf{q}_{i,j-\frac{1}{2}}.\mathbf{S}_{i,j-\frac{1}{2}}\right) + P_{i,j-\frac{1}{2}}S^{(2)}_{i,j-\frac{1}{2}} \\ (\rho H)_{i,j-\frac{1}{2}}\left(\mathbf{q}_{i,j-\frac{1}{2}}.\mathbf{S}_{i,j-\frac{1}{2}}\right) \end{bmatrix},$$

where $(S^{(1)}, S^{(2)})$ are two components of $\mathbf{S}_{i,j-\frac{1}{2}}$. The quantities at the midpoints are the simple averages of their values at the nodal points. Further details of the flux computation can be found in [100, 107].

The finite volume discretization as explained above, amounts to central differencing and thus requires explicit addition of dissipation terms for stability. Dissipative fluxes are added to the average flux Q_{ij} in Eq.(7.14) to give

$$V_{ij}\left(\frac{d}{dt}w_{ij}\right) + Q_{ij} - D_{ij} = 0, \tag{7.16}$$

where D_{ij} is the dissipative flux, and is given by

$$D_{ij} = \mathbf{d}_{i+1/2,j} - \mathbf{d}_{i-1/2,j} + \mathbf{d}_{i,j+1/2} - \mathbf{d}_{i,j-1/2}. \tag{7.17}$$

In Jameson et.al.(1981) scheme $\mathbf{d}_{ij}$ are a special kind of blend of second and fourth order differences expressed by

$$\begin{aligned}\mathbf{d}_{i+1/2,j} = \alpha_{i+1/2,j} \Big\{ & \varepsilon^{(2)}_{i+1/2,j} \left(w_{i+1,j} - w_{i,j}\right) \\ & - \varepsilon^{(4)}_{i+1/2,j} \left(w_{i+2,j} - 3w_{i+1,j} + 3w_{i,j} - w_{i-1,j}\right) \Big\}. \end{aligned} \tag{7.18}$$

Here $\varepsilon^{(2)}_{i+1/2,j}$ and $\varepsilon^{(4)}_{i+1/2,j}$ are adaptive coefficients designed to switch on enough dissipation where it is needed and are defined in terms of pressure as

$$\nu_{i,j} = \frac{|\, p_{i+1,j} - 2p_{i,j} + p_{i-1,j} \,|}{p_{i+1,j} + 2p_{i,j} + p_{i-1,j}}, \tag{7.19}$$

$$\varepsilon^{(2)}_{i+1/2.j} = k^{(2)} max(\nu_{i+1.j}, \nu_{i,j}), \tag{7.20}$$

$$\varepsilon^{(4)}_{i+1/2,j} = max\{0, (k^{(4)} - \varepsilon^{(2)}_{i+1/2,j})\}, \tag{7.21}$$

with $k^{(2)}, k^{(4)}$ being suitable constants.

The scaling factor $\alpha_{i+1/2,j}$ is given by

$$\alpha_{i+1/2,j} = \frac{1}{2} \left(\frac{V_{i,j}}{\Delta t_{i,j}} + \frac{V_{i+1,j}}{\Delta t_{i+1,j}} \right), \tag{7.22}$$

where $V_{i,j}$ is the cell volume and $\Delta t_{i,j}$ is an estimate of the time step limit for a nominal Courant number $(= c\frac{\Delta t}{\Delta x})$ of unity. The coefficient $\varepsilon^{(2)}$ is proportional to the second difference of pressure in smooth regions of the flow proportional to the square of the mesh size, while $\varepsilon^{(4)}$ is of order one. The quantity ρH has been used instead of ρE in the dissipative terms in the energy equation in order to admit H=constant, a solution of that. It is to be noted that the basic fourth order dissipation is necessary for stability, even if no shock wave is present in the flow field.

These equations are then integrated in time using a 5-stage, 4th order, Runge-Kutta type scheme. This scheme takes the following form for Eq.(7.16) at time level n:

$$\begin{aligned} w^{(0)}_{i,j} &= w^{n}_{i,j}, \\ w^{(1)}_{i,j} &= w^{(0)}_{i,j} - \alpha_1 \Delta t P^{(0)}_{i,j}, \\ &\vdots \\ w^{(5)}_{i,j} &= w^{(0)}_{i,j} - \alpha_5 \Delta t P^{(4)}_{i,j}, \\ w^{(n+1)}_{i,j} &= w^{(5)}_{i,j}, \end{aligned} \tag{7.23}$$

where the residuals

$$P_{i,j}^{(k)} := \frac{1}{V_{i,j}} \left(Q_{i,j}^{(k)} - D_{i,j}^{(k)} \right), \quad k = 0,1,2,3,4$$

and the values of the constant coefficients are $\alpha_1 = 1/4$, $\alpha_2 = 1/6$, $\alpha_3 = 3/8$, $\alpha_4 = 1/2$, $\alpha_5 = 1$.

In this case the steady state is independent of the time step Δt and is amenable to a variety of techniques for rapid convergence. For stability, a modified condition, as in [107],

$$\Delta t_{i,j} \leq \kappa V_{i,j} \left[|\bar{q}_{i,j} \cdot \mathbf{S}_{i+\frac{1}{2},j}| + |\bar{q}_{i,j} \cdot \mathbf{S}_{i,j+\frac{1}{2}}| + a_{i,j} \left\{ |\mathbf{S}_{i+\frac{1}{2},j}| + |\mathbf{S}_{i,j+\frac{1}{2}}| \right\} \right]^{-1}, \quad (7.24)$$

has been used to determine the time step for each cell. Here κ is the Courant number and $\bar{q}_{i,j}$ and $a_{i,j}$ represent the velocity vector and the velocity of sound respectively at (i,j). Thus, the stability limit on Δt for a time accurate calculation is

$$\Delta t = \min_{i,j} \Delta t_{i,j}. \quad (7.25)$$

The solution is advanced in time using the local time step $\Delta t_{i,j}$ as in Eq.(7.24), based on κ, instead of Eq.(7.25). This allows for faster signal propagation and thus faster convergence. The details of the grid generation and solution methodology can be found in [108, 109] as well as in [56, 63, 64, 57].

Costate equations: Due to structural similarity of the state and costate equations, it is obvious that one can use the same solver for both sets of equations. These equations are also discretized in space using a cell centered finite volume scheme on the same computational grid as described for the Euler equations. The adjoint equations in each cell (i,j) are written as

$$\frac{d}{dt} \int\limits_{\Omega_{i,j}} \lambda \, d\Omega + \int\limits_{\partial \Omega_{i,j}} \bar{F} \cdot \mathbf{n} \, ds = 0. \quad (7.26)$$

Analogous to Eq.(7.15), the adjoint flux $\tilde{Q}_{i,j}$ is computed as

$$\tilde{Q}_{i,j} := \bar{F}_{i,j+\frac{1}{2}} \mathbf{S}_{i,j+\frac{1}{2}} - \bar{F}_{i,j-\frac{1}{2}} \mathbf{S}_{i,j-\frac{1}{2}} + \bar{F}_{i+\frac{1}{2},j} \mathbf{S}_{i+\frac{1}{2},j} - \bar{F}_{i-\frac{1}{2},j} \mathbf{S}_{i-\frac{1}{2},j},$$

where the averaged tensors of flux density is computed as

$$\bar{F}_{i,j+\frac{1}{2}} := \left[\left(\frac{\partial f}{\partial w} \right)_{i,j}^{T} \frac{\lambda_{i,j+1} + \lambda_{i,j}}{2}, \left(\frac{\partial g}{\partial w} \right)_{i,j}^{T} \frac{\lambda_{i,j+1} + \lambda_{i,j}}{2} \right],$$

with $\lambda_{i,j}$ being the volume averaged value of λ as defined in Eq.(7.13). Averaging the cell face normals $\mathbf{S}$ leads to

$$\tilde{Q}_{i,j} = \sum_{\substack{l\in\{i,j\}\\ m\in\{i,j\}\setminus\{l\}}} \left(\bar{s}_x^{(l)} \left(\frac{\partial f}{\partial w}\right)^T_{i,j} + \bar{s}_y^{(l)} \left(\frac{\partial g}{\partial w}\right)^T_{i,j} \right) \frac{\lambda_{m,l+1} - \lambda_{m,l-1}}{2},$$

where

$$\begin{pmatrix} \bar{s}_x^{(l)} \\ \bar{s}_y^{(l)} \end{pmatrix} := \frac{\mathbf{S}_{m,l+\frac{1}{2}} + \mathbf{S}_{m,l-\frac{1}{2}}}{2},$$

and enables to evaluate the adjoint flux efficiently by introducing the transformation

$$T^{-1} = \begin{pmatrix} 1 & -u & -v & \frac{1}{2}(u^2+v^2) \\ 0 & 1 & 0 & -u \\ 0 & 0 & 1 & -v \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

so that

$$\tilde{Q}^{(l)}_{i,j} = T \left(\bar{s}_x^{(l)} \left(\frac{\partial f}{\partial w}\right)^T_{i,j} + \bar{s}_y^{(l)} \left(\frac{\partial g}{\partial w}\right)^T_{i,j} \right) T^{-1}.$$

Introducing $\hat{\lambda}^{(l)}_{i,j} := T \frac{\lambda_{m,l+1}-\lambda_{m,l-1}}{2}$, one gets

$$\tilde{Q}_{i,j} = \sum_{l\in\{i,j\}} T^{-1} \tilde{Q}^{(l)}_{i,j} \hat{\lambda}^{(l)}_{i,j}, \quad \tilde{Q}^{(l)}_{i,j} = \begin{pmatrix} u\bar{s}_x^{(l)} + v\bar{s}_y^{(l)} & 0 & 0 & 0 \\ \bar{s}_x^{(l)} & u\bar{s}_x^{(l)} + v\bar{s}_y^{(l)} & 0 & \frac{\gamma}{\gamma-1}\frac{p}{\rho}\bar{s}_x^{(l)} \\ \bar{s}_y^{(l)} & 0 & u\bar{s}_x^{(l)} + v\bar{s}_y^{(l)} & \frac{\gamma}{\gamma-1}\frac{p}{\rho}\bar{s}_y^{(l)} \\ 0 & (\gamma-1)\bar{s}_x^{(l)} & (\gamma-1)\bar{s}_y^{(l)} & u\bar{s}_x^{(l)} + v\bar{s}_y^{(l)} \end{pmatrix}.$$

These matrices $\tilde{Q}^{(l)}_{i,j}$ are easy to evaluate and one finally obtains the finite volume scheme

$$V_{i,j} \left(\frac{d\lambda_{i,j}}{dt} \right) + \tilde{Q}_{i,j} - D_{i,j} = 0,$$

where $D_{i,j}$ is again the artificial dissipation as described in [100] for the adjoint field vector λ. These equations are integrated in time using the same Runge-Kutta scheme as described above. The details of the spatial discretization and the adjoint flux computations are described in [45].

Surface parameterization: The airfoil is decomposed into thickness and camber distributions. During the optimization only camber distributions are modified. This allows the thickness of the airfoil to remain constant (otherwise the drag reduction problem will result in a flat geometry). In this representation the y-coordinates of the surface are written in parametric form

$$y(\bar{r}) = \sum_{i=1}^{n} \tau_i \psi_i(\bar{r})$$

where $\bar{r}$ is the coordinate along the airfoil chord, ψ_i are the base function and τ_i are the design parameters (here $q = (\tau_1, \tau_2, \cdots, \tau_n)^\top$). In the current study Hicks-Henne functions [78] are selected as base functions which are given by:

$$\psi_i(r) = \sin\left(\pi r^{\frac{\ln(0.5)}{ln(a)}}\right)^b$$

where a and b control the center and thickness of the perturbations and r is the normalized coordinate along the chord.

Gradient computation: As an efficient method of calculating the gradient $(\delta I)_{m=1,\ldots,n} := \nabla_q I$, by evaluating the integrals (7.10), we use, as in [46], the so called 'grid moving technique' based on J. Reuther's approach (s.[99]). These integrals (7.10) are dependent on the adjoint field vector λ and the metric sensitivities generated by the perturbation of the geometry (by the n design variables). The idea is to allow the geometry perturbation in the whole flow field (the whole grid) while keeping the far field boundary fixed. For this, one introduces a distance function $R(\xi,\eta) = 1 - \frac{\eta}{\eta_B(\xi)}$, where $\eta_B(\xi)$ denotes the values of η at the far field corresponding to the points $(\xi,0)$ at the wall (see Fig. 7.1). Then it holds that

$$R(\xi,0) = 1 \quad \text{(at the wall)} \quad \text{and} \quad R(\xi,\eta_B(\xi)) = 0 \quad \text{(at the far field)}.$$

The grid for the perturbed (new) geometry is defined by

$$x_{new} - x_{old} := R \cdot (x_{new_s} - x_{old_s}),$$

$$y_{new} - y_{old} := R \cdot (y_{new_s} - y_{old_s}),$$

where the index s refers to the values at the surface and x_{new}, y_{new} correspond to x_{new_s}, y_{new_s} respectively. Thus, the perturbations are

$$\delta x = R\delta x_s, \quad \delta y = R\delta y_s,$$

where $\tilde{q}^x = \delta x_s$, $\tilde{q}^y = \delta y_s$ in equation (7.10), and the metric sensitivities are

$$\delta x_\xi = R \cdot (\delta x_s)_\xi + R_\xi \delta x_s, \quad \delta y_\xi = R \cdot (\delta y_s)_\xi + R_\xi \delta y_s,$$

$$\delta x_\eta = R \cdot (\delta x_s)_\eta + R_\eta \delta x_s, \quad \delta y_\eta = R \cdot (\delta y_s)_\eta + R_\eta \delta y_s.$$

The second term of each of these equalities is very small (s.[46]) and, therefore, could be neglected from the above expressions. The effect of this simplification has been studied and justified in [46]. In that case the metric sensitivities can be expressed as

$$\delta x_\xi \approx R \cdot (\delta x_s)_\xi, \quad \delta y_\xi \approx R \cdot (\delta y_s)_\xi,$$

$$\delta x_\eta \approx R \cdot (\delta x_s)_\eta, \quad \delta y_\eta \approx R \cdot (\delta y_s)_\eta,$$

and one obtains the components of the gradient (for the cost function (7.3)) as integrals along the geometry C

$$\delta I = -\int_C \left((\delta y_s)_\eta \int_\eta \frac{\partial \lambda^T}{\partial \xi}(R(\eta)\cdot f)d\eta \right) d\xi + \int_C \left((\delta x_s)_\eta \int_\eta \frac{\partial \lambda^T}{\partial \xi}(R(\eta)\cdot g)d\eta \right) d\xi$$
$$+ \int_C \left((\delta y_s)_\xi \int_\eta \frac{\partial \lambda^T}{\partial \eta}(R(\eta)\cdot f)d\eta \right) d\xi - \int_C \left((\delta x_s)_\xi \int_\eta \frac{\partial \lambda^T}{\partial \eta}(R(\eta)\cdot g)d\eta \right) d\xi$$
$$- \int_C p\left(-\lambda_2(\delta y_s)_\xi + \lambda_3(\delta x_s)_\xi \right) d\xi + \frac{1}{S_{ref}} \int_C C_p \left((\delta y_s)_\xi \cos\alpha - (\delta x_s)_\xi \sin\alpha \right) d\xi .$$

The integrals in η have to be evaluated once. Then, for each design variable only the integration in ξ along C remains to be evaluated. For the computation of gradients in 3D, the integral in (7.10) is to be evaluated which requires the variation of the metric in the complete flow field domain. This is a difficult and time consuming task, especially in complex 3D configurations. As an alternative to this, surface formulations are proposed in [162, 95] where the data is required at the surface only. The details of computations for 3D problem can be found in [70].

Grid-perturbation strategy: As the shape of the airfoil changes during the optimization process, the location of the grid nodes has to be adjusted. This can be done by generating a new grid after each design iteration or by using a grid-perturbation strategy after each design iteration.

The grid perturbation strategy follows the idea of [129], but in the present study, a specific property of structured mesh is used. A finite number of cells, namely j_0, surrounding the airfoil are defined, and all nodes belonging to this area are moved exactly as the nodes at the boundary. The remaining unchanged cells are smoothly moved until the farfield. The modified grid is then given by, for each cell i,

$$y(i,j)_{(new)} = y(i,j)_{(old)} + Dy(i) \quad \text{if} \quad j \le j_0$$
$$y(i,j)_{(new)} = y(i,j)_{(old)} + 0.5 * Dy(i)(1+\cos(\pi * Sj)) \quad \text{if} \quad j > j_0$$

where $Dy(i)$ represents the deformation of the surface of airfoil at the cell i , i.e. $Dy(i) = y(i,1)_{(new)} - y(i,1)_{(old)}$ and Sj is given by $Sj = (j-j_0)/(jmax-j_0)$, which represents the distance in index notation between the deformed cell (i,j) and the last non-deformed cell belonging to the same i indices, i.e. (i,j_0) cell. Here $jamx$ represents the total number of cells in j-direction.

7.5 Reduced Hessian Updates

Within the inexact rSQP-preconditioner, one has to look for an appropriate approximation of the inverse of reduced Hessian, $B^{-1} = H$(say) . The problem of aerodynamic shape optimization involves the nonlinear system of hyperbolic PDEs. The solution of these equations often contains discontinuities, specially in transonic and supersonic regimes. There the symbol of the Hessian is difficult to deduce in terms of Pseudo-differential operators. Therefore it is necessary to find some other means to approximate the reduced Hessian during the optimization iterations.

We have used the ideas of updates in Quasi-Newton methods for large scale optimization problems. Instead of recomputing the iteration matrices from scratch at

every iteration, use the most recently observed curvature information of the objective function to construct the approximation of the Hessian. We have used three different ways for such updates:

Case 1: The simplest approximation is to set it to a multiple of the identity matrix, $H_k = \beta\,\delta_{ij}$ where β is a constant and δ_{ij} is the identity matrix. The constant has to be chosen such that it reflects the scaling of the updates of design variables.

Case 2: In this case we use another scaling factor which is based on most recent reduced gradient and parameter update informations, as in the case of (memoryless) BFGS updates. We define $s_k := (q_{k+1} - q_k)$ and $z_k := (g_{k+1} - g_k)$, where $g_k = \left\{ \nabla_q L - (\frac{\partial \mathbf{c}}{\partial q})^\top (A^\top)^{-1} \nabla_w L \right\}$ is the reduced gradient and k represents the iteration number. Then the reduced Hessian update is based on the sign of the product $(z_k^T s_k)$. If the sign is positive, the reduced Hessian is approximated by

$$B_k = \bar{\beta}\,\Gamma_k \delta_{ij}, \text{ with } \Gamma_k = \frac{z_k^T s_k}{z_k^T z_k},$$

where $\bar{\beta}$ is a constant. Otherwise, it is approximated by $\beta\,\delta_{ij}$, where β is a constant as case 1. Additionally, we impose upper and lower limits on the factor so that

$$\beta_{min} < B_k^{-1} < \beta_{max}.$$

This prevents the optimizer from taking steps that are too small or too large. The constants can be chosen, e.g., depending on the accuracy achieved in one time step by the forward and adjoint solver.

The accuracy achieved in one time step depends on many factors in a CFD code, such as the geometry, the type and the size of the computational grid, the type and the order of spatial discretization scheme, time-stepping scheme, CFL condition, acceleration techniques used, etc. For example, the same time-stepping scheme results in larger reduction of residual in a coarser grid computation than in a finer grid computation. Similarly, larger reduction of residual results when multigrid acceleration technique is used in the PDE solver than single grid computations. Hence, accordingly the betas $(\bar{\beta}, \beta_{min}, \beta_{max})$ are to be chosen so that larger design step is used in a coarser grid computation than in a finer grid computation. Similarly, for multigrid CFD solver larger design step is used than single grid CFD solver.

To present an example, for the FLOWer (version 116) code [108, 109], with the following few specifications/parameter values (see the FLOWer User Handbook for details), the values of $\bar{\beta} = 1.0/6.0$ and $\beta_{min} = 0.1$, $\beta_{max} = 0.6$.

Specifications/parameter values - *geometry*: RAE2822 airfoil, *grid size*: 193×33, *Mach*: 0.73, *Alpha*: 2.0 degrees, *discretization*: cell-centered finite-volume, *time-stepping*: Runge-Kutta (central differencing scheme, 5-stage, 4th-order), *tolerance level for global rms value of density residual*: 1.0E-10 (*for adjoint solver*:

1.0E-04), *tolerance level for lift and drag*: 1.0E-08, *number of dummy layers in the grid*: 2, *discretization levels*: 1 (single grid computation), *CFL number*: 6.5 (*for adjoint solver*: 3.5), *max CFL number for non-smoothed scheme*: 3.75, *reciprocal of first order dissipation coefficient (in JST dissipation scheme)*: 2.0, *reciprocal of third order dissipation coefficient (in JST dissipation scheme)*: 48.0, *exponent for the calculation of the scaling factors for the artificial dissipation*: 0.667, *switch for implicit residual smoothing*: 2 (smoothing with variable coefficients), *coefficients in x-, y-, z-direction for residual smoothing*: (0.2,1.2,0.0), *switch for different cut data exchange strategies*: 0 (cut data exchange before every Runge-Kutta stage and before computation of residual for forcing function), *coefficients for 5-stage Runge-Ktta scheme*: (0.25,0.166666,0.375,0.5,1.0).

The method mentioned here is a general one and is independent of a particular simulation code. Hence, by no means those values of the constants are to be taken as reference values. They will not be the same for a different simulation code and they may even not be the same for the same code with any change in the above mentioned parameters. There is no mathematical formula so far to determine the constants, the user of the method needs to make few trials to come up with optimized values of those, so that they represent proper scaling of the updates of design parameters for the CFD solver being used for a particular application example in the optimization problem.

Case 3: In this case we use the ideas of **L-BFGS updates** (as discussed in [131, 121, 132]). The details of implementation for this problem class can be found in [59]

The overall algorithm reads as follows:

Algorithm 1: *The simultaneous pseudo-time-stepping for the preconditioned system*
(0) Set $k := 0$*; start at some initial guess* w_0, λ_0, q_0.
(1) Compute λ^{k+1} *using (7.23) with* Δt *from Eq.(7.24)*
(2) Determine some approximation B_k *of the reduced Hessian of the Lagrangian.*
(3) March in time one step, using Δt *from Eq.(7.25), the design equation.*

$$q^{k+1} = q^k - \Delta t \left\{ B_k^{-1} \nabla_q L - B_k^{-1} (\frac{\partial \mathbf{c}}{\partial q})^\top (A^\top)^{-1} \nabla_w L \right\}$$

(4) Compute w^{k+1} *using (7.23) with* Δt *from Eq.(7.24)*
(5) Set $k := k+1$*; go to (1) until convergence.*

Step (1) represents a Runge-Kutta-version of the first step $(-(A^\top)^{-1}\nabla_w L)$ of the reduced SQP-method (7.1). The block matrices A and $A^\top$ corresponding to the state and costate equations in the preconditioner are just identity matrices in the current as well as in the subsequent implementations. The algorithm above is a 'one-shot' method since we perform one time-step for each design update. However, it differs with the 'one-shot' methods of [113, 158] in which the design variables are updated in a hierarchical manner.

7.6 Numerical Results and Discussion

7.6.1 *Drag Reduction with Geometric Constraint for an RAE2822 Airfoil*

The optimization method is applied to test cases of an RAE2822 airfoil at Mach number 0.73 and angle of incidence 2.0 degrees. The physical domain is discretized using an algebraically generated (193×33) C-grid (Figure 7.3). On this grid, preconditioned pseudo-stationary equations are solved. The airfoil is decomposed into thickness and camberline distributions. The camberline distributions of the airfoil is parameterized by 21 Hicks-Henne parameters, as discussed in Section 7.4. The geometrical constraint of constant thickness is maintained by not changing the parameters representing the thickness. All the computations are carried out on a Linux machine with AMD Opteron Processor 850, CPU 2405 MHz and RAM 16 MB. We start the optimization iteration with initial solutions (i.e., w_0 and λ_0) obtained after 100 time steps (except for the Case 3 updates of the reduced Hessian where we need to take solutions obtained after 150 time steps) of the state and costate equations. After the convergence of the optimization problem another 100 time-steps are carried out for the state equation to get more accurate values of the surface pressure distributions and the force coefficients (which are comparable to the values obtained by other optimal design methods). We use the FLOWer code [108, 109] of the German Aerospace Center (DLR) for solving the forward and adjoint equations. This code has been modified and enhanced for 'one-shot' pseudo-time-stepping method.

The design equation is integrated in time using an explicit Euler scheme and the state and costate equations are integrated in time using a 5-stage Runge-Kutta scheme. Therefore, the time steps used for the three sets of equations are not the same. In the current implementation of FLOWer, the time steps are not same even in each discretization cell as they are determined independently according to the local stability. However, this has no effect on the final solution at steady state.

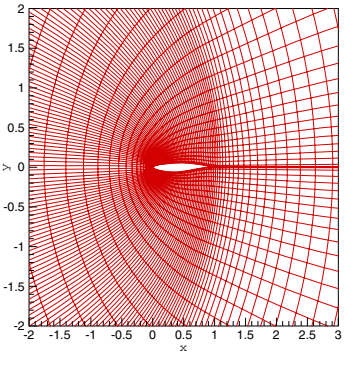


Fig. 7.3 Computational grid (zoomed) around RAE2822 airfoil

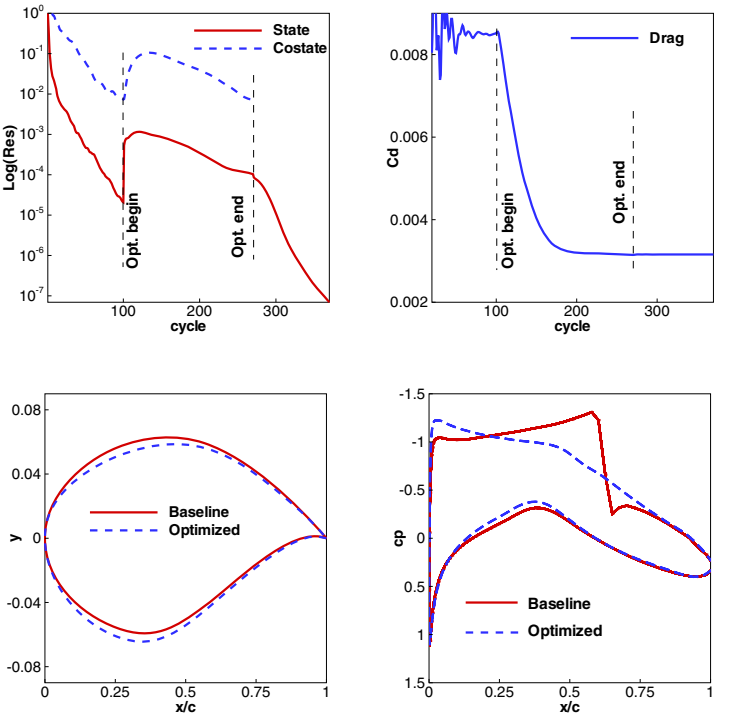


Fig. 7.4 Convergence history of the optimization iterations (top) and comparison of the geometries and surface pressure distributions (bottom) for Case 1

One of the main issues of using this kind of preconditioned pseudo-time-stepping is the approximation of the reduced Hessian. A better approximation will lead to faster convergence of the optimization algorithm. In the current study we compare the use of three different approximations. For L-BFGS updates of Case 3, we have used 4 different values of m={3,6,9,11} and denoted by notations lm3, lm6, lm9 and lm11 respectively. Table 7.1 presents the number of iterations required for the convergence of the optimization problem around a local minima (where a shock free airfoil results) and comparison of the force coefficients of the baseline and the optimized geometries of all the cases.

In case of inverse reduced Hessian approximation of Case 1, convergence of the optimization is achieved after 170 time-steps. The convergence history is presented in Figure 7.4 (top). Figure 7.4 (bottom) presents the baseline and optimized airfoils and the surface pressure distributions.

In case of inverse reduced Hessian approximation of Case 2, the convergence of the optimization is achieved after 130 iterations. Figure 7.5 (top) presents the optimization convergence history of this case. Also presented in Figure 7.5 (bottom) is a comparison of baseline and optimized airfoils and surface pressure distributions.

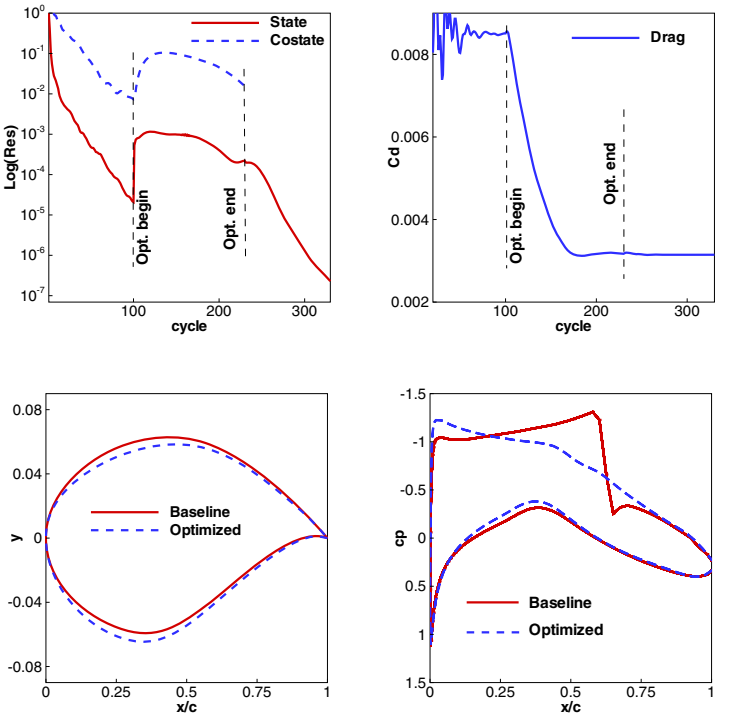


Fig. 7.5 Convergence history of the optimization iterations (top) and comparison of the geometries and surface pressure distributions (bottom) for Case 2

In case of inverse reduced Hessian approximations based on L-BFGS updates of Case 3, we have started the optimization iterations with little more accurate state and costate solutions obtained after 150 iterations. For the approximation of H_k^0, the initial approximation, if we use

$$H_k^0 = \frac{s_{k-1}^T z_{k-1}}{z_{k-1}^T z_{k-1}},$$

as suggested in [132] (page 226), the convergence of the optimization problem is very slow. Therefore, we have used $H_k^0 = B_k^{-1}$ where B_k is defined in the approximation used in Case 2 with different constants. Also in first $(m-1)$ iterations we have used that approximation for H_k in all the L-BFGS updates. The design equation update (step (3) of Algorithm 1) is done as

$$q^{k+1} = q^k - \mu \Delta t \rho,$$

where μ (> 1) is a constant.

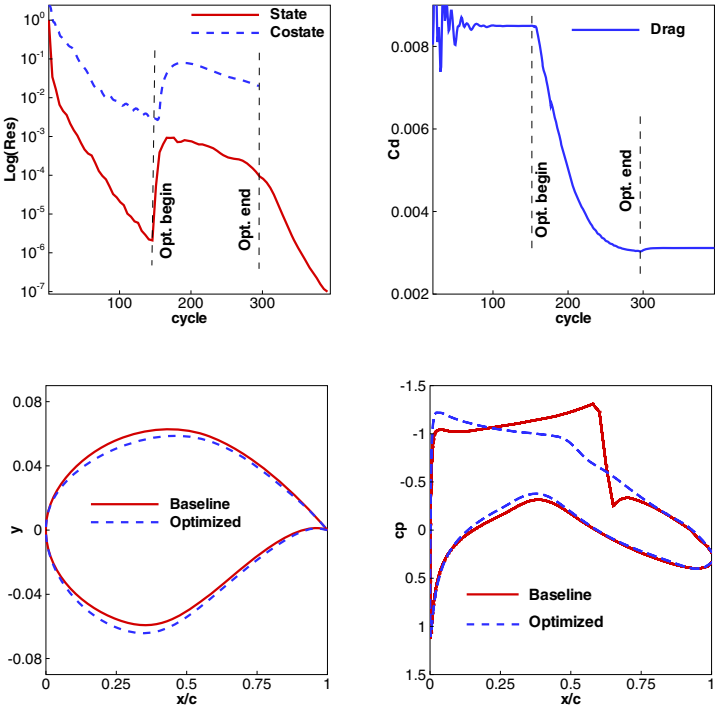


Fig. 7.6 Convergence history of the optimization iterations (top) and comparison of the geometries and surface pressure distributions (bottom) for lm3

For the case with $m = 3$ (lm3, in our notation), the convergence of the optimization is achieved in 145 iterations. Figure 7.6 (top) presents the optimization convergence history of this case. Figure 7.6 (bottom) presents a comparison of the baseline and the optimized airfoils and surface pressure distributions.

In case of inverse reduced Hessian approximation of Case 3 with $m = 6$ (lm6, in our notation), the convergence of the optimization is achieved after 135 iterations. Figure 7.7 (top) presents the optimization convergence history of this case. Figure 7.7 (bottom) presents a comparison of the baseline and the optimized airfoils and surface pressure distributions.

In case of inverse reduced Hessian approximation of Case 3 with $m = 9$ (lm9 in our notation), the convergence of the optimization is achieved after 130 iterations. Figure 7.8 (top) presents the optimization convergence history of this case. Figure 7.8 (bottom) presents a comparison of the baseline and the optimized airfoils and surface pressure distributions.

In all the cases of pseudo-time optimization iterations, the drag reduction is about 63%. The lift and pitching moment coefficients are also presented in Table 7.1. Since there is no constraint on these two quantities, they are also reduced by about 10% and 18% respectively.

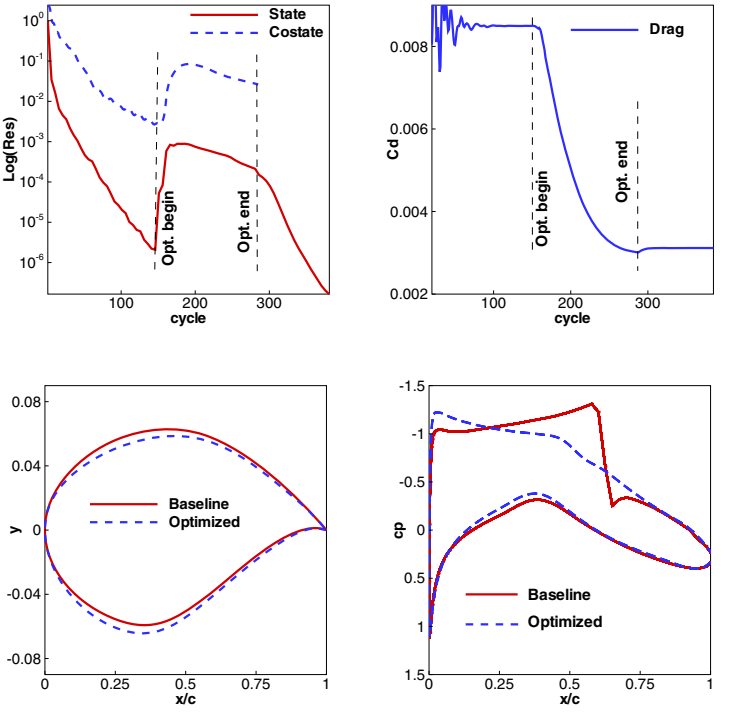


Fig. 7.7 Convergence history of the optimization iterations (top) and comparison of the geometries and surface pressure distributions (bottom) for lm6

As we see from the computational results, the reduced Hessian approximation in Case 2 leads to the convergence of the optimization problem in 130 iterations. In L-BFGS updates we achieve the convergence for $m = 9$ in the same number of iterations and for $m = 11$ in 128 iterations. However, in these cases total number of iterations are little more since we have to start with more accurate initial state and costate solutions. Also we need more storage and some additional multiplications in each optimization iterations. But the total number of optimization iterations are not very different. The total computational time with Case 2 updates is the minimum. From these computations we can say that the reduced Hessian approximation based on most recent gradients and most recent parameter updates is good enough for this class of problems. Hence, in all the applications mentioned hereafter in this chapter as well as in the subsequent chapters, we use the reduced Hessian updates of Case 2. In this case we need total computational effort which is just twice as that of forward simulation runs. This is a huge reduction of computational cost in compared to the 'traditional' gradient methods.

Figure 7.9 presents the baseline and the optimized camberlines and airfoils resulted in all the cases. We do not observe any noticeable difference in all of the optimized profiles.

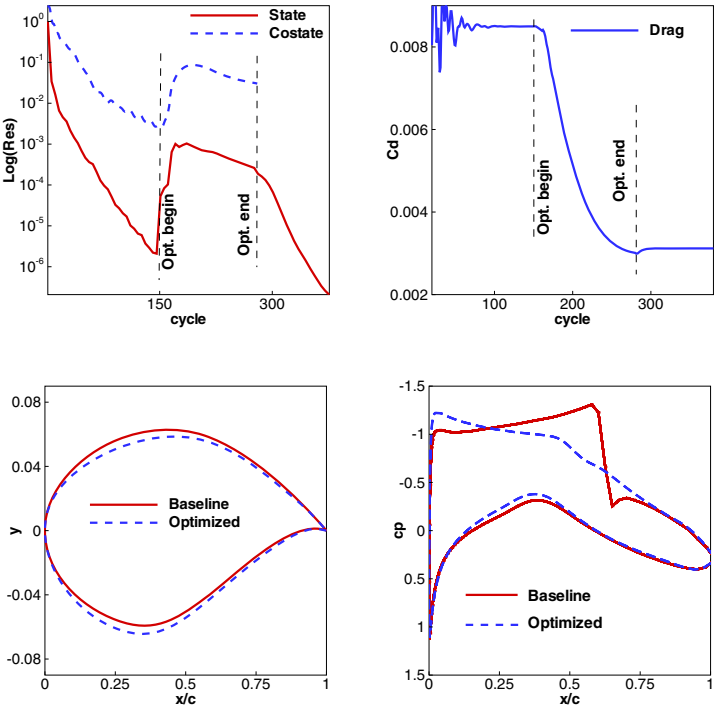


Fig. 7.8 Convergence history of the optimization iterations (top) and comparison of the geometries and surface pressure distributions (bottom) for lm9

Table 7.1 Comparison of number of iterations and force coefficients for baseline and optimized airfoil using different inverse Hessian approximations

Geometry	Iter	CD	CL	CM	Time
Baseline		0.849651E-02	0.826399E+00	0.126806E+00	
Opt.(Case 1)	170	0.315774E-02	0.752009E+00	0.106900E+00	1m48.615s
Opt.(Case 2)	130	0.314641E-02	0.746177E+00	0.105484E+00	1m30.268s
Opt.(lm=3)	145	0.311610E-02	0.748009E+00	0.104793E+00	1m50.293s
Opt.(lm=6)	135	0.311548E-02	0.746973E+00	0.104598E+00	1m43.606s
Opt.(lm=9)	130	0.311848E-02	0.746884E+00	0.104560E+00	1m40.835s
Opt.(lm=11)	128	0.311959E-02	0.746817E+00	0.104535E+00	1m41.624s

Figure 7.10 represents the contour plots of the Mach and the surface pressure at the initial condition and after the optimization (in Case 2). The initial shock, which causes the major drag in transonic regimes, has disappeared completely after the optimization.

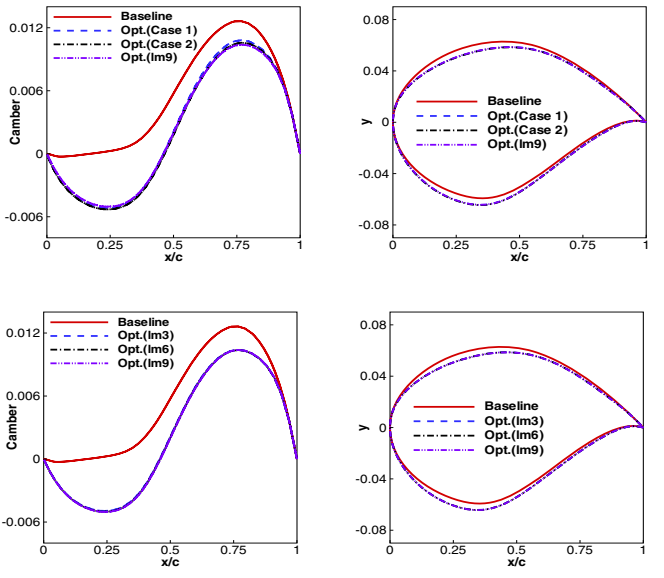


Fig. 7.9 Comparison of baseline and optimized camberlines and airfoils for Case 1, Case 2, lm9 (top) and for lm3, lm6, lm9 (bottom)

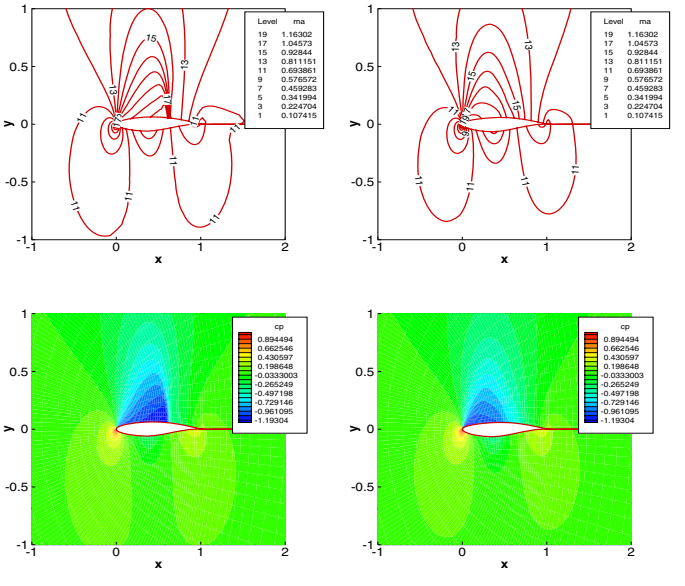


Fig. 7.10 Comparison of baseline and optimized Mach contours (top) and pressure contours (bottom)

7.6.2 *Drag Reduction with Geometric Constraints for Supersonic Cruise Transport (SCT) Wing*

In this case optimization is carried out for drag reduction with geometric constraints for an SCT wing at Mach number 2.0 and angle of incidence 3.22949 degrees. The geometric constraints are taken care via the parameterization of the wing. The physical domain is discretized by a grid of C-H topology consisting of $(97 \times 17 \times 25)$ grid points (Figure 7.11). The wing is decomposed into thickness, camberline and twist distributions for parameterization purposes (Figure 7.12, see [70] for details). The resulting wing is constructed by linear lofting of the modified wing sections. The thickness deformation has been based on B-splines which set free the range and the chord wise position of the maximum thickness, the leading edge radius and the trailing edge angle at 8 wing sections. The position of these sections are chosen according to the span-wise distribution of the geometrical constraints on maximum thickness. The camber line has been modified by adding 10 Hicks-Henne functions at 8 wing sections. The twist distribution has been described by a Bezier curve defined by 10 nodes. The center of rotation for the twist has been set at the leading edge of the wing. A total of 122 design variables are used to change the twist, the

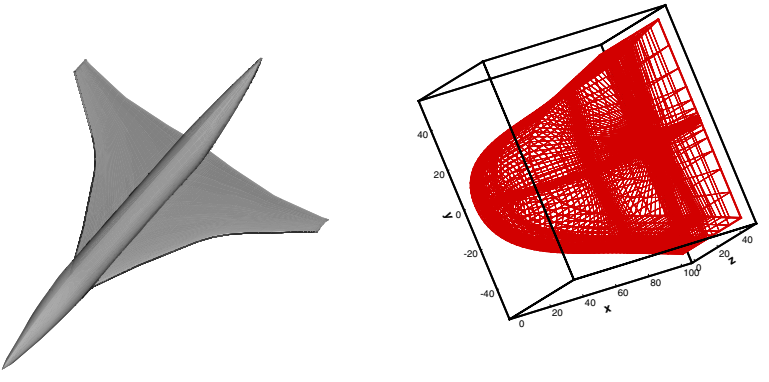


Fig. 7.11 SCT aircraft (left) and grid of C-H topology around the wing (right)

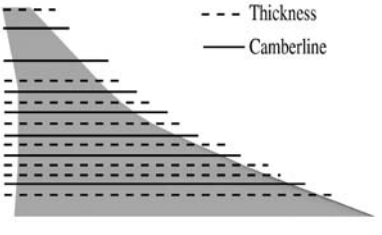


Fig. 7.12 Parameterization of the wing

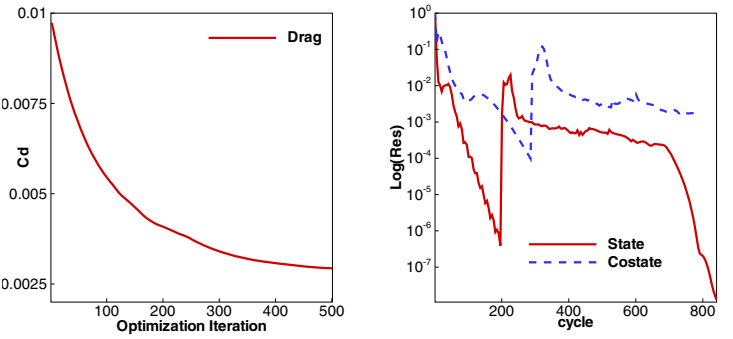


Fig. 7.13 Convergence histories of the optimization iterations for the wing

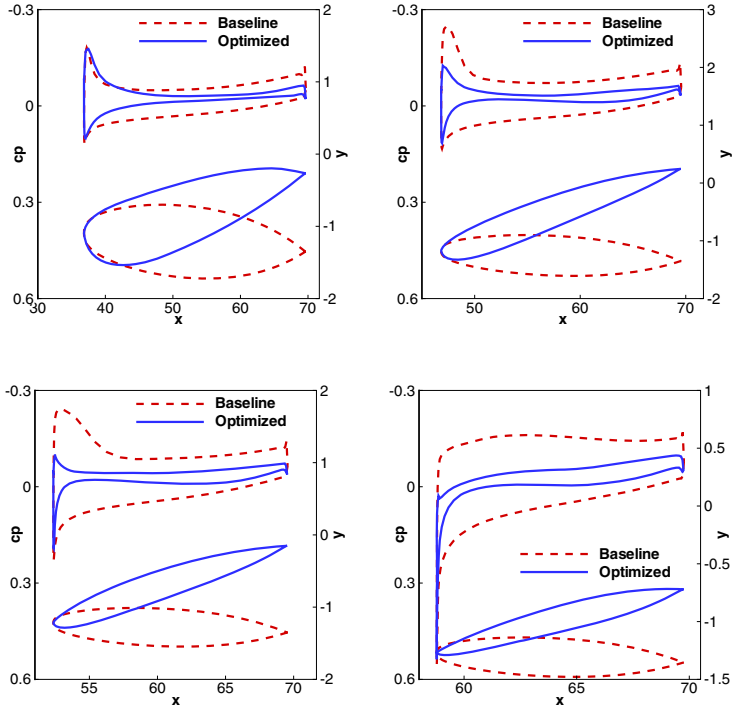


Fig. 7.14 Comparison of initial and optimized wing-sections and pressure distributions at 4 different sections $\eta = 0.24, 0.39, 0.49, 0.70$ (from top-left to bottom)

thickness and the camber line. Complete optimization cycle is performed under the optimization platform SynapsPointerPro [44].

We start the optimization iteration with initial solution (i.e., w_0 and λ_0) obtained after 200 time steps of the state and 250 time steps of costate equations. A total of

Table 7.2 Comparison of force coefficients for baseline and optimized wing

Geometry	Iter	CD	CL	CM
Baseline		0.972837E-02	0.120660E+00	0.350336E-01
Optimized	500	0.293458E-02	0.452202E-01	0.473493E-02

500 optimization iterations are carried out for drag reduction. Convergence histories of the drag as well as state and costate solutions are presented in Figure 7.13. Figure 7.14 presents a comparison of initial and final geometries and pressure distributions at 4 different span-wise sections. From the pressure distributions in the figure, we see that the pressure peak is reduced almost all over the wing. Table 7.2 presents a comparison of baseline and optimized force coefficients.

7.7 Conclusions

The new method has been applied to aerodynamic shape optimization which is based on simultaneous pseudo-time-stepping. The preconditioned pseudo-stationary state, costate and design equations are integrated simultaneously in time until a steady state is reached. The preconditioner used in this study is motivated by a continuous re-interpretation of rSQP methods. A better approximation of the reduced Hessian in the preconditioner leads to faster convergence of the optimization problem. Problems of drag reduction with only geometric constraints lead to non-unique solutions. However, successful implementation of the method has been possible for these problems both in 2D as well as in 3D where, in all the cases, a local minimum could be reached. The overall cost of computation is approximately 15% of that of a straight forward application of the steepest descent method. The generalization of the proposed strategy to problems with state constraints (e.g., drag reduction with constant lift) is addressed in the subsequent chapters.

Chapter 8
Indirect Treatment of State Constraints in Aerodynamic Shape Optimization Using Simultaneous Pseudo-Time-Stepping

8.1 Introduction

Practical shape design problems involve additional state constraints, e.g., shape optimization of aircraft for drag reduction with constant lift. Effectiveness of the optimization methods for aerodynamic shape design depends crucially on proper choice of the cost function, the constraints and their treatment during the optimization. One can treat the constraints 'indirectly', e.g., by making some kind of transformation so that the constraints are added to the objective function (with some weighting) and the constrained problem is reduced to an unconstrained one. This kind of treatment is termed as 'soft' constraints in [49]. Indirect treatment of constraints using traditional gradient methods can be found, among others, in [103, 142, 143].

Since problems of practical applications involve additional state constraints, we extend the simultaneous pseudo-time-stepping method to such problems in this chapter.[1] The constraint is treated indirectly, that is by adding it to the objective function. The correspondence to a rigorous treatment is explained below. We present applications to wing and body optimizations of an SCT aircraft for drag reduction with constant lift. The number of iterations required for the full optimization problem is about 8 times that of the analysis problem. This means a drastic reduction of the computational cost compared to traditional 'black-box' gradient methods.

8.2 Pseudo-Time-Stepping for the Constrained Optimization Problem

We rewrite the shape optimization problem that we are dealing with in this study in abstract form as

$$\begin{aligned} &\min I(w,q,\alpha) \\ \text{s. t.} \quad &\mathbf{c}(w,q,\alpha) = 0, \\ &h(w,q,\alpha) = h_0, \end{aligned} \tag{8.1}$$

[1] Materials presented in this chapter can also be found in [70], reprinted with kind permission of Walter de Gruyter.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 105–116.
springerlink.com

where, as in Chapter 6, $(w,q,\alpha) \in W \times Q \times \mathbb{R}$ (W, Q are appropriate Hilbert spaces), $I : W \times Q \times \mathbb{R} \to \mathbb{R}$, $h : W \times Q \times \mathbb{R} \to \mathbb{R}$ and $\mathbf{c} : W \times Q \times \mathbb{R} \to Y$ is twice Frechet-differentiable (with Y an appropriate Hilbert space). The Jacobian, $\mathbf{J} = \dfrac{\partial \mathbf{c}}{\partial w}$, is, as before, assumed to be invertible. Here, the equation $\mathbf{c}(w,q,\alpha) = 0$ represents the steady-state flow equations (in the present chapter, the Euler equations) together with the boundary conditions, $w \in W$ is the vector of dependent variables, $q \in Q$ is the vector of design variables and $\alpha \in \mathbb{R}$ is the angle of incidence. The objective $I(w,q,\alpha)$ is the drag and $h(w,q,\alpha)$ is the lift on the aircraft for the purposes of this chapter. We distinguish α from the geometric design parameters q since it is changed between the optimization iterations to achieve the constraint of constant lift using a procedure corresponding to the discussion below. Due to this reason, we present here the objective functional, the state equation and the constraint as explicit function of α.

The lift constraint is typically of the form

$$h(w,q,\alpha) \geq h_0.$$

Since we know apriori that this constraint is active, we treat it as equality constraint.

In this chapter we adopt the indirect way of treating the additional state constraint of constant lift as discussed and applied to practical problems in [143, 103]. The direct treatment of the state constraint requires an additional adjoint solution for the lift constraint which is discussed in detail in the next chapter. In this problem class, a prescribed lift can be achieved by changing the angle of incidence, as long as this lift value is less than the maximum possible lift value of the geometry.

The typical aerodynamic computational procedure to achieve this is (for fixed q):
1) choose h_0, the reference lift
2) adjust α so that $h(w(\alpha),q,\alpha) = h_0$
3) evaluate new drag $I(w(\alpha),q,\alpha)$.
Hence, one can define a mapping $\phi : h_0 \mapsto \alpha$ defined implicitly by $h(w(\alpha),q,\alpha) = h_0$ and thus there is a mapping from h_0 to I defined by the chain $h_0 \overset{\phi}{\mapsto} \alpha \overset{I}{\mapsto} I(w(\alpha),q,\alpha)$. From this, one can determine

$$\frac{dI}{dh_0} = \frac{dI}{d\alpha} \cdot \frac{d\phi}{dh_0} = \frac{dI}{d\alpha} \Big/ \frac{dh}{d\alpha} = \left(\frac{\partial I}{\partial w}\frac{\partial w}{\partial \alpha} + \frac{\partial I}{\partial \alpha} \right) \Big/ \left(\frac{\partial h}{\partial w}\frac{\partial w}{\partial \alpha} + \frac{\partial h}{\partial \alpha} \right). \quad (8.2)$$

The second equality is due to the fact that $\dfrac{d\phi}{dh_0} = \left(\dfrac{dh}{d\alpha} \right)^{-1}$. The computational implementation uses finite differences as

$$\frac{\partial I}{\partial w}\frac{\partial w}{\partial \alpha} + \frac{\partial I}{\partial \alpha} \approx \left(I(w(\alpha+\varepsilon),q,\alpha+\varepsilon) - I(w(\alpha),q,\alpha) \right)/\varepsilon, \quad (8.3)$$

where $w(\alpha+\varepsilon)$ is another flow solution with slightly perturbed angle of incidence. Similarly,

$$\frac{\partial h}{\partial w}\frac{\partial w}{\partial \alpha}+\frac{\partial h}{\partial \alpha}\approx\left(h(w(\alpha+\varepsilon),q,\alpha+\varepsilon)-h(w(\alpha),q,\alpha)\right)/\varepsilon. \tag{8.4}$$

From this deduction, we derive the formal mathematical description of a reduced problem below.

The necessary optimality conditions for problem (8.1) can be formulated using the Lagrangian functional

$$L(w,q,\alpha,\lambda,\mu)=I(w,q,\alpha)-\lambda^{\top}\mathbf{c}(w,q,\alpha)-\mu^{\top}\left(h(w,q,\alpha)-h_0\right), \tag{8.5}$$

where λ and μ are primal representations of the Lagrange multipliers $\lambda^{\top}$ and $\mu^{\top}$ from the dual Hilbert space. If $\hat{z}=(\hat{w},\hat{q},\hat{\alpha})$ is a minimum, then there exists $\hat{\lambda}$ and $\hat{\mu}$ such that

$$\nabla_z L(\hat{z},\hat{\lambda})=\nabla_z I(\hat{z})-\hat{\lambda}^{\top}\nabla_z\mathbf{c}(\hat{z})-\hat{\mu}^{\top}\nabla_z h(\hat{z})=0. \tag{8.6}$$

Hence, the necessary optimality conditions are

$$\nabla_w L(w,q,\alpha,\lambda,\mu)=0, \tag{8.7a}$$
$$\nabla_q L(w,q,\alpha,\lambda,\mu)=0, \tag{8.7b}$$
$$\nabla_\alpha L(w,q,\alpha,\lambda,\mu)=0, \tag{8.7c}$$
$$\mathbf{c}(w,q,\alpha)=0, \tag{8.7d}$$
$$h(w,q,\alpha)=h_0. \tag{8.7e}$$

It is well known from optimization theory that $\mu=\dfrac{dI}{dh_0}$, thus representing the effect of a perturbation in h_0 on I, in the solution. Also, it is to be noted, as mentioned in Chapter 6, that the statement of the optimality conditions is formal for the target problems, both in the function space setting as well as for the discretized problems in this problem class.

From an implementation point of view, it is advantageous to decouple the necessary conditions (8.7a-8.7e) in a block-Gauss-like algorithm:
step 1: estimate $\mu=\frac{dI}{dh_0}$ (using equation (8.2))
step 2: solve (8.7b,8.7d) for given α and μ from step 1
step 3: solve (8.7e) for α, goto step 1.

If this algorithm converges, then it solves (8.7a-8.7e). The equations in step 2 can be reformulated as an optimization problem

$$\begin{aligned}\min_{(w,q)}\ & I(w,q,\alpha)-\mu\left(h(w,q,\alpha)-h_0\right)\\ \text{s. t.}\quad & \mathbf{c}(w,q,\alpha)=0,\end{aligned} \tag{8.8}$$

for (μ,α) given. The objective function incorporates the lift constraint in a 'Lagrangian' way to avoid the drag reduction by reducing the lift. Due to non-linearity of the problem, there is still some loss of the lift, which is recovered by changing the

angle of incidence. Note that (8.8) looks like a penalty representation, but it differs from that insofar as μ does not tend towards ∞.

Hence, for given μ and α the necessary optimality conditions are

$$\nabla_w L(w,q,\lambda) = 0, \tag{8.9a}$$
$$\nabla_q L(w,q,\lambda) = 0, \tag{8.9b}$$
$$\mathbf{c}(w,q) = 0, \tag{8.9c}$$

corresponding to (8.7a,8.7b,8.7d). 'Black-box' gradient methods, which are widely used in many practical applications, involve well converged solutions of the state and the costate equations at each update of the design variables. This leads to high computational costs of these methods, especially, when applied to 3D problems.

We use the 'one-shot' or simultaneous pseudo-time stepping method for solving the above problem (8.9). This means we look for the steady states of an overall evolution equation system of the form

$$\begin{aligned} \frac{dw}{dt} + \mathbf{c}(w,q) &= 0, \\ \frac{d\lambda}{dt} + \nabla_w L(w,q,\lambda) &= 0, \\ \frac{dq}{dt} + \nabla_q L(w,q,\lambda) &= 0. \end{aligned} \tag{8.10}$$

As discussed earlier, this pseudo-time embedded system (after semi-discretization) is usually a stiff system of ordinary differential equations (ODEs). Explicit time-stepping schemes (which are used in most applications of this problem class) may converge very slowly or may even diverge. In order to accelerate convergence, this system needs preconditioning. The preconditioner that we use stems from reduced SQP methods as discussed in detail in previous chapter, as well as in [67, 71]. The resulting preconditioned system of differential equations that we consider is

$$\begin{pmatrix} \dot{w} \\ \dot{q} \\ \dot{\lambda} \end{pmatrix} = \begin{bmatrix} 0 & 0 & \mathbf{A}^\top \\ 0 & B & \left(\dfrac{\partial \mathbf{c}}{\partial q}\right)^\top \\ \mathbf{A} & \dfrac{\partial \mathbf{c}}{\partial q} & 0 \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -\mathbf{c} \end{pmatrix}. \tag{8.11}$$

Within the approximate reduced SQP-preconditioner, one has to look for an appropriate approximation of the reduced Hessian B. In this implementation, the reduced Hessian approximation is based on the most recent reduced gradient and parameter update informations as discussed in Chapter 7 (Case 2).

The one-shot algorithm: We compute a-priori μ using equation (8.2) in which the derivatives are computed using finite-difference formulas (equations (8.3),(8.4)). The cost of this computation is about 2 forward simulation runs. Since the variation of μ in each new geometry is very small, we keep the value of μ as constant till

the convergence of the optimization problem. This has a negligible effect on our inexact iterations. The overall algorithm reads as follows:

Algorithm

(0) Set $k := 0$; for a given μ and α start at some initial guess w_0, λ_0, q_0.

(1) Compute λ^{k+1} using a 5-stage Runge-Kutta time-stepping as discussed in Chapter 7

(2) Determine some approximation B_k of the reduced Hessian of the Lagrangian.

(3) March the design equation one step in time using

$$q^{k+1} = q^k - \Delta t B_k^{-1} \left(\nabla_q L - (\frac{\partial \mathbf{c}}{\partial q})^\top \lambda^{k+1} \right)$$

(4) Compute w^{k+1} using a 5-stage Runge-Kutta time-stepping as discussed in Chapter 7

(5) Change the angle of incidence α to satisfy the feasibility of the lift constraint

(6) Set $k := k+1$; go to (1) until convergence.

Remark: Step 5 is necessary due to the fact that the problem is highly nonlinear and $h(w,q,\alpha) = h_0$ is not maintained accurately enough without this step.

We use the 'FLOWer' [108, 109] code which has been modified and enhanced for 'one-shot' pseudo-time-stepping method. As before, the block matrices A and $A^\top$ in the preconditioner are the identity matrices. The method is a 'one-shot' method since we perform one time-step for each design update.

8.3 Numerical Results and Discussion

The complete optimization cycle is performed with the optimization platform SynapsPointerPro [44].

Case I (Wing Optimization): The optimization method is applied to a test case of an SCT aircraft (see Figure 7.11 (left)) wing optimization. The physical domain is discretized by a grid of C-H topology consisting of 41000 grid points (Figure 7.11 (right)). On this grid the preconditioned pseudo-unsteady equations are solved. The wing is decomposed into thickness, camberline and twist distributions for parameterization purposes (Figure 7.12). The resulting wing is constructed by linear lofting of the modified wing sections. The details of the parameterization is discussed in the previous chapter. A total of 122 design variables are used for this computation.

We first compute the (approximately constant) adjoint variable μ using the forward mode of FLOWer. We start the optimization iteration (i.e., w_0 and λ_0) with the solution obtained after 240 time steps of the state and 300 time steps of costate equations. The optimization requires 650 iterations to reach the convergence criterion (maximum norm of the design velocity $< 10^{-5}$). The drag reduction is 12.59%. Figure 8.1 presents the optimization convergence history of this case. The constant lift is maintained by changing the angle of incidence (step (5) in the *Algorithm*).

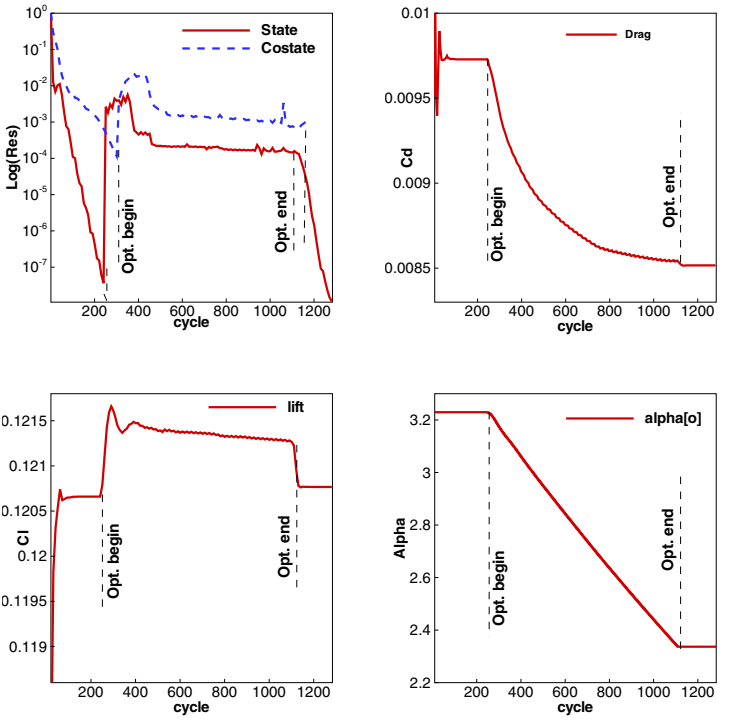


Fig. 8.1 Convergence history of the optimization iterations for the wing

Table 8.1 Comparison of force coefficients for baseline and optimized wing

Geometry	CL	CD	CM	Alpha
Baseline	0.120660E+00	0.972837E-02	0.350336E-01	0.322949E+01
Opt.(CG)	0.120661E+00	0.848634E-02	0.381244E-01	0.231456E+01
Opt.(One-Shot)	0.120661E+00	0.850397E-02	0.380276E-01	0.233355E+01

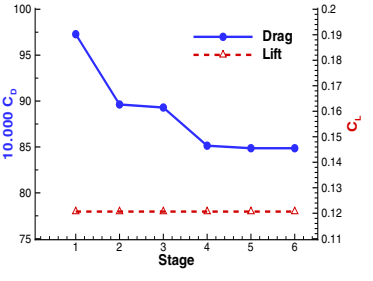


Fig. 8.2 Convergence history of the optimization for the wing using a black-box implementation of a nonlinear conjugate gradient method

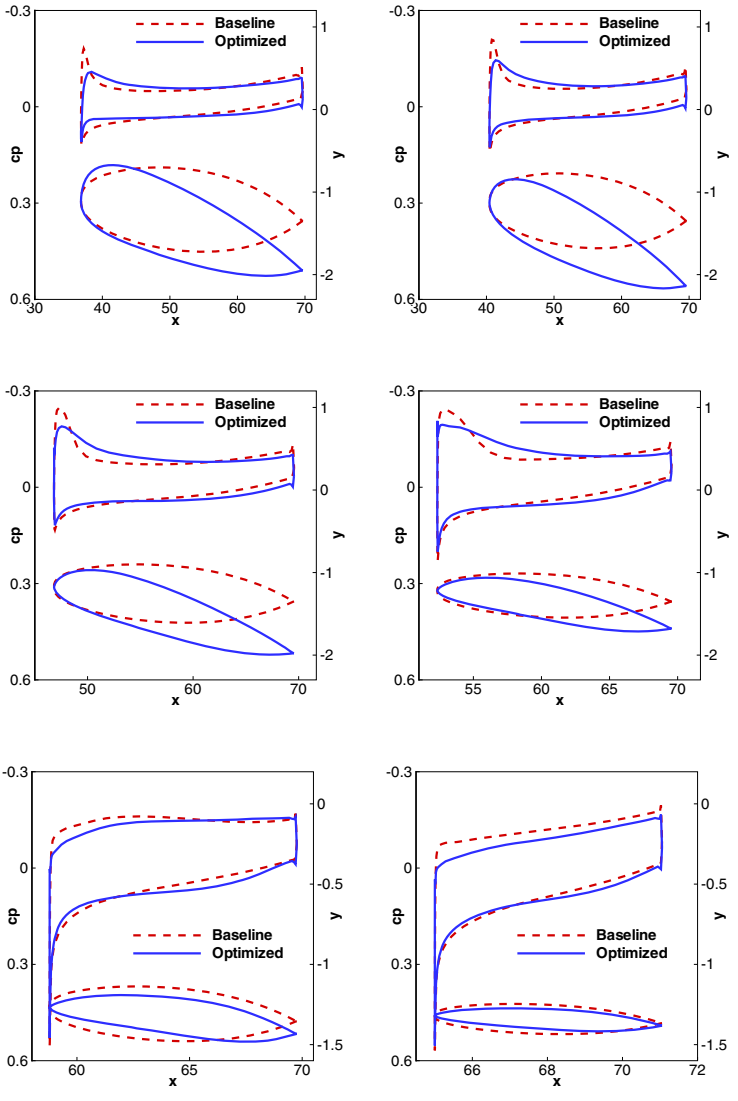


Fig. 8.3 Comparison of initial and optimized wing-sections and pressure distributions at 6 different sections $\eta = 0.24, 0.29, 0.39, 0.49, 0.70, 0.92$ (from top-left to bottom-right)

This facility is available in FLOWer. Since the change in the geometry is very small in one optimization iteration, we execute step (5) after every 3rd optimization iteration. To run the FLOWer code in this mode, we require two iterations in the forward (and also adjoint) solver (one for changing the angle and afterwards another one to get the solution using the current angle of attack). Therefore, the total number of state and costate iterations is 866 between optimization begin and its end. The computational effort to compute μ is about two times that of a full state solution. We

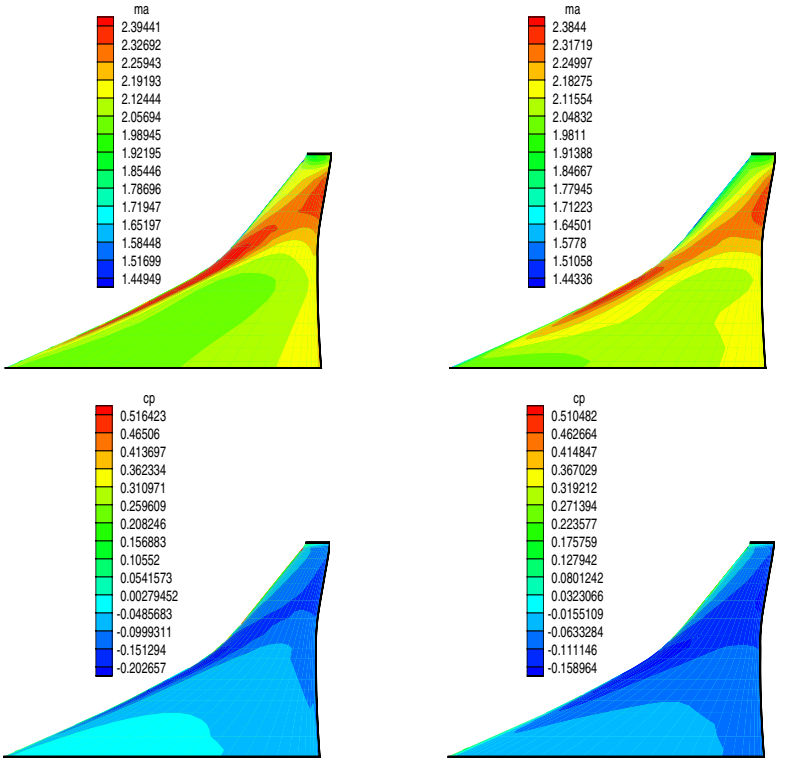


Fig. 8.4 Comparison of initial (left column) and final (right column) Mach (top) and pressure (bottom) contours

Table 8.2 Comparison of force coefficients for baseline and optimized body

Geometry	CL	CD	CM	Alpha
Baseline	0.119995E+00	0.997215E-02	0.363288E-01	0.322566E+01
Opt.(CG)	0.119995E+00	0.957578E-02	0.332888E-01	0.309017E+01
Opt.(One-Shot)	0.119996E+00	0.955603E-02	0.336248E-01	0.309277E+01

have additional overhead due to reading/writing the solution files, gradient computation and update of computational grid using grid deformation technique. The total effort of gradient computation and grid update is less than one flow computation. The read/write operation is currently performed after every forward and adjoint run. Each state solution takes about 400 iterations in the baseline and about 1000 iterations in the optimized geometry to converge (residual $< 10^{-7}$), this difference is due to changing the angle of incidence in baseline and optimized geometry to achieve the prescribed lift value. The adjoint solver needs about 1500 iterations to converge. If we count all the numbers necessary to obtain the optimized solution in the present case, the effort is equivalent to 8 forward simulation runs.

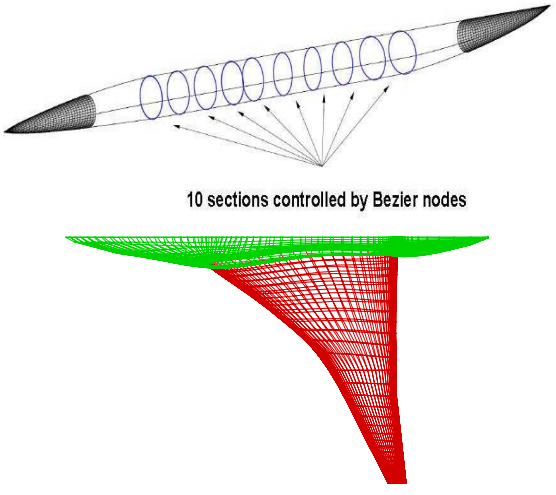


Fig. 8.5 Parameterization of the body and surface grid of the wing-body combination

Using the same FLOWer code and a nonlinear conjugate gradient optimization approach, the design optimization is carried out for the same geometry in [23]. The cost of computation in this approach is 39 state computations and 5 adjoint flow computations (see Figure 8.2). Table 8.1 presents a comparison of force coefficients for baseline and optimized wing (both using nonlinear CG and One-Shot methods). The results obtained in both methods are quite similar. However, the new optimization strategy leads to a reduction in computational effort by a factor of 5.5.

Figure 8.3 presents the comparisons of initial and final geometries and pressure distributions at 6 different span-wise sections. For the sections close to the root, the airfoils are mainly characterized by a round leading edge with a shift of the maximum thickness location to the leading edge. For the sections towards the wing tip, this change is reversed. From the pressure distributions in the same figure, we see that the pressure peak is reduced almost all over the wing. Figure 8.4 presents the contour plots of the Mach and the surface pressure at the initial condition and after the optimization on the wing.

Case II (Body Optimization): The optimization method is applied to a body optimization of the same aircraft. The fuselage contraction has been parameterized with 10 design variables which change the stream-wise law of the body radius (Figure 8.5). The body centerline has been kept unchanged during the optimization. The cross-section between the wing and the body has been recalculated after each optimization iteration using DLR's software MegaCads. MegaCads also generates a grid of the same topology (structured multi-block) consisting of 229425 cells.

The optimization iterations start at an initial state solution obtained after 600 time-steps, and a costate solution obtained after 1000 time steps. The optimization requires 90 iterations to converge. In this case step (5) of the *Algorithm* is executed after every second optimization iterations. The optimization convergence history is

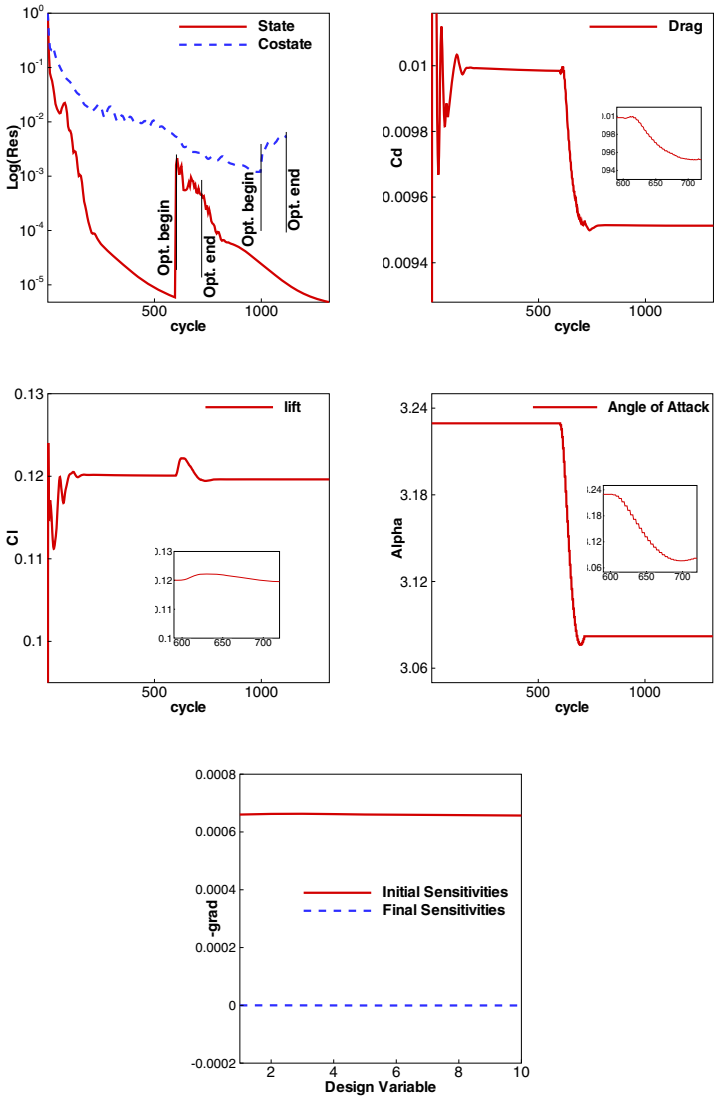


Fig. 8.6 Convergence history of the optimization iterations and comparison of the initial and final sensitivities for the body

presented in Figure 8.6. The drag reduction is about 4.17% in this case. The total number of state and costate iterations required (including the computation of μ) is about 4 forward simulation runs. The overhead of computing the gradients is considerably higher in compared to the state and costate simulation runs since the parameter sensitivities are computed by finite differences in MegaCads which by

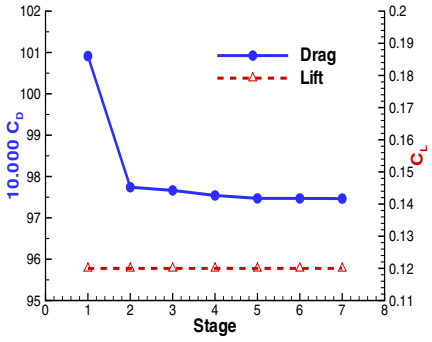


Fig. 8.7 Convergence history of the optimization for the body using a black-box implementation of a nonlinear conjugate gradient method

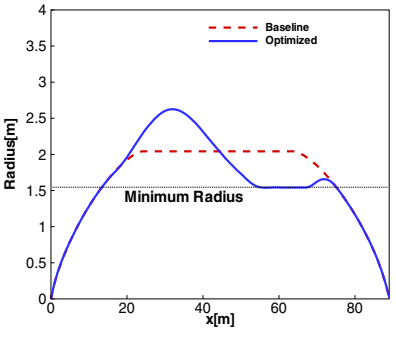


Fig. 8.8 Baseline and optimized body radius

itself is already an expensive tool for one parameter setting. This has been computed in parallel using 10 processors in different machines. The baseline and optimized body radius is presented in Figure 8.8.

The same computations were carried out for this case by using a nonlinear CG methods in [23]. There, 6 CG cycles are required, which consist of 40 state computations and 6 adjoint flow computations (see Figure 8.7). Table 8.2 presents a comparison of force coefficients for baseline and optimized (both using nonlinear CG and One-Shot methods) body. The results are quite similar in both optimization methods. In the traditional gradient method, the overhead of state and costate computations is much higher than our present approach, but the number of gradient computations are fewer than our current approach.

Finally, for comparison purposes, one forward flow solution is carried out combining the optimized wing with the optimized body. The drag reduction is about 11% compared to the baseline. Figure 8.9 presents the pressure and the Mach contours of the baseline and the optimized aircraft.

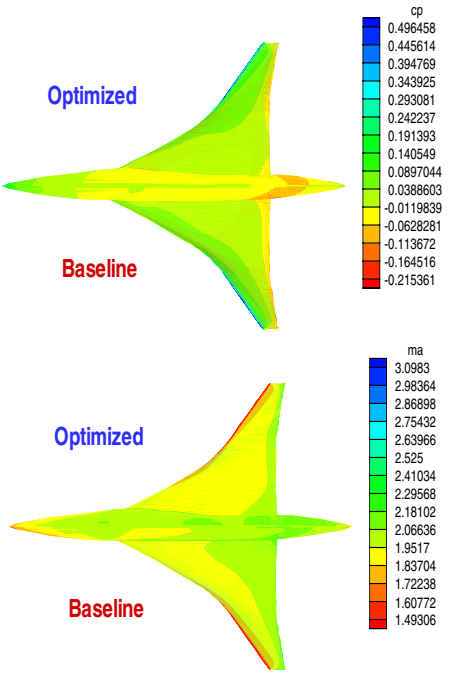


Fig. 8.9 Comparison of initial and final pressure (top) and Mach (bottom) contours

8.4 Conclusions

Simultaneous pseudo-time-stepping is used in conjunction with an rSQP preconditioner to solve the state constrained aerodynamic shape optimization problems in 3D. The total effort for the wing optimization is equivalent to about 8 forward flow solutions and for the body (with frozen wing) it is about 4 forward flow solutions. This is a huge reduction of computational cost in comparison to 'black-box' gradient methods.

Chapter 9
Direct Treatment of State Constraints in Aerodynamic Shape Optimization Using Simultaneous Pseudo-Time-Stepping

9.1 Introduction

In this chapter, we adopt a direct treatment of the state constraints in simultaneous pseudo-time-stepping method for the optimization problem. As discussed in the previous chapter, one can treat the constraints 'indirectly', e.g., by making some kind of transformation so that the constraints are added to the objective function (with some weighting) and the constrained problem is reduced to an unconstrained one. However, it is well known (see, for example, [43], subsection 7.2, pp.144) that the reduced problem may not correspond always to the original problem and the solution process may not be efficient one. Therefore, we discuss an alternative, direct way of treating the constraints.[1]

Direct treatment of state constraints using traditional gradient methods to such problems in 2D are carried out, among others, in [24, 47]. The computational effort required there is about 40 forward simulation runs and 27 adjoint runs (9 adjoint runs for each of the objective function and the state constraints). Direct treatment of a single state constraint using the 'one-shot' pseudo-time-stepping method has been carried out in [68, 69] for 2D problems. We have extended the method for two state constraints in [60]. The basic solution strategy is based on projecting the unconstrained design velocity onto the tangent space of the state constraints by solving a Quadratic Programming (QP) problem involving the reduced Hessian and the reduced gradients. Application examples for drag reduction with constant lift and constant pitching moment for an RAE2822 airfoil are included. In the previous chapter we have used the indirect treatment of additional state constraint using pseudo-time-stepping method. Also. we present here the results of the direct treatment of the state constraint for the same application example. The results show clear evidence of the advantages of the direct treatment of additional state constraints.

[1] Materials presented in this chapter can also be found in [60], reprinted with kind permission of American Institute of Aeronautics and Astronautics, Inc.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 117–133.
springerlink.com

9.2 Scalar State Constraints

We present the abstract formulation of the optimization problem with two additional state constraints. In that case the problem formulation (6.3) results in

$$\begin{aligned} &\min\ I(w,q) \\ \text{s. t.}\quad &\mathbf{c}(w,q) = 0, \\ &h_1(w,q) = 0, \\ &h_2(w,q) = 0, \end{aligned} \tag{9.1}$$

where, again as in Chapter 6, $(w,q) \in W \times Q$ (W, Q are appropriate Hilbert spaces), $I : W \times Q \to \mathbb{R}$ and $\mathbf{c} : W \times Q \to Y$ are twice Frechet-differentiable (with Y an appropriate Hilbert space). The Jacobian, $\mathbf{J} = \dfrac{\partial \mathbf{c}}{\partial w}$, is assumed to be invertible. The equation $\mathbf{c}(w,q) = 0$ represents a differential model equation (in this case the Euler equations) together with its boundary conditions, w is the vector of dependent variables and q is the vector of design variables. $h_1(w,q)$ is the lift on the airfoil/wing and $h_2(w,q)$ is the pitching moment on the airfoil for the purposes of this chapter.

Typically, in practical applications, additional constraints are of the form

$$h_i(w,q) \geq 0,$$

and represent the validity region of the model or the design construction. For the sake of simplicity in presentation we discuss only equality constraints, which is the case in our applications. Inequality constraints can be handled by an active set strategy, which is trivial for scalar inequalities. More difficult constraints, e.g., constraints in contact problems, which are not scalar valued, are out of the scope of this book.

One approximate Newton step for necessary conditions of problem (9.1) corresponding to an rSQP method is derived from the following system of equations

$$\begin{pmatrix} 0 & 0 & \left(\frac{\partial h_1}{\partial w}\right)^\top & \left(\frac{\partial h_2}{\partial w}\right)^\top & \mathbf{A}^\top \\ 0 & B & \left(\frac{\partial h_1}{\partial q}\right)^\top & \left(\frac{\partial h_2}{\partial q}\right)^\top & \left(\frac{\partial \mathbf{c}}{\partial q}\right)^\top \\ \frac{\partial h_1}{\partial w} & \frac{\partial h_1}{\partial q} & 0 & 0 & 0 \\ \frac{\partial h_2}{\partial w} & \frac{\partial h_2}{\partial q} & 0 & 0 & 0 \\ \mathbf{A} & \frac{\partial \mathbf{c}}{\partial q} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta q \\ \Delta \mu_1 \\ \Delta \mu_2 \\ \Delta \lambda \end{pmatrix} = \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -h_1 \\ -h_2 \\ -\mathbf{c} \end{pmatrix}, \tag{9.2}$$

where μ_1 and μ_2 are the Lagrange multipliers corresponding to the additional state constraints $h_1(w,q)$ and $h_2(w,q)$ respectively.

9.2.1 Partial Reduction of the Problem

Lemma 7. *The above system (9.2) can be reduced to a system*

$$\begin{pmatrix} B & g_{h_1} & g_{h_2} \\ g_{h_1}^\top & 0 & 0 \\ g_{h_2}^\top & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta q \\ \Delta\mu_1 \\ \Delta\mu_2 \end{pmatrix} = \begin{pmatrix} -\nabla_q L + \left(\dfrac{\partial \mathbf{c}}{\partial q}\right)^\top \mathbf{A}^{-\top} \nabla_w L \\ -h_1 + \dfrac{\partial h_1}{\partial w}\mathbf{A}^{-1}c \\ -h_2 + \dfrac{\partial h_2}{\partial w}\mathbf{A}^{-1}c \end{pmatrix}, \tag{9.3}$$

where $g_{h_1} = \left(\dfrac{\partial h_1}{\partial q} - \dfrac{\partial h_1}{\partial w}\mathbf{A}^{-1}\dfrac{\partial \mathbf{c}}{\partial q}\right)^\top$ *is the reduced gradient of the constraint* $h_1(w,q)$ *and* $g_{h_2} = \left(\dfrac{\partial h_2}{\partial q} - \dfrac{\partial h_2}{\partial w}\mathbf{A}^{-1}\dfrac{\partial \mathbf{c}}{\partial q}\right)^\top$ *is the reduced gradient of the constraint* $h_2(w,q)$.

Proof: In the following we use Gaussian elimination for partial reduction of the system of equations (9.2) by eliminating the variables Δw and $\Delta\lambda$ from the system. To do that, we consider the last equation and solve for

$$\Delta w = \mathbf{A}^{-1}\left(-\mathbf{c}(w,q) - \frac{\partial \mathbf{c}}{\partial q}\Delta q\right).$$

A substitution of this for Δw in the fourth equation results in

$$\left(\frac{\partial h_2}{\partial q} - \frac{\partial h_2}{\partial w}\mathbf{A}^{-1}\frac{\partial \mathbf{c}}{\partial q}\right)\Delta q = -h_2 + \frac{\partial h_2}{\partial w}\mathbf{A}^{-1}\mathbf{c}.$$

If we define

$$g_{h_2} := \left(\frac{\partial h_2}{\partial q} - \frac{\partial h_2}{\partial w}\mathbf{A}^{-1}\frac{\partial \mathbf{c}}{\partial q}\right)^\top = \begin{bmatrix} -\mathbf{A}^{-1}\dfrac{\partial \mathbf{c}}{\partial q} \\ I \end{bmatrix}^\top \begin{bmatrix} \nabla_w h_2 \\ \nabla_q h_2 \end{bmatrix},$$

as the reduced gradient of $h_2(w,q)$, then we get

$$g_{h_2}^\top \Delta q = -h_2 + \frac{\partial h_2}{\partial w}\mathbf{A}^{-1}\mathbf{c}. \tag{9.4}$$

Similarly, a substitution of Δw in the third equation results in

$$g_{h_1}^\top \Delta q = -h_1 + \frac{\partial h_1}{\partial w}\mathbf{A}^{-1}\mathbf{c}, \tag{9.5}$$

where g_{h_1} is the reduced gradient of $h_1(w,q)$. Next we consider the first equation and solve for

$$\Delta\lambda = \mathbf{A}^{-\top}\left(-\nabla_w L - (\frac{\partial h_1}{\partial w})^\top \Delta\mu_1 - (\frac{\partial h_2}{\partial w})^\top \Delta\mu_2\right).$$

A substitution of this for $\Delta\lambda$ in the second equation results in

$$B\Delta q + \left((\frac{\partial h_1}{\partial q})^\top - (\frac{\partial \mathbf{c}}{\partial q})^\top \mathbf{A}^{-\top}(\frac{\partial h_1}{\partial w})^\top\right)\Delta\mu_1 + \left((\frac{\partial h_2}{\partial q})^\top - (\frac{\partial \mathbf{c}}{\partial q})^\top \mathbf{A}^{-\top}(\frac{\partial h_2}{\partial w})^\top\right)\Delta\mu_2 = -\nabla_q L + (\frac{\partial \mathbf{c}}{\partial q})^\top A^{-\top}\nabla_w L.$$

This can be written in terms of the reduced gradients as

$$B\Delta q + g_{h_1}\Delta\mu_1 + g_{h_2}\Delta\mu_2 = -\nabla_q L + (\frac{\partial \mathbf{c}}{\partial q})^\top A^{-\top}\nabla_w L. \tag{9.6}$$

Writing Eqs.(9.4), (9.5) and (9.6) in matrix-vector notation results in the reduced system(9.3) □

9.2.2 Solution Strategy of the Constrained Problem

We use the above reduction strategy to solve problem (9.1) using simultaneous pseudo-time-stepping. The preconditioned pseudo-time embedded non-stationary system to be solved reads as

$$\begin{pmatrix} 0 & 0 & \left(\frac{\partial h_1}{\partial w}\right)^\top & \left(\frac{\partial h_2}{\partial w}\right)^\top & \mathbf{A}^\top \\ 0 & B & \left(\frac{\partial h_1}{\partial q}\right)^\top & \left(\frac{\partial h_2}{\partial q}\right)^\top & \left(\frac{\partial \mathbf{c}}{\partial q}\right)^\top \\ \frac{\partial h_1}{\partial w} & \frac{\partial h_1}{\partial q} & 0 & 0 & 0 \\ \frac{\partial h_2}{\partial w} & \frac{\partial h_2}{\partial q} & 0 & 0 & 0 \\ \mathbf{A} & \frac{\partial \mathbf{c}}{\partial q} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{w} \\ \dot{q} \\ \dot{\mu_1} \\ \dot{\mu_2} \\ \dot{\lambda} \end{pmatrix} = \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -h_1 \\ -h_2 \\ -\mathbf{c} \end{pmatrix}. \tag{9.7}$$

For the solution of this problem, we first solve system (7.1) for the design velocity $\dot{q}$. This design velocity is then projected onto the tangent space of the state constraints through the solution of the following QP

$$\begin{aligned} \min\ & \frac{1}{2}\dot{q}^\top B\dot{q} + g^\top \dot{q} \\ \text{s.t. } & g_{h_1}^\top \dot{q} = -h_1(w,q) + \frac{\partial h_1}{\partial w}\mathbf{A}^{-1}c, \\ & g_{h_2}^\top \dot{q} = -h_2(w,q) + \frac{\partial h_2}{\partial w}\mathbf{A}^{-1}c, \end{aligned} \tag{9.8}$$

where $g = \left(\nabla_q L - \left(\frac{\partial \mathbf{c}}{\partial q}\right)^\top \mathbf{A}^{-\top}\nabla_w L\right)$ is the reduced gradient of the Lagrangian.

Lemma 8. *The first order necessary conditions for solving (9.8) leads to the same system as in (9.3) with the unknowns replaced by $\dot{q}$ and $\dot{\mu}_1$, $\dot{\mu}_2$ (which are the Lagrange multipliers for the system (9.8)) respectively.*

Proof: Writing the Lagrangian functional of the system (9.8) and a straightforward calculation of the 1st order necessary conditions will lead to the result. □

Therefore, the above mentioned reduction can be interpreted as a projection of the design velocity from equation (7.1) towards the linearized state constraints $h_1(w,q)$ and $h_2(w,q)$, thus resembling dynamic projection strategies onto invariants as in [149]. For the construction of the reduced gradients g , g_{h_1} and g_{h_2}, one has to solve one adjoint problem (approximately) for g and one each for g_{h_1} and g_{h_2}. The reduced problem for a single additional state constraint can be found in [68].

9.2.3 Back Projection

Due to nonlinearity of the problem there is some deviation from the additional state constraints. Therefore we use the following correction strategy in each optimization step to avoid this deviation. We minimize the distance between the point q_0 and the manifold $\mathscr{S}$ of constant lift and constant pitching moment $\mathscr{S} = \{q \,|\, h_1(w(q)) = l_0^1, h_2(w(q)) = l_0^2\}$. This is done by solving ideally the problem

$$\begin{aligned} &\min \ \tfrac{1}{2}||q-q_0||^2 \\ &\text{s. t. } h_1(w(q)) - l_0^1 = 0, \\ &\qquad h_2(w(q)) - l_0^2 = 0. \end{aligned} \tag{9.9}$$

We use one step of a generalized Gauss-Newton method to solve this problem. Since the stiffness matrix of the flow equations is approximated by A when forming the reduced gradients g_{h_1} and g_{h_2}, we compute the step Δq from

$$\begin{aligned} &\min \ \tfrac{1}{2}||\Delta q||^2 \\ &\text{s.t. } g_{h_1}^\top \Delta q = -h_1(w,q) + \frac{\partial h_1}{\partial w}\mathbf{A}^{-1}c, \\ &\qquad g_{h_2}^\top \Delta q = -h_2(w,q) + \frac{\partial h_2}{\partial w}\mathbf{A}^{-1}c. \end{aligned} \tag{9.10}$$

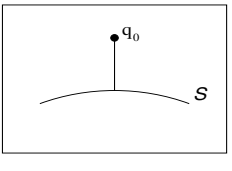


Fig. 9.1 Projection

The necessary optimality conditions to solve this problem will lead to the system of equations

$$\begin{pmatrix} I & g_{h_1} & g_{h_2} \\ g_{h_1}^\top & 0 & 0 \\ g_{h_2}^\top & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta q \\ \nu_1 \\ \nu_2 \end{pmatrix} = \begin{pmatrix} 0 \\ -(h_1(w,q) - \dfrac{\partial h_1}{\partial w}\mathbf{A}^{-1}c) \\ -(h_2(w,q) - \dfrac{\partial h_2}{\partial w}\mathbf{A}^{-1}c) \end{pmatrix},$$

where ν_1 and ν_2 are the Lagrange multipliers for the system (9.10). Solving this system gives Δq which is used to get the corrected step given by

$$q^{k+1} = q_0^{k+1} - \Delta q. \tag{9.11}$$

The overall algorithm reads as follows:

Algorithm 1: *The simultaneous pseudo-time-stepping for the preconditioned system*
(0) Set $k := 0$; start at some initial guess w_0, λ_0 and q_0.
(1) Compute λ^{k+1} marching one step in time for the adjoint equations.
(2) Compute sensitivities using state and adjoint solutions.
(3) Determine some approximation B_k of the projected Hessian of the Lagrangian.
(4) Solve the quadratic subproblem Eq.(9.8) to get $\dot{q}$.
(5) March in time one step for the design equation as follows:
$q_0^{k+1} = q^k + \Delta t \cdot \dot{q}$
(6) Use the correction step Eq.(9.11) for the new step.
(7) Compute w^{k+1} marching one step in time for the state equations.
(8) Set $k := k+1$; go to (1) until convergence.

In the current implementation, $A^{-1}c$ in the right hand side of the constraint in Eq.(9.8) is approximated by (scaled) $\dot{w}$ value from the previous iteration as it is updated only at step (7) of the above algorithm. Δt in step (5) of the algorithm is the minimum time step length of the forward solver, as mentioned in Chapter 7. Instead of the exact reduced gradient of the Lagrangian (e.g., in Eq.(9.8)), we use an approximation to it. It is a 'one-shot' method since we perform one time-step for each design update. The details of the governing equations, their discretization, gradient computation, surface parameterization and grid perturbation strategies can be found in Chapter 7 and also discussed in [70] (for 3D) and in [71] (for 2D).

9.3 Numerical Results and Discussion

The optimization method is applied to test cases in 2D as well as in 3D. The FLOWer code [108, 109], modified and enhanced for one-shot method, is used for solving the forward and adjoint equations. The design equation is integrated in time using an explicit Euler scheme and the state and costate equations are integrated in time using a 5-stage Runge-Kutta scheme. The reduced Hessian approximation is based on the current gradients and parameter update informations as discussed in Chapter 7 (Case 2).

9.3.1 Applications in 2D

The optimization method is applied to test cases of an RAE2822 airfoil for drag reduction with constant lift and constant pitching moment together with geometric constraint of constant thickness. The flow conditions are described at Mach number 0.73 and angle of incidence 2 degrees. The physical domain is discretized using an algebraically generated (193×33) C-grid. The airfoil is decomposed into thickness and camberline distributions for parameterization purposes. The parameters corresponding to the thickness have not been changed during optimization to satisfy the geometric constraint. The camberline has been parameterized by 21 Hicks-Henne [78] parameters.

The forward and adjoint solutions are computed using the multigrid solvers in FLOWer. Optimization iterations start with the initial solutions obtained after 150 time-steps of the state equations (w_0) and of the costate equations (λ_0). The optimization iterations stopped when $||\Delta q||_\infty < 0.0008$. After the convergence of optimization iterations, another 200 time-steps are performed for the state equation on the optimized geometry to reduce its residual further so that the force coefficients and surface pressure distribution can be compared with those obtained by other methods.

Case 1. Drag reduction with constant lift and constant pitching moment with 21 design parameters
In this case 500 iterations are required for the convergence of the optimization problem. Optimization convergence histories are presented in Figure 9.2. Figure 9.3 presents a comparison of initial and final airfoils and surface pressure distributions. As we see in this figure, the optimization results in a shockfree airfoil in this case. This is due to the fact that in the inviscid transonic regime, major amount of drag is caused by the shock. Hence, drag reduction results a shockfree airfoil. Table 9.1 presents a comparison of the force coefficients for baseline and optimized geometries. The drag reduction is 60.30%, the lift is well maintained with an increase by only 0.016% and the pitching moment is also well maintained with a decrease by 0.016%.

The forward solver requires about 350 time-steps to produce 'well' converged solution. Total number of time-steps required for this optimization is (150+500+200) (before optimization, during optimization, and after optimization respectively) for the state solver, and three times (150+500) (before optimization, and during optimization respectively) for the adjoint solver. These together are approximately 8 times that of the forward simulation runs. The effort to solve the design equation and gradient computations is almost negligible in comparison to state or costate solutions. Additionally, the simultaneous pseudo-time method needs a new grid, obtained by grid perturbation, after each optimization iteration. Additional time is required to write the output after every iteration and read the same before each iteration, as the iterations start with solution values from the previous iteration. If we add on everything, the total effort is less than 10 times that of the forward simulation runs.

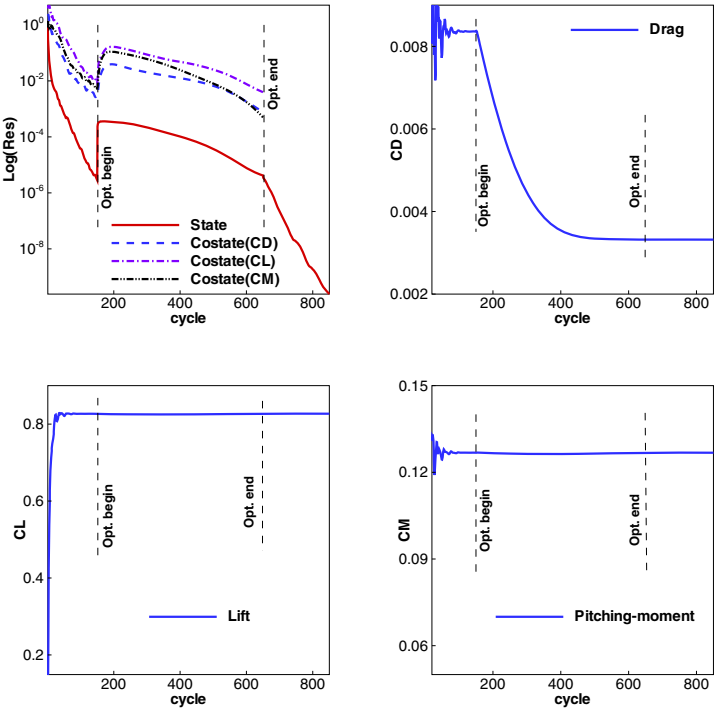


Fig. 9.2 Convergence history of the optimization problem of Case 1

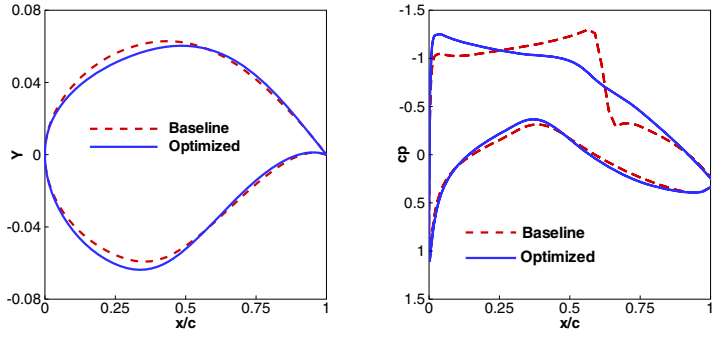


Fig. 9.3 Comparison of initial and final airfoils and surface pressure distributions of Case 1

Case 2. Drag reduction with constant lift and constant pitching moment with 40 design parameters

In this case we use 40 design parameters to parameterize the airfoil. The convergence of the optimization problem is faster, requiring only 300 iterations. Faster convergence of the method can be explained from the physics of the problem.

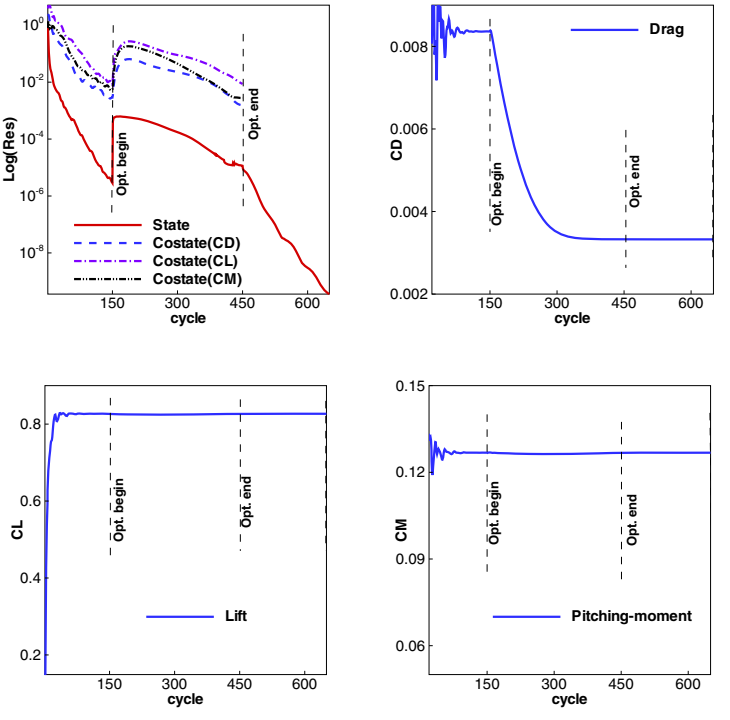


Fig. 9.4 Convergence history of the optimization problem of Case 2

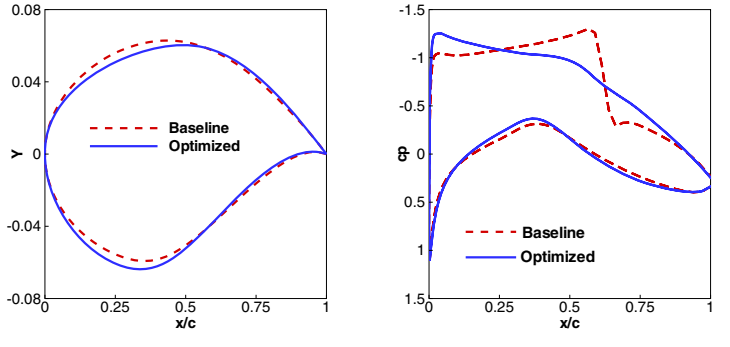


Fig. 9.5 Comparison of initial and final airfoils and surface pressure distributions of Case 2

The optimization problem deals with drag reduction together with additional state constraints. As mentioned, in inviscid transonic regime, major drag is caused by the shock jump. The task of the optimizer is to change the design parameters in such a way that the drag disappears. In case of finer parameterization, a small change in parameters has more effect on the shock than that of a coarser parameterization. To

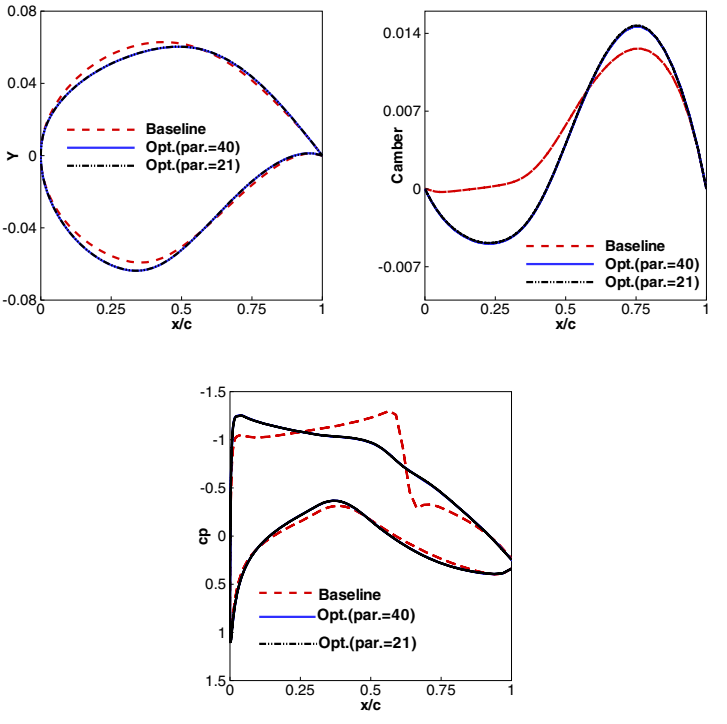


Fig. 9.6 Comparison of initial and final airfoils, camberlines and surface pressure distributions of Case 1 and Case 2

achieve the same effect in coarser parameterization one has to have larger change in the design parameters, which causes constraint violation. That is why in this particular problem class, and for this particular optimization method, finer parameterization leads to faster convergence.

Since we use continuous adjoint method, the cost of gradient computation is independent of number of design variables. Therefore, it is advantageous to use finer design space in the context of one-shot pseudo-time method. Optimization convergence histories are presented in Figure 9.4. Figure 9.5 presents a comparison of the initial and the final airfoils and surface pressure distributions. A comparison of force coefficients for the baseline and the optimized geometries are presented in Table 9.1. The drag reduction is 60.22%, the lift is well maintained by a decrease of only 0.005% and the pitching moment is also well maintained by a decrease of only 0.008%. The optimized force coefficients are almost the same in both cases.

Figure 9.6 presents a comparison of airfoils, camberlines and surface pressure distributions obtained in Case 1 and Case 2. There is no noticeable difference in them as well. However, the number of optimization iterations is little more than half

Table 9.1 Comparison of number of iterations and force coefficients for baseline and optimized airfoil for different number of design parameters (on 193×33 grid)

Geometry	Iter	CD	ΔC_D	CL	ΔC_L	CM	ΔC_M
Baseline		0.836150E-02		0.826810		0.12679	
Opt.(Case 1)	500	0.331930E-02	60.30%	0.826940	-0.016%	0.12677	0.016%
Opt.(Case 2)	300	0.332590E-02	60.22%	0.826770	0.005%	0.12678	0.008%

Table 9.2 Comparison of force coefficients for baseline and optimized airfoil in Case 3 computation (on 321×57 grid)

Geometry	Iter	CD	ΔC_D	CL	ΔC_L	CM	ΔC_M
Baseline		0.852910E-02		0.829500		0.12920	
Opt.(Case 3)	800	0.325770E-02	61.80%	0.829530	-0.004%	0.12923	-0.02%

of that of Case 1 and the total effort in this case is less than 7 times that of the forward simulation runs.

Case 3. Drag reduction with constant lift and constant pitching moment with 40 design parameters on 321×57 **grid**
In this case we study the fine grid optimization with computational grid around the airfoil being of size 321×57. Here also the optimization is started with initial state and costate solutions (w_0, λ_0) obtained after 150 time-steps. The optimization requires 800 iterations to converge. After the convergence of the optimization, another 200 time steps are carried out for the state solution to achieve sufficiently accurate force coefficients. Figure 9.7 presents the optimization convergence histories of this case. Figure 9.8 presents a comparison of the initial and final airfoils and surface pressure distributions. A comparison of the force coefficients for the baseline and optimized geometries are presented in Table 9.2. The drag reduction is 61.80% which is a bit more than the last two cases, the lift is again well maintained with an increase by only 0.004% and the pitching moment is also well maintained with an increase by only 0.02%. In this case the steady state forward solution is reached by 450 multigrid iterations and thus, if we add on everything, the total effort of the optimization problem is less than 11 forward simulation runs. Figure 9.9 presents the baseline and optimized pressure and Mach contours of this case.

Optimization of the same geometry has been carried out in [24] using traditional gradient methods. The drag reduction there is about 60.00% and the constraint violations are about 0.1%. The total effort required there consist of 40 forward simulation runs and 27 adjoint runs. In comparison to that, the current effort is a reduction of about 83%.

9.3.2 Application in 3D

In this case we apply our method to a 3D problem of the SCT wing optimization at supersonic Mach number 2.0 and angle of incidence 3.22949 degrees. The physical

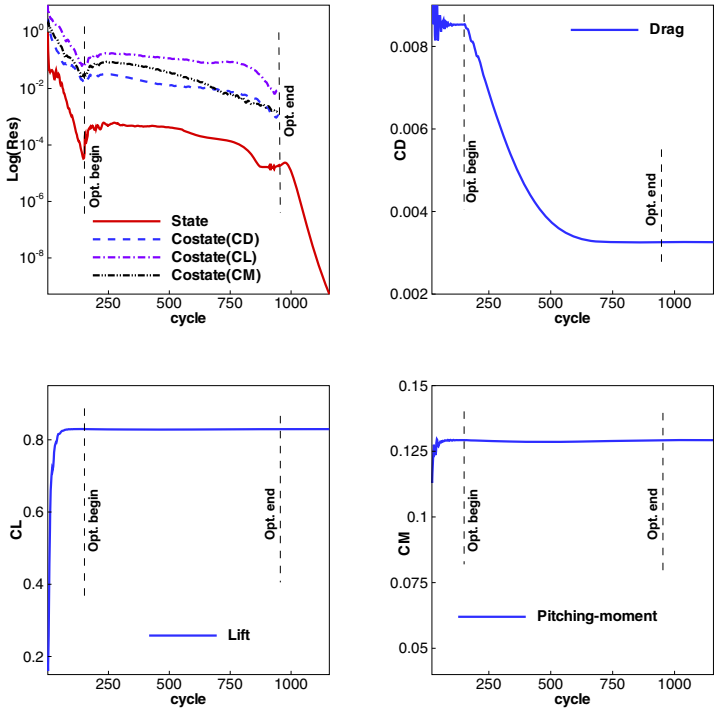


Fig. 9.7 Convergence history of the optimization problem of Case 3

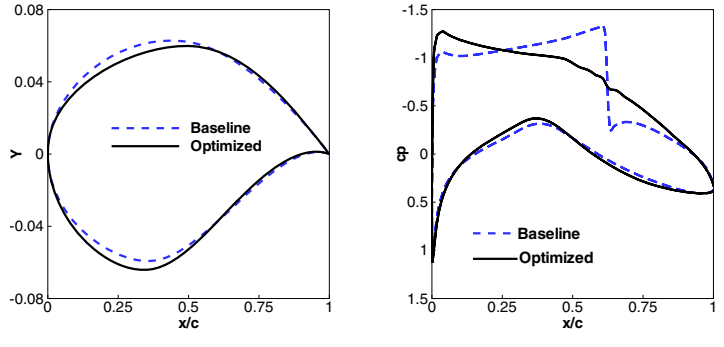


Fig. 9.8 Comparison of initial and final airfoils and surface pressure distributions of Case 3

domain is discretized into a grid of C-H topology consisting of $(97 \times 17 \times 25)$ grid points. The wing is decomposed into thickness, camberline and twist distributions for parameterization purposes as described in Chapter 7. A total of 122 design

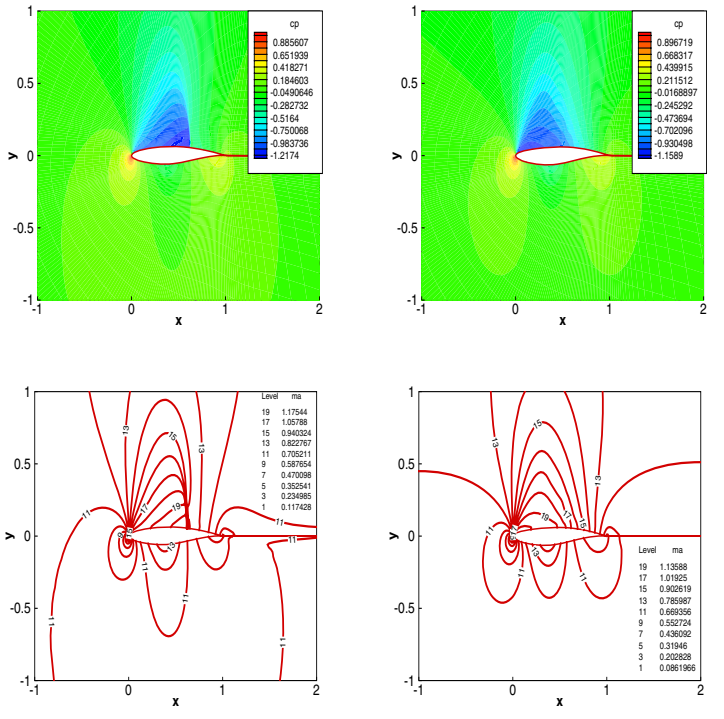


Fig. 9.9 Comparison of baseline (left) and optimized (right) pressure (top) and Mach (bottom) contours of Case 3

variables are used to change the twist, the thickness and the camberline at specific wing sections.

Case 4. Drag reduction with constant lift for a Supersonic Cruise Transport aircraft wing

The optimization is performed with initial state solution (w_0) obtained after 150 time-steps and initial costate solutions (λ_0) obtained after 200 time-steps. The optimization needs 300 iterations to converge. The convergence history of the optimization iterations are presented in Figure 9.10. Table 9.3 presents a comparison of the baseline and optimized force coefficients. The drag reduction in this case is 17.05% and the lift is well maintained with an increase by only 0.016%. Figure 9.11 presents a comparison of the baseline and optimized pressure distributions (left) and geometries (right) at 4 different span-wise sections. From pressure distributions in the same figure we see that the pressure peak is reduced almost all over the wing. Figure 9.12 presents the initial and final Mach contours on the wing.

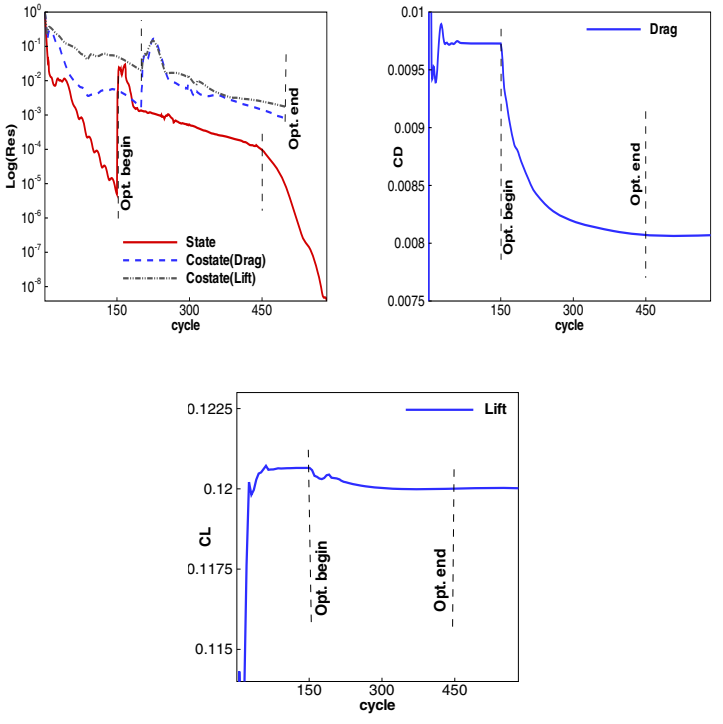


Fig. 9.10 Convergence history of the optimization of Case 4

Optimization of the same geometry in the same flow conditions has been carried out using our 'one-shot' method and using the same FLOWer code as discussed in the previous chapter and in [70]. There the state constraint has been treated 'indirectly' by adding it to the objective function and solving the reduced unconstrained problem. The constant lift is maintained by changing the angle of incidence. There the optimization required 650 iterations to converge. The drag reduction has been 12.59%. In the current computation we could achieve a drag reduction of 17.05% and the total number of optimizations iterations required are 300. This confirms that direct treatment of additional state constraints is advantageous. However, in the current computation we need to solve an additional adjoint equation for the lift constraint. If we add on everything, the total computational effort in this case

Table 9.3 Comparison of force coefficients for baseline and optimized wing

Geometry	Iter	CD	ΔC_D	CL	ΔC_L
Baseline		0.972837E-02		0.120000E+00	
Optimized	300	0.806945E-02	17.05%	0.120019E+00	-0.016%

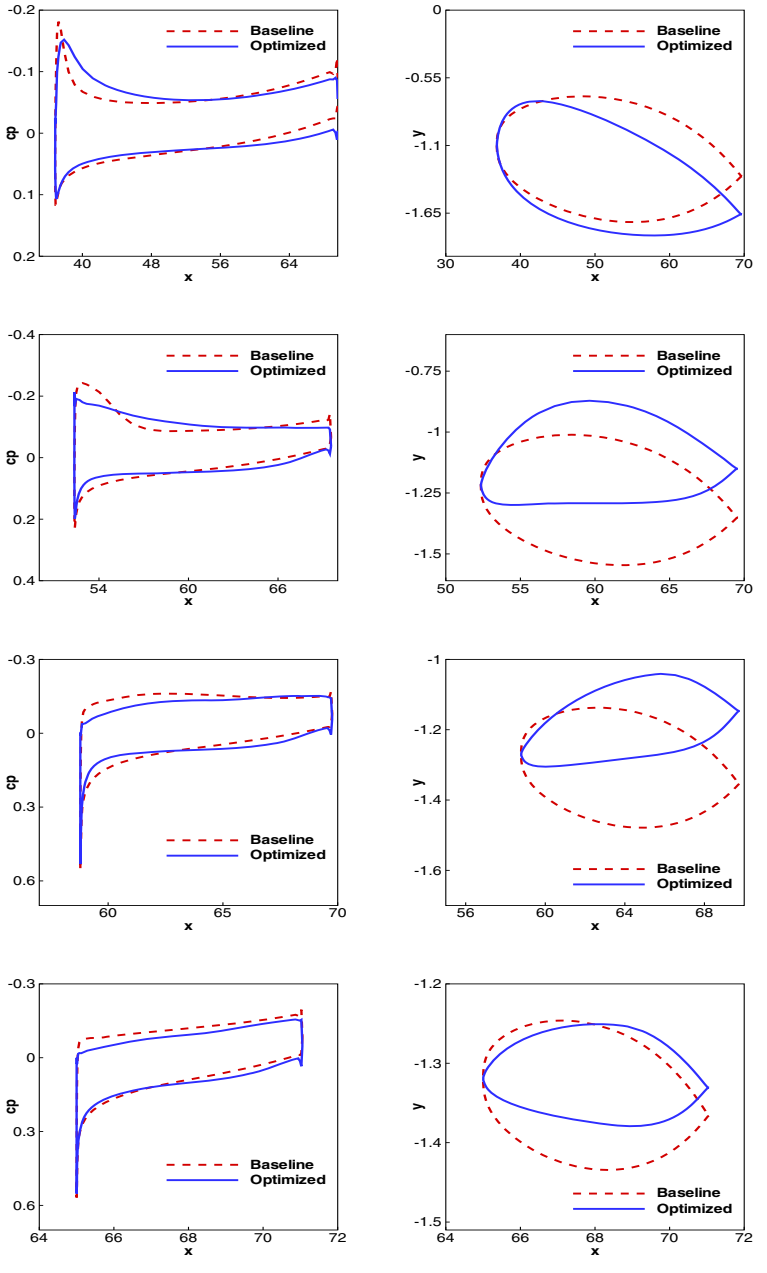


Fig. 9.11 Comparison of surface pressure distributions (left) and geometries (right) at 4 sections (from top to bottom) at $\eta = 0.24, 0.49, 0.70, 0.92$ of the wing

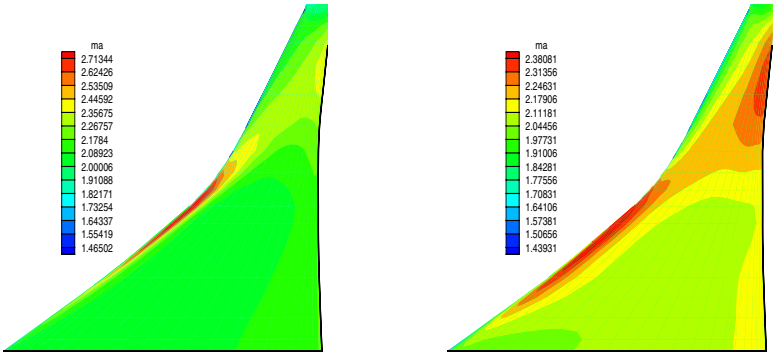


Fig. 9.12 Baseline (left) and optimized (right) Mach contours on the wing

is less than 6 times that of forward simulation runs. In [23] optimization of the same geometry has been carried out using traditional gradient methods together with indirect treatment of the state constraint (as in [70]). The total effort required there consist of 39 forward simulation runs plus 5 adjoint runs and the drag reduction has been 12.58%. In comparison to that, the current effort is a reduction of 86%.

9.4 Conclusions

Problems of aerodynamic shape optimization with additional state constraints have been solved using simultaneous pseudo-time-stepping. The preconditioned pseudo-stationary state, costate and design equations are integrated simultaneously in time until a steady state is reached. In the direct treatment of the state constraints, the solution strategy consists of projection of unconstrained design velocity onto the tangent space of the constraints. Computational examples, at different discretization levels using different design parameter spaces, in 2D as well as in 3D are provided. This kind of 'direct' treatment of constraints leads to efficiency in the solution process and to a better optimum of the optimization problem than that of the 'indirect' treatment presented in the previous chapter. Faster convergence is achieved in finer design parameter space using the current pseudo-time-stepping method. The overall cost of computation is approximately 7-11 times (depending on the grid size and the number of design parameters) that of the forward simulation runs for two additional state constraints in 2D and less than 6 times that of the forward simulation runs for a single additional state constraint in 3D.

Analog to the results presented in Chapters 7 and 8 using one-shot pseudo-time-stepping method, in the convergence histories of all the computations reported in this chapter, a linear convergence with respect to the objective is observed. Starting from an almost feasible initial guess, each iteration step in the primal, adjoint and design

space is so small that the process truly reflects a continuous behavior. Each integration step of the preconditioned pseudo-time formulation is contracting enough so that no additional globalization strategy is necessary. The preconditioning in the design space gives short enough integration steps (corrected by back projection) so that the iterates ”surf” along an almost feasible manifold towards optimality.

Chapter 10
Multigrid One-Shot Pseudo-Time-Stepping Method for Aerodynamic Shape Optimization

10.1 Introduction

In all the applications mentioned in earlier Chapters 6-9, the cost of computation is reduced drastically in comparison to the traditional gradient methods. However, the number of optimization iterations is comparatively large since we update the design parameters after each time-step of the state and costate solver. Therefore, additional computational overhead due to, for example, grid generation, surface parameterization, etc., is high, specially for problems in 3D, since they are to be performed in each optimization iteration. In this paper we use a multigrid strategy to accelerate the optimization convergence.[1]

The 'optimization-based' multigrid method, as proposed and applied to model problems in [157, 128, 117, 118], has been used here. The basic difference is that we use the multigrid in the context of simultaneous pseudo-time-stepping. That means different optimization subproblems of similar structure are solved on different discretization levels using one-shot simultaneous pseudo-time-stepping. The coarse grid solution can be used to find the optimal direction of the fine grid optimization problem efficiently. Also, the problem on a coarse grid is computationally less expensive than that on a fine grid. Since the subproblems on different grid levels are of similar structure, another advantage is the use of the same algorithm and the same software modules to solve them on all grid levels. We have included computational examples of drag reduction, with some geometrical constraints, for an RAE2822 airfoil and for a Supersonic-Cruise-Transport (SCT) wing. The number of optimization iterations are reduced by more than 65% of that of single grid computations and the overall cost of computation of the optimization problem on the fine grid is less than 2 forward simulation runs in 2D and less than 4 forward simulation runs in 3D.

[1] Materials presented in this chapter can also be found in [61], Copyright ©2008 Society for Industrial and Applied Mathematics, reprinted with permission, all rights reserved.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 135–153.
springerlink.com

10.2 The Multigrid Algorithm

The optimization problem that we are dealing with involves PDEs as constraints. Therefore, multigrid methods for PDEs [22, 53, 124, 89] can be used to accelerate the convergence of the PDE solver, thereby accelerating the convergence of the optimization problem. This has been done in [65, 59] to improve the results of [71]. In this chapter, we use an 'optimization-based' multigrid method for the full optimization problem. The basic difference in the current implementation is that we use multigrid in the context of simultaneous pseudo-time-stepping. In the multigrid algorithm we solve different optimization subproblems in different discretization levels. The coarse grid solution, which is achieved with much less computational effort, is used to accelerate the convergence of the optimization problem on the fine grid. The solution methodology is based on simultaneous pseudo-time-stepping as explained in Chapter 6. The ideas are demonstrated through practical aerodynamic shape optimization applications with the Euler equations in 2D as well as in 3D.

We denote by n the current mesh resolution and by N the next coarser mesh resolution. $\hat{I}_N^n$ denotes the prolongation operator and $\hat{I}_n^N$ denotes the restriction operator. The multigrid algorithm reads as

Algorithm 1

i) If on the finest level, solve partially

$$\begin{aligned} &\min I(w_n, q_n) \\ &\text{s. t.} \quad \mathbf{c}(w_n, q_n) = 0, \end{aligned} \tag{10.1}$$

and get $q_n^{(1)}$.
ii) Compute $g^{(1)} = \hat{I}_n^N \nabla I$.
iii) Compute $q_N^{(1)} = \hat{I}_n^N q_n^{(1)}$.
iv) Solve in each coarse grid iteration

$$\begin{aligned} &\min I(w_N, q_N) + {g^{(1)}}^T q_N \\ &\text{s. t.} \quad \mathbf{c}(w_N, q_N) = 0, \end{aligned} \tag{10.2}$$

to get the update vector $e_N = -\Delta t B_k^{-1} g_N$ *(as in step (3) of Algorithm 1 of Chapter 7).*
v) Compute $e_n = \hat{I}_N^n e_N$
vi) Update $q_n^{new} = q_n^{(1)} + e_n$.
vii) Goto step (iii) in case of more than one coarse grid iterations.
Otherwise,
viii) Solve partially

$$\begin{aligned} &\min I(w_n, q_n) \\ &\text{s. t.} \quad \mathbf{c}(w_n, q_n) = 0, \end{aligned} \tag{10.3}$$

with initial solution q_n^{new}.

This defines the V-cycle template of the multigrid algorithm. The objective function of coarse-grid problem differs from that in [117, 118] since we use inexact gradients (simple adjustment of a correction term for inexact gradients, as suggested in [128], can be made which will lead to the current objective function).

Lemma 9. *The gradient of the coarse-grid problem (10.2) at $q_N^{(1)}$ is the projection of the gradient of the fine-grid problem (10.1) at $q_n^{(1)}$ together with an additive correction.*

Proof: Straightforward calculation of gradients of the objective functions will lead to the result. □

This fact assures that the steps based on coarse-grid problem will lead to (faster) improvement for the fine-grid problem. The computations are started on the finest level. Solve 'partially' means a few iterations of the one-shot method are carried out. Linear interpolation is used for prolongation and simple injection is used for restriction. Problems (10.1) and (10.2) are of similar structure. Hence, all steps of Algorithm 1 of Chapter 7 can be carried out at respective discretization levels. Only for problem (10.2), step 3 of the Algorithm is carried out after the prolongation of the update vector to next higher level. Therefore, we can use all the modules of the codes, developed in our earlier works, in different discretization levels with minor modifications.

10.3 Numerical Results and Discussion

The test cases chosen here, as well as in [61], are to minimize the drag of the profile or of the wing with some geometric constraints.

10.3.1 Drag Reduction with Constant Thickness for RAE2822 Airfoil

The optimization method is applied to test cases of the RAE2822 airfoil at Mach number 0.73 and angle of incidence 2 degrees. The physical domain is discretized using an algebraically generated (193×33) C-grid. This is the grid on the finest level (denoted by L1). Next coarser grid (denoted by L2) is of size (97×17) and the coarsest grid (denoted by L3) is of size (49×9). On these grid levels the preconditioned pseudo-stationary equations, resulting from the necessary optimality conditions corresponding to the optimization subproblems of Section 10.2, are solved using Algorithm 1 of Chapter 7. The airfoil is decomposed into thickness and camberline for parameterization purposes. The parameters for thickness are kept unchanged to satisfy the constraint of constant thickness. The camberline is parameterized by 21 Hicks-Henne [78] parameters and this is the number of design parameters in L1. The number of design parameters in L2 is 11 and in L3 is 6. Complete optimization cycle is performed under the optimization platform SynapsPointerPro [44]. We start the optimization iteration (i.e., w_0 and λ_0) with the solution

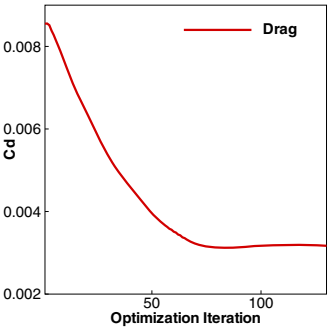


Fig. 10.1 Convergence history of the optimization iterations (single grid)

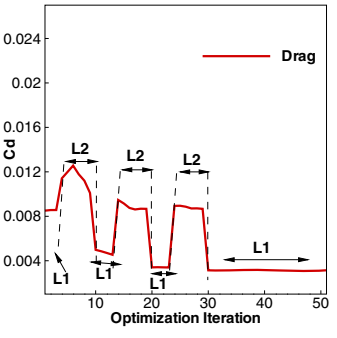


Fig. 10.2 Convergence history of the optimization iterations (Case 1)

obtained after 100 time steps of the state and costate equations on any level. During a switch from 'h' to 'H' or from 'H' to 'h' the 'restart' facility is used to read the solution of last iteration on the same level. Since there is considerable change in geometry when the computations return to a particular level, few (35, in the current implementations) time-steps of state and costate solver are carried out to reduce the numerical error in the computation (see the Figures of state and costate convergence histories). After the convergence of the optimization problem, another 100 time-steps are carried out (in L1) for the state equation to get more accurate values of the surface pressure and force coefficients (which are comparable to the values obtained by other methods). We use the FLOWer code [108, 109, 46], which has been modified and integrated for one-shot methods.

In the current study we have used the reduced Hessian approximation as explained in Chapter 7 (Section 7.5, Case 2) at all grid levels. The values of the constants β, as well as β_{min} and β_{max}, which represent the scaling of the updates of design parameters, are chosen so that the design steps are larger in coarser levels than those used in finest level. Table 10.1 presents the number of iterations required

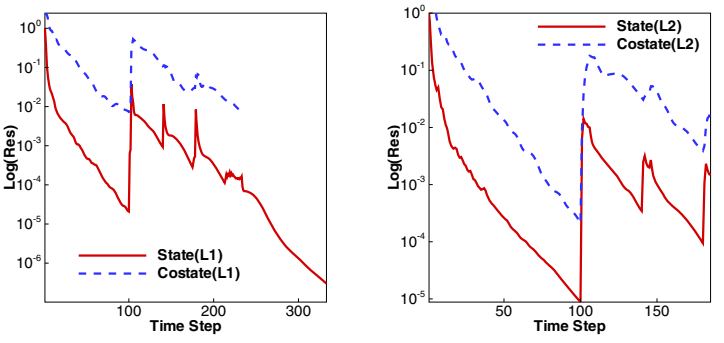


Fig. 10.3 Convergence history of state and costate residuals on level-1 (left) and level-2 (right) (Case 1)

for the convergence of the optimization problem around a local minimizer (where a shock free airfoil results) in all the cases of 2D computations reported below.

Figure 10.1 presents the optimization convergence history of the single grid computation reported in Chapter 7, Figure 7.5. Here the convergence of the objective function values between optimization begin and its end is presented. Convergence history of the state and the costate residuals and a comparison of baseline and optimized airfoils and surface pressure distributions for the single grid computation is also presented in Figure 7.5.

Case 1: Multigrid computation on two grid levels

In this case the computations on two grid levels are carried out using the multigrid strategy as explained in Algorithm 1. We start the computation on the finest level (L1). Each V-cycle consists of 4-iterations on the finest level and 6 iterations on the coarser level (L2). The optimization requires 3 V-cycles to approach convergence. After the last V-cycle another 16 iterations are carried out in the finest level to reach the convergence of the optimization problem. The optimization convergence history is presented in Figure 10.2. Figure 10.3 presents the convergence history of the state and the costate residuals on both levels. One notices in this figure that the drag value on L2 (coarser grid) is higher than that on L1 (finer grid) even though the shock is much weaker on the coarser grid (see Figure 10.19). As is well-known in CFD in 2D, the computed drag value has two components, one due to shock (also known as 'wave drag') and the other due to numerical error (also known as 'spurious' drag). For coarser grids the wave drag is less but the drag due to numerical error leads to higher value. Figure 10.4 presents a comparison of baseline and optimized camberlines, airfoils and surface pressure distributions.

In these applications, the airfoil shapes are smooth (except at the leading and trailing edges) on any of the meshes used. On the other hand, smooth features in the airfoil can give rise to non-smooth features (e.g., shocks) in the flow. This coupling between low-frequency and high-frequency features suggests that it is not clear a priori whether one should necessarily reduce the number of design variables on

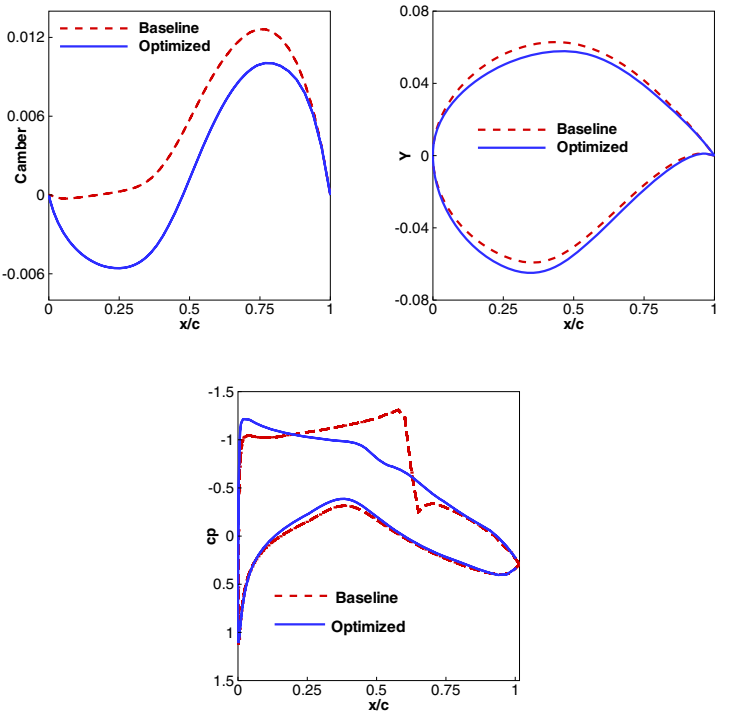


Fig. 10.4 Comparison of Camberlines, airfoils and surface pressure distributions (Case 1)

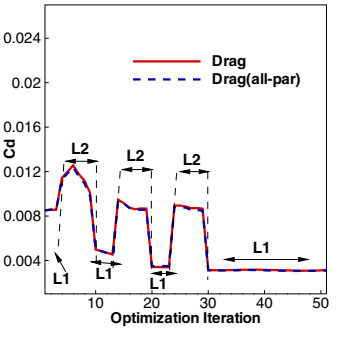


Fig. 10.5 Comparison of convergence history of the optimization iterations (Case 1)

coarser meshes since the airfoil profiles are already smoothed by being represented on coarser meshes. We carry out that test by using the same number of design variables in L1 and L2. All the other parameters and criterion remain same as above. Figure 10.5 presents the comparison of the convergence histories for both cases (one with reduced number of design parameters in L2 as above (marked 'Drag') and the

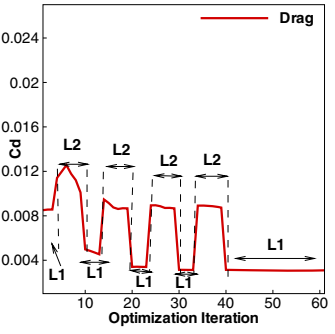


Fig. 10.6 Convergence history of the optimization iterations (Case 1, 4 V-cycles)

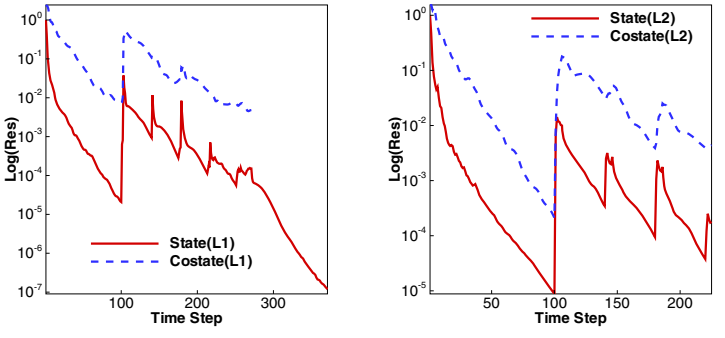


Fig. 10.7 Convergence history of state and costate residuals on level-1 (left) and level-2 (right) (Case 1, 4 V-cycles)

other with same number of design parameters in L2 (marked 'Drag(all-par)')). The force coefficients obtained are also presented in Table 1 (Case 1 (3V, all-par)). The results obtained are quite close. This also proves the fact that since the design variables themselves are not discretized quantities, so one can, in principle, use the same set of design variables at all mesh levels. However, in our computations reported here we used reduced number of design variables on the coarser mesh levels.

Next, we carry out the same computation for one more V-cycle, that is, for 4 V-cycles. After the 4th V-cycle another 16 iterations are carried out in L1 as in the case of earlier computation. The optimization convergence history is presented in Figure 10.6. As we see, the last V-cycle has almost no effect in reducing the drag on fine grid level, since the solution has reached very close to the optimum. This confirms that the multigrid computations are effective during the early optimization iterations, i.e., when far away from the solution. This also shows a kind of numerical stability of the method for computing solutions. Figure 10.7 presents the convergence history of state and costate solutions on both levels. Figure 10.8 presents a comparison of

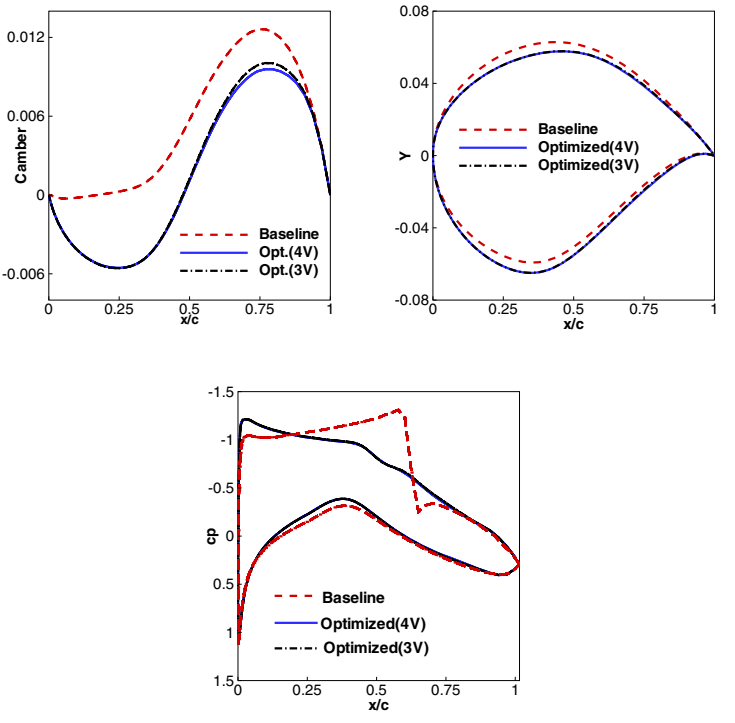


Fig. 10.8 Comparison of Camberlines, airfoils and surface pressure distributions (Case 1)

baseline and optimized camberlines, airfoils and surface pressure distributions of both the computations. There is no significant difference in the final values.

Case 2: Multigrid computation on three grid levels

In this case the multigrid computations are carried out on three grid levels. Initial iterations are started at the finest level (L1). 4 iterations are carried out on this level. Then the computations are carried out, for just 1 iteration, in the next coarser level (L2). Next we pass on to the coarsest level (L3). 3 iterations are carried out in this level. Then 2 iterations are carried out in the prolongation from L3 to L2. Finally, 4 iterations are carried out on the finest L1-level. Two V-cycles and a total of 50 optimization iterations are required for convergence. The convergence histories are presented in Figures 10.9 and 10.10. A comparison of baseline and optimized camberlines, airfoils and surface pressure-distributions are presented in Figure 10.11. In this case, the number of optimization iterations remains same as in Case 1, only one V-cycle is reduced, thereby reducing the total number of state and costate iterations in the finest level. The grid in L3 is too coarse and due to dominating 'spurious' drag (see Figure 10.19) the computational cost is not reduced significantly from that of two level computations. Since the computations on the coarser level are cheaper, one would expect more number of optimization iterations on the coarser level. But,

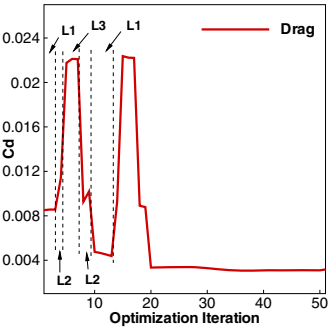


Fig. 10.9 Convergence history of the optimization iterations (Case 2)

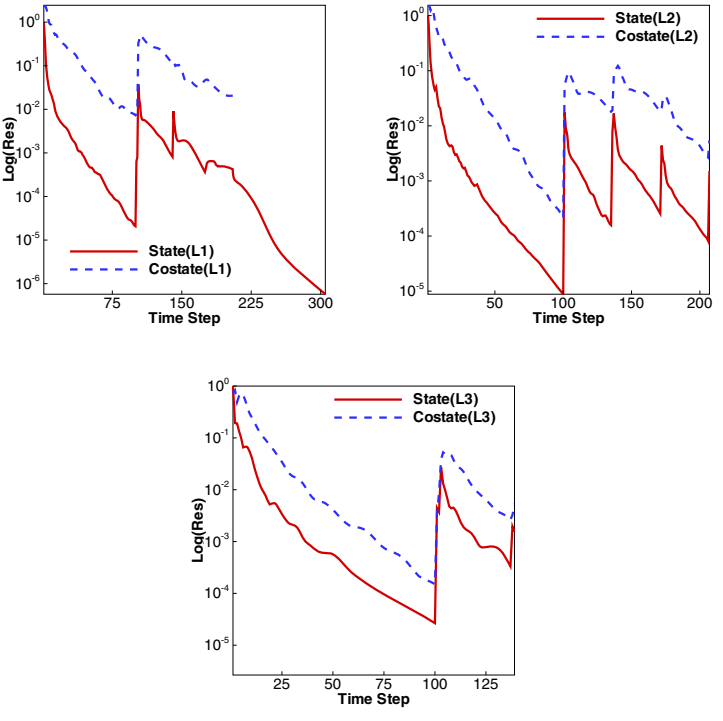


Fig. 10.10 Convergence history of state and costate residuals on level-1 (left), level-2 (middle) and level-3 (right) (Case 2)

as mentioned, due to dominating 'spurious' darg in coarser level, this can not be done. One has to find a balance between the number of iterations and the numerical error present in the solution. The numbers mentioned here, in all the computations, are based on computational experience.

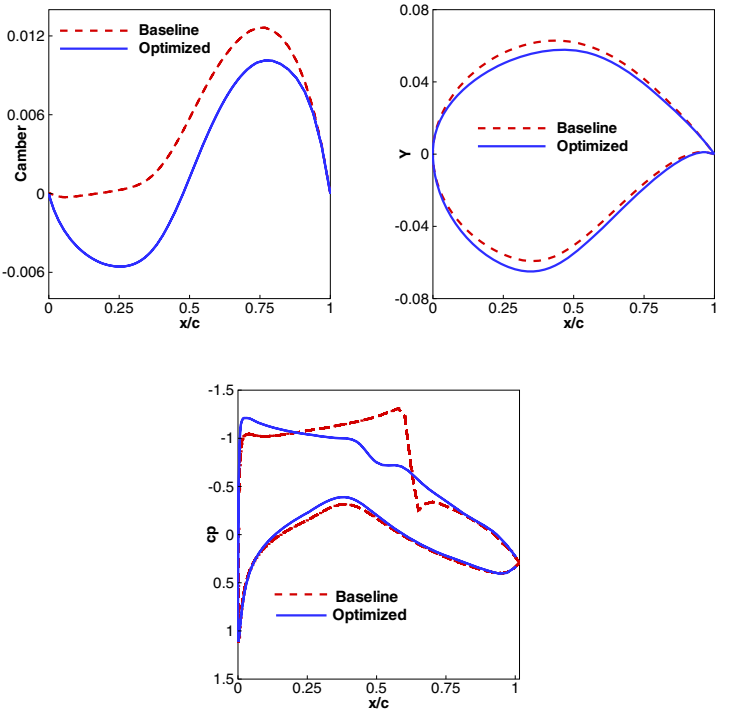


Fig. 10.11 Comparison of Camberlines, airfoils and surface pressure distributions (Case 2)

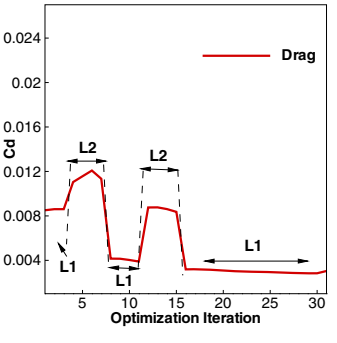


Fig. 10.12 Convergence history of the optimization iterations (Case 3)

Case 3: Multigrid computation on two grid levels with larger parameter-space In this case the design space is parameterized by 41 design parameters. This means the number of design parameters in L1, L2 and L3 are 41, 21 and 11 respectively. The computations are carried out on two-grid levels. Initially 4 iterations are carried out on the finest level(L1). Then 4 iterations are carried out on L2-level. Two

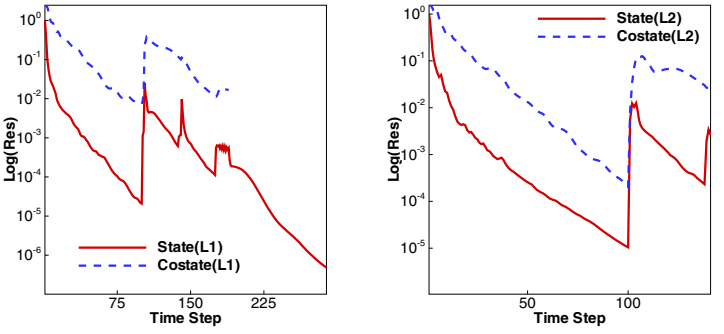


Fig. 10.13 Convergence history of state and costate residuals on level-1 (left) and level-2 (right) (Case 3)

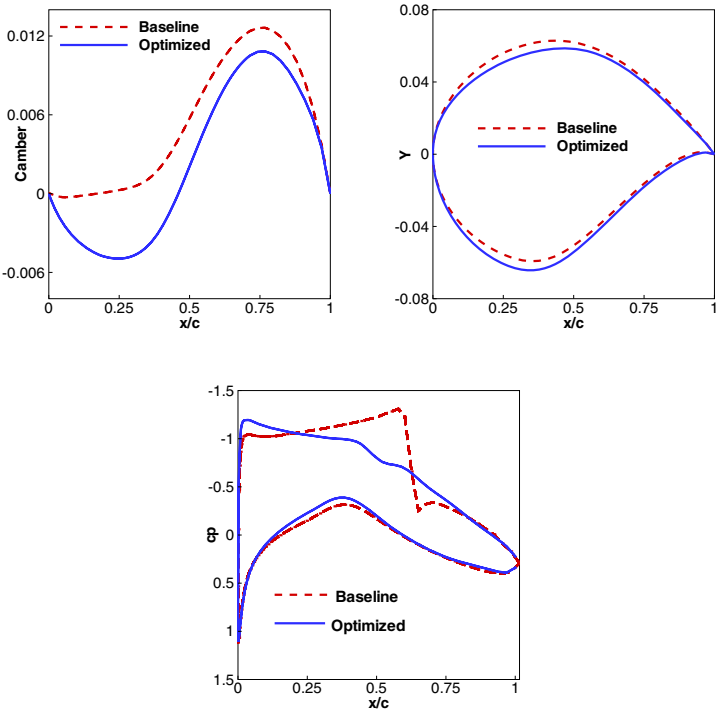


Fig. 10.14 Comparison of Camberlines, airfoils and surface pressure distributions (Case 3)

V-cycles and a total of 30 iterations are required for the convergence of the optimization problem. The convergence histories are presented in Figures 10.12 and 10.13. A comparison of baseline and optimized camberlines, airfoils and surface pressure distributions are presented in Figure 10.14. As we argued in Chapter 9, with finer

Table 10.1 Comparison of number of iterations and force coefficients for baseline and optimized airfoil using different multigrid iterations

Geometry	Iter	CD	CL	CM
Baseline		0.849651E-02	0.826399E+00	0.126806E+00
Single grid	130	0.314641E-02	0.746177E+00	0.105484E+00
Case 1 (3V)	50	0.314252E-02	0.732291E+00	0.103026E+00
Case 1 (3V, all-par)	50	0.310599E-02	0.726345E+00	0.101171E+00
Case 1 (4V)	60	0.309381E-02	0.724621E+00	0.100792E+00
Case 2	50	0.317398E-02	0.733226E+00	0.102741E+00
Case 3	30	0.304247E-02	0.733132E+00	0.102170E+00
Case 4	30	0.307547E-02	0.736523E+00	0.103127E+00

parameter-space the convergence of the optimization is faster in the pseudo-time one-shot method. This is true in the multigrid context as well.

Case 4: Multigrid computation on three grid levels with larger parameter-space

In this case the computations are carried out on three grid levels as in Case 2 with the parameterization as explained in Case 3. On the finest level 4 iterations are carried out. On the next coarser level (L2) 1 iteration as well as on the coarsest level (L3) 1 iteration is carried out. In the prolongation steps from L3 to L2 and from L2 to L1, 1 iteration each is carried out. The convergence of the optimization requires one V-cycle and a total of 30 optimization iterations. The convergence history of the drag is presented in Figure 10.15. The convergence history of the state and the costate iterations are presented in Figure 10.16. A comparison of camberlines, airfoils and surface pressure distributions are presented in Figure 10.17. In this case also, the total number of optimization iterations is the same as in Case 3, but one V-cycle is reduced, thereby reducing the total number of state and costate iterations on the fine

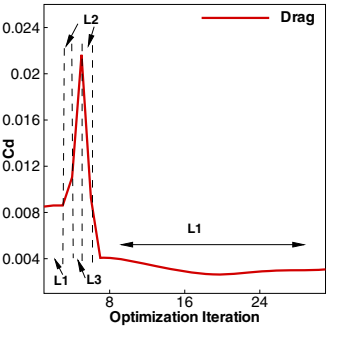


Fig. 10.15 Convergence history of the optimization iterations (Case 4)

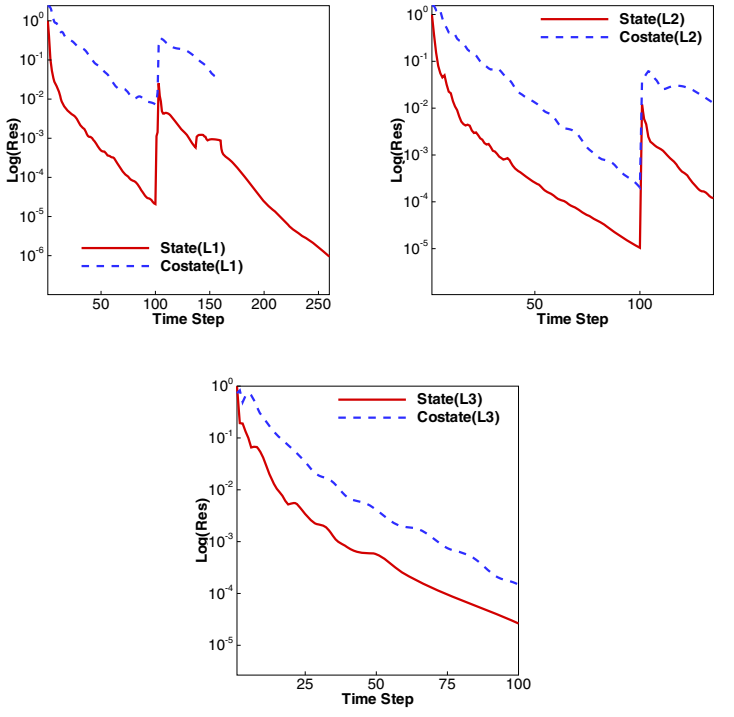


Fig. 10.16 Convergence history of state and costate residuals on level-1 (left), level-2 (middle) and level-3 (right) (Case 4)

grid level. One sufficiently converged forward solution needs about 350 time-steps. As we see, the total effort in the finest level is less than two forward solutions.

Table 10.1 presents a comparison of the number of iterations and the force coefficients of baseline and optimized geometries of all the above cases of multigrid optimization iterations. The optimized force coefficients are almost the same in all the cases. However, the total number of optimization iteration is reduced up to 75% as that of the single grid computation. As we see, the drag reduction is about 63% in all the cases. The lift and pitching moment coefficients are also presented in the same table. Since there is no constraint on these two quantities, they are also reduced by about 11% and 19% respectively.

Figure 10.18 presents pressure and Mach contours of the baseline and optimized (Case 4) geometries. This also confirms the shock-free airfoil as a result of the optimization.

10.3.2 Drag Reduction with Geometric Constraints for SCT Wing

In this case optimization is carried out for drag reduction with geometric constraints for an SCT wing at Mach number 2.0 and angle of incidence 3.22949 degrees. The

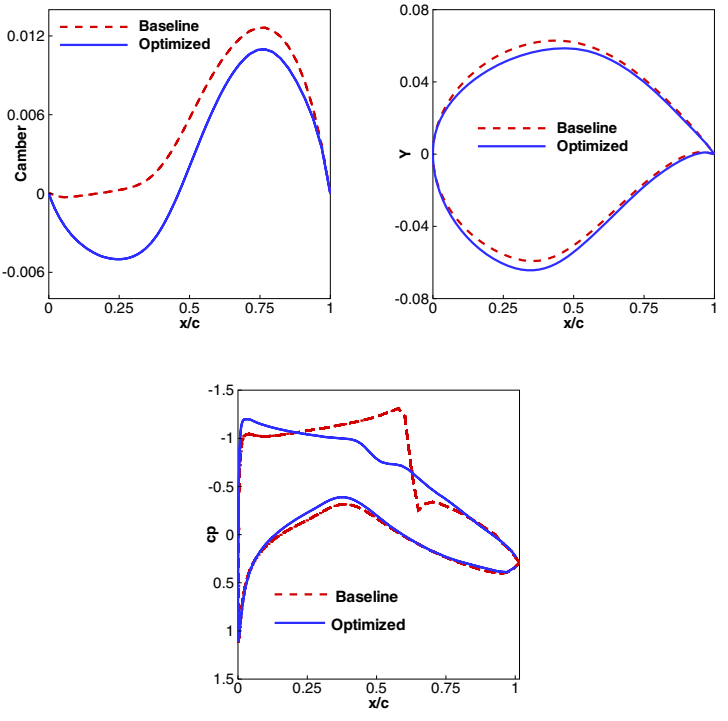


Fig. 10.17 Comparison of Camberlines, airfoils and surface pressure distributions (Case 4)

geometric constraints are taken care via the parameterization of the wing. The physical domain is discretized by a grid of C-H topology consisting of $(97 \times 17 \times 25)$ grid points. Details of parameterization of the wing and single grid computational results are presented in Chapter 7, Section 7.6.2. A total of 500 optimization iterations are carried out for drag reduction. Convergence histories of the drag as well as state and costate solutions are presented in Figure 7.13. Figure 7.14 presents a comparison of initial and final geometries and pressure distributions at 4 different span-wise sections.

Case 6: Multigrid results on two grid levels

In this case the same computations are carried out using the optimization based multigrid methods on two grid levels. The fine-grid discretization and parameterization is described as in Chapter 7, Section 7.6.2. The coarser computational grid consists of $(49 \times 9 \times 13)$ grid points. In the parameterization for coarser level, the thickness-parameterization has been kept unchanged from that described earlier for the fine level, since the geometric constraints are taken care by this. The camberline is modified by 6 Hicks-Henne functions at 8 wing sections and the twist distribution has been described by a Bezier curve defined by 6 nodes. This results in total 86 parameters in the coarser level.

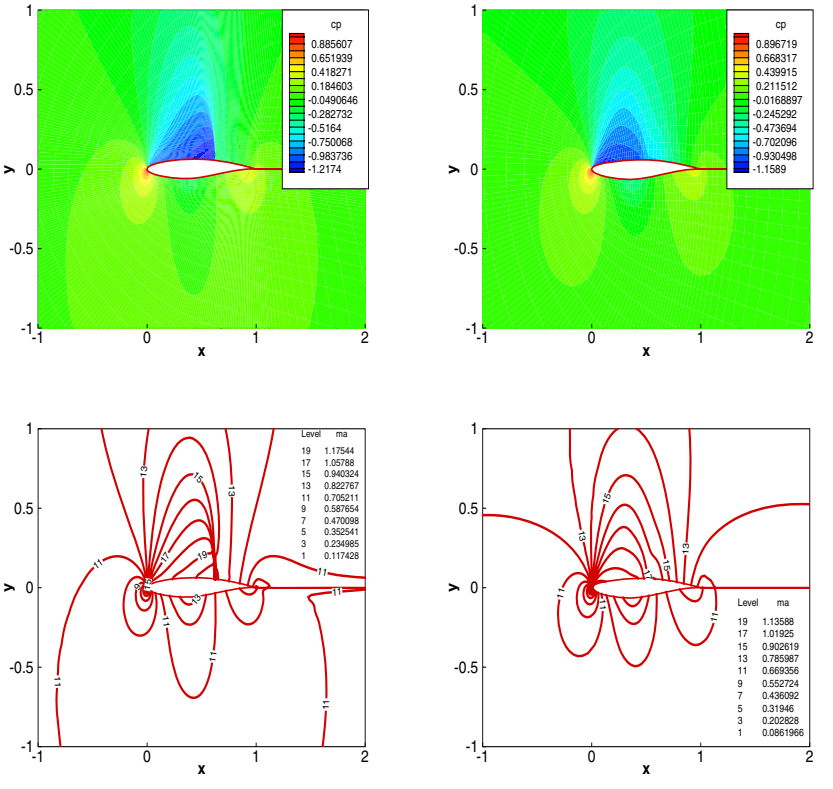


Fig. 10.18 Comparison of baseline (left) and optimized (right) pressure (top) and Mach (bottom) contours (Case 4)

Initial optimization iteration starts at the fine level (L1) where 15 iterations are carried out. In coarser level (L2) 4 iterations are carried out. The optimization requires 4 V-cycles and a total of 180 optimization iterations. The optimization convergence history is presented in Figure 10.20. The convergence history of state and costate residuals are presented in Figure 10.21. Baseline and optimized geometries and surface pressure distributions at 4 wing sections are presented in Figure 10.22. The results are quite similar to those obtained by single grid computation. Similarity can also be seen in the pressure and Mach contours presented in Figure 10.23.

Table 10.2 presents a comparison of number of iterations, baseline and optimized force coefficients obtained using single grid as well as multigrid computations. Using the multigrid strategy a 68% drag reduction could be achieved by 180 iterations, whereas using single grid computation this needs about 500 iterations. One fully converged forward simulation run needs about 400 iterations on the fine grid level. Hence, the total effort required in the fine grid level is less than 4 forward simulation runs. From the lift coefficient (CL) values presented in the same table, we see that

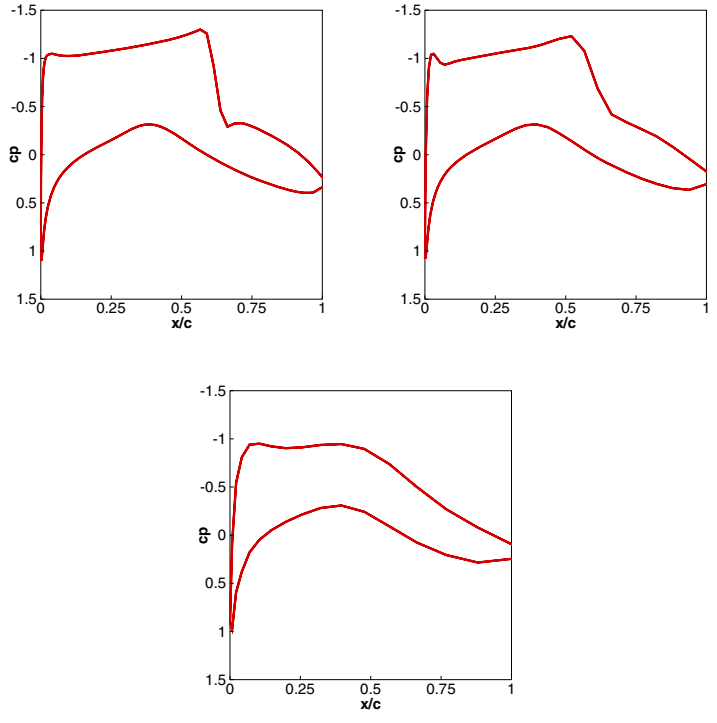


Fig. 10.19 Surface pressure distributions on (193×33) (left), (97×17) (middle) and (49×9) (right) grid

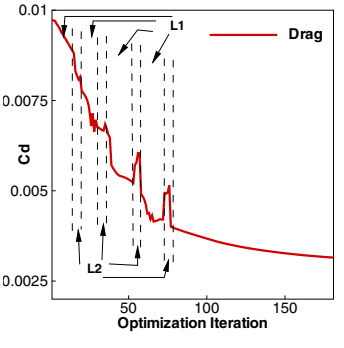


Fig. 10.20 Convergence history of the optimization iterations (Case 6)

the reduction of this value is more than 58%. This is due to the fact that in 3D the computed drag value contains those components mentioned in 2D and additionally a third component, known as induced drag, which is due to lift [82]. Therefore, huge

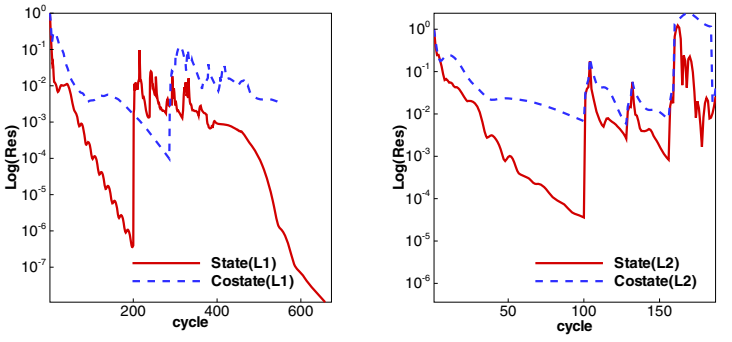


Fig. 10.21 Convergence history of state and costate residuals on level-1 (left) and level-2 (right) (Case 6)

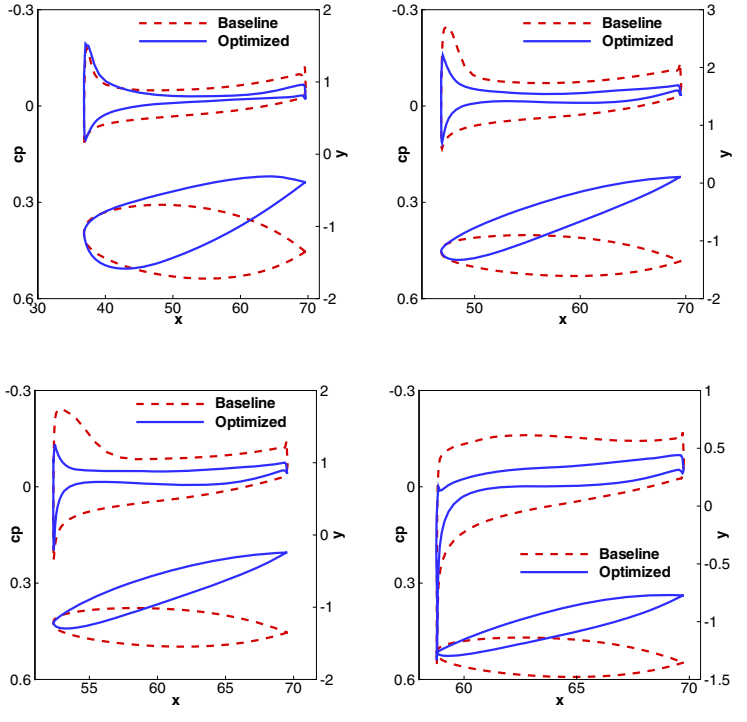


Fig. 10.22 Comparison of initial and optimized wing-sections and pressure distributions at 4 different sections at $\eta = 0.24, 0.39, 0.49, 0.70$ (from top-left to bottom) (Case 6)

drag reduction in this case is at the cost of loss of lift. Hence, practical shape optimization problem (specially, in 3D) should involve additional constraints (e.g., drag reduction with constant lift together with geometrical constraints).

Table 10.2 Comparison of number of iterations and force coefficients for baseline and optimized wing using single grid and multigrid computations

Geometry	Iter	CD	CL	CM
Baseline		0.972837E-02	0.120660E+00	0.350336E-01
Single grid	500	0.293458E-02	0.452202E-01	0.473493E-02
Multigrid	180	0.315004E-02	0.515958E-01	0.705556E-02

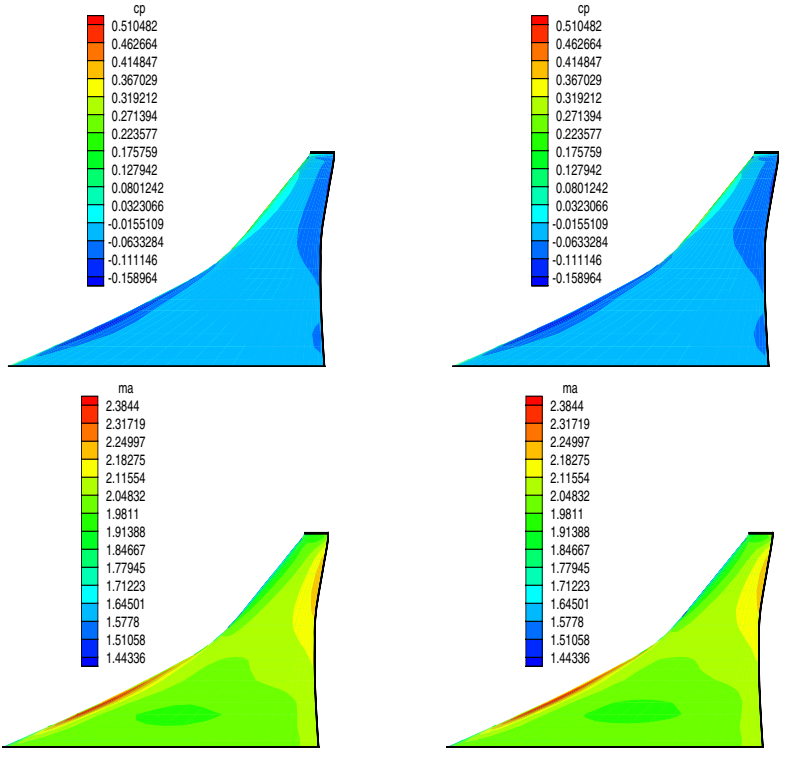


Fig. 10.23 Pressure (top) and Mach (bottom) contours on the wing obtained by single grid (left) and multigrid (right) computations

10.4 Conclusions

An 'Optimization-based' multigrid strategy is used in the context of simultaneous pseudo-time-stepping methods for aerodynamic shape optimization. The preconditioned pseudo-stationary state, costate and design equations are integrated simultaneously in time at different discretization levels. Coarse grid solution, which is less expensive to compute, is used to accelerate the convergence of the optimization problem on the fine grid. Due to similar structure of the optimization subproblems at different levels, the same algorithm and the same software modules can be used,

with minor modifications, to solve those subproblems. The overall convergence is achieved in about 25%-35% of the effort of that required by single grid computations. The overall cost of computation is less than 2 times the forward simulation runs in 2D and is less than 4 times the forward simulation runs in 3D. Application to problems with additional state constraint is discussed in the next chapter.

Chapter 11
Multigrid One-Shot Pseudo-Time-Stepping Method for State Constrained Aerodynamic Shape Optimization

11.1 Introduction

As shown in the previous chapter, the 'optimization based' multigrid method brings efficiency in the convergence of the optimization problem. This helps in considerable reduction of the number of optimization iterations required by the one-shot pseudo-time-stepping method. Hence, it reduces the over-all cost of computation. Since practical aerodynamic shape optimization problems involve additional state constraints, these problems are more challenging, and also more demanding, than those discussed in the previous chapter. Any reduction of computational cost of these problems pose a great challenge, since achieving the optimality faster leads to loss of feasibility of the constraint. We extend the multigrid strategy for these problems in this chapter.[1]

Due to involvement of additional state constraint, the validity region of the solution is restricted. Hence, faster movement towards the optimality causes loss of feasibility. Therefore, one has to find a balance in some way so that faster convergence of the problem is achieved maintaining the feasibility. In our case, we introduce additional corrections to the objective function as well as to the state constraint in the coarse grid sub-problem. We extend the multigrid method of previous chapter to the constrained problem in the context of simultaneous pseudo-time-stepping. That means, here also, different optimization constrained subproblems of similar structure are solved on different discretization levels using one-shot simultaneous pseudo-time-stepping method. The coarse grid solution is used to find the optimal direction of the fine grid optimization problem efficiently. The optimization subproblem in coarse grid differs from that of the fine grid subproblem through the additional corrections in the objective function as well as in the state constraint. We have included computational examples of drag reduction with constant lift, together with geometrical constraint of constant thickness, for RAE2822 airfoil. The number of optimization iterations are reduced by more than 65% of that of single grid

[1] Materials presented in this chapter can also be found in [62], Copyright ©2008 Society for Industrial and Applied Mathematics, reprinted with permission, all rights reserved.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 155–174.
springerlink.com

computations and the overall cost of computation of the optimization problem on fine grid is about 3-5 forward simulation runs.

11.2 The Multigrid Algorithm

In the multigrid algorithm we solve different optimization subproblems on different discretization levels. The coarse grid solution, which is achieved in much less computational effort, is used to accelerate the convergence of the optimization problem in fine grid. The solution methodology is based on simultaneous pseudo-time-stepping as discussed in Chapter 9. The ideas are demonstrated through practical aerodynamic shape optimization applications in 2D.

We denote by n the current mesh resolution and by N the next coarser mesh resolution. $\hat{I}_N^n$ denotes the prolongation operator and $\hat{I}_n^N$ denotes the restriction operator. The multigrid algorithm reads as

Algorithm 1

i) If on the finest level, solve partially

$$\begin{aligned} &\min I(w_n, q_n) \\ \text{s. t.} \quad &\mathbf{c}(w_n, q_n) = 0, \\ &h(w_n, q_n) = 0. \end{aligned} \tag{11.1}$$

and get $q_n^{(1)}$.

ii) Compute $g^{(1)} = \hat{I}_n^N g$ *and* $g_h^{(1)} = \hat{I}_n^N g_h$.

iii) Compute $q_N^{(1)} = \hat{I}_n^N q_n^{(1)}$.

iv) Solve in each coarse grid iteration

$$\begin{aligned} &\min I(w_N, q_N) + {g^{(1)}}^T q_N \\ \text{s. t.} \quad &\mathbf{c}(w_N, q_N) = 0, \\ &h(w_N, q_N) + {g_h^{(1)}}^T q_N = 0. \end{aligned} \tag{11.2}$$

to get the update vector $e_N = -\Delta t B_k^{-1}(g_N + g_{hN}\mu_N) - \Delta q_N$ *(as in steps (5) and (6) of Algorithm 1 of Chapter 9).*

v) Compute $e_n = \hat{I}_N^n e_N$

vi) Update $q_n^{new} = q_n^{(1)} + e_n$.

vii) Goto step (iii) in case of more than one coarse grid iterations. Otherwise,

viii) Solve partially

$$\begin{aligned} &\min I(w_n, q_n) \\ \text{s. t.} \quad &\mathbf{c}(w_n, q_n) = 0, \\ &h(w_n, q_n) = 0, \end{aligned} \tag{11.3}$$

with initial solution q_n^{new}.

This defines the V-cycle template of the multigrid algorithm. In simultaneous pseudo-time-stepping we use in-exact reduced gradients in each optimization update, therefore, these are appropriate correction terms for the objective function and for the state constraint in the coarse grid subproblem.

Lemma 10. *The reduced gradients of the coarse-grid problem (11.2) at* $q_N^{(1)}$ *is the projection of the reduced gradients of the fine-grid problem (11.1) at* $q_n^{(1)}$ *together with an additive corrections.*

Proof: Straightforward calculation of reduced gradients of the objective function and of the state constraint will lead to the result. □

This fact assures that the steps based on coarse-grid problem will lead to (faster) improvement of the optimal solution of the fine-grid problem maintaining the feasibility. The computations are started in the finest level. Again, solve 'partially' means a few iterations of one-shot method are carried out. Linear interpolation is used for prolongation and simple injection is used for restriction. Problems (11.1) and (11.2) are of similar structure. Hence, all steps of Algorithm 1 of Chapter 9 can be carried out at respective discretization levels. Only for problem (11.2), step 6 of the Algorithm is carried out after the prolongation of the update vector to finest level. Therefore, we can use all the modules of the codes, developed in our earlier works, in different discretization levels with minor modifications.

In the present chapter, as well as in [62], we have implemented this method for the shape design example using Euler equations. The details of governing equations, discretization, geometry parameterization, gradient computation and grid perturbation strategy can be found in [71, 68].

11.3 Numerical Results and Discussions

The optimization method is applied to test cases of the RAE 2822 airfoil for drag reduction with constant lift together with geometric constraint of constant thickness. The flow conditions are given at transonic Mach number 0.73 and angle of incidence 2 degrees. The airfoil is decomposed into thickness and camberline for parameterization purposes. The parameters for thickness kept unchanged to satisfy the constraint of constant thickness. The camberline is parameterized by Hicks-Henne functions. We use the FLOWer code which has been modified and integrated for the one-shot method discussed in Chapter 9.

In the current study we have used the reduced Hessian approximation as explained in Section 7.5 (Case 2) at all grid levels. Here also, the values of the constants β, as well as β_{min} and β_{max}, which represent the scaling of the updates of design parameters, are chosen so that the design steps are larger in coarser levels than those used in finest level.

11.3.1 Drag Reduction with Constant Lift on (193×33) Grid

The physical domain is discretized using an algebraically generated (193×33) C-grid. This is the grid on the finest level (denoted by L1). Next coarser level grid (denoted by L2) is of size (97×17). In the multigrid method on these grid levels, the preconditioned pseudo-stationary equations, resulting from the necessary optimality conditions corresponding to the optimization subproblems of Section 11.2, are solved using Algorithm 1 of Chapter 9. We start the optimization iteration (i.e., w_0 and λ_0) with the solution obtained after 100 time steps of the state and costate equations on L1 level and 50 time steps of the state and costate equations on L2 level. During a switch from 'n' to 'N' or from 'N' to 'n' the 'restart' facility is used to read the solution of last iteration on the same level. Since there is considerable change in geometry when the computations return to a particular level, few (35, in the current implementations) time-steps of state and costate solver are carried out to reduce the numerical error in the computation (see in the Figures of state and costate convergence histories of case 2 and case 4). After the convergence of the optimization problem, another 100 time-steps are carried out (in L1) for the state equation to get more accurate values of the surface pressure and force coefficients (which are comparable to the values obtained by other methods).

Case 1: Single grid computation with 41 design parameters
In this case we carry out computations on single grid level. Number of design parameters is 41. The optimization problem requires 225 iterations to converge. Figure 11.1 presents the optimization convergence history of this case. Figure 11.2 presents a comparison of the baseline and the optimized airfoils and surface pressure distributions of this computation.

Case 2: Multigrid computation on two grid levels with 41 design parameters
In this case the computation is carried out on two grid levels using the multigrid strategy as explained in Algorithm 1 in this chapter. Number of design parameters on L1 is 41 and that on L2 is 21. The computation starts on the finest level (L1). Each V-cycle consists of 4-iterations on the finest level and 6 iterations on the coarser level (L2). The optimization requires 3 V-cycles to reach very close to the convergence. After the last V-cycle another 50 iterations are carried out in the finest level to reach the convergence of the optimization problem. The total number of optimization iterations required in this case is 80. The optimization convergence history is presented in Figure 11.4. Figure 11.5 presents the convergence history of the state and the costate residuals on both grid levels. Figure 11.6 presents a comparison of the baseline and the optimized airfoils and surface pressure distributions. Figure 11.7 presents a comparison of camberlines, airfoils and surface pressure distributions obtained in both the cases. As one sees, there is no noticeable difference in these quantities resulting in single grid and in multigrid computations. The resulting force coefficients are presented in Table 11.1 where one notices very close optimized values.

As we see in Figure 11.4, the drag value (0.35637E-02) after the third V-cycle is very close to the final optimum value (0.34935E-02). This confirms that the

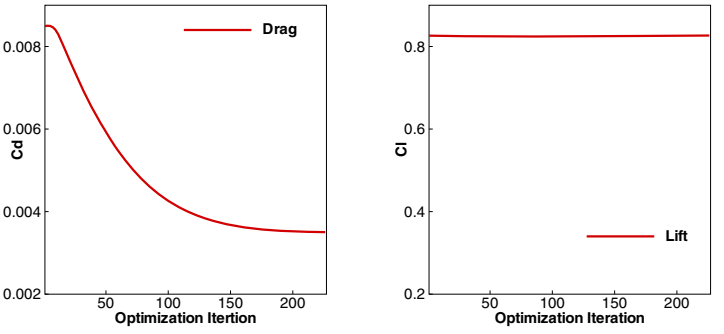


Fig. 11.1 Convergence history of the optimization iterations (Case 1)

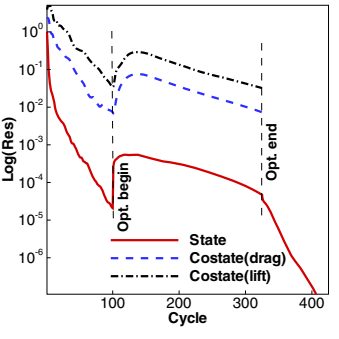


Fig. 11.2 Convergence history of state and costate residuals (Case 1)

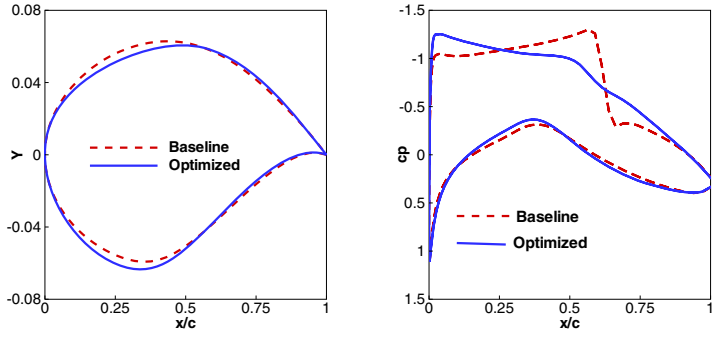


Fig. 11.3 Comparison of airfoils and surface pressure distributions (Case 1)

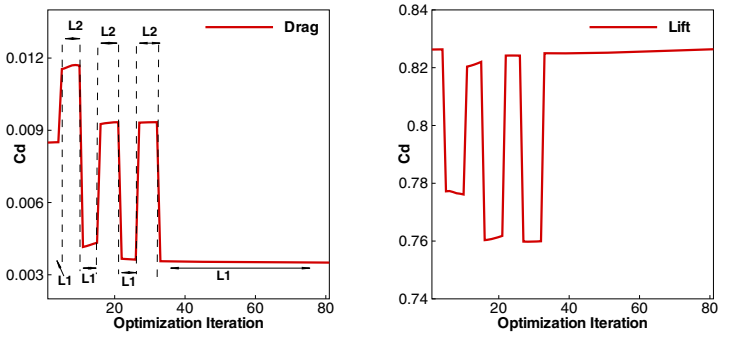


Fig. 11.4 Convergence history of the optimization iterations (Case 2)

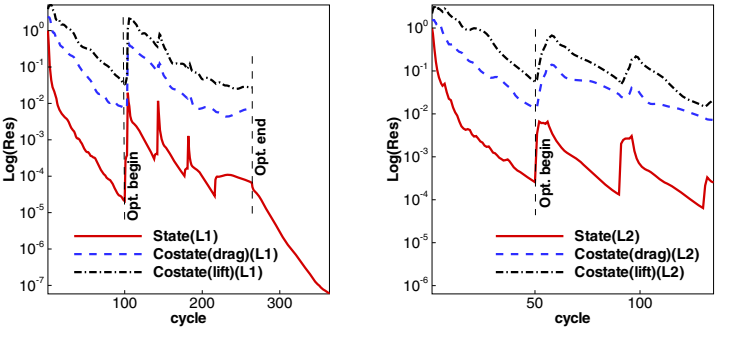


Fig. 11.5 Convergence history of state and costate residuals on level-1 (left) and level-2 (right) (Case 2)

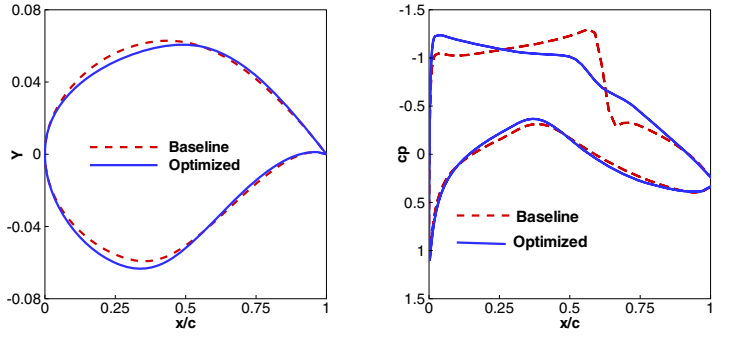


Fig. 11.6 Comparison of airfoils and surface pressure distributions (Case 2)

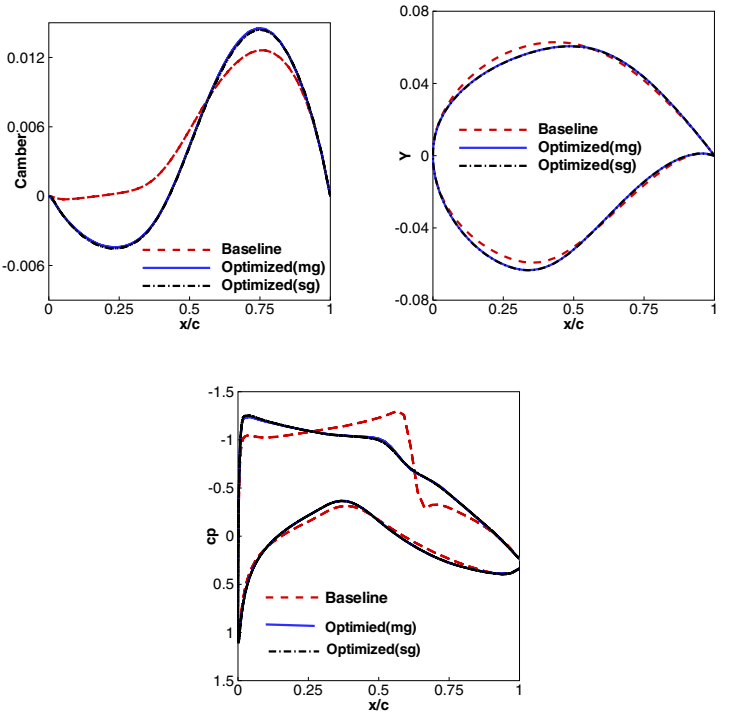


Fig. 11.7 Comparison of Camberlines, airfoils and surface pressure distributions of Case 1 and Case 2

multigrid computations are effective during the early optimization iterations, i.e., when far away from the solution. This is the case in all the multigrid computations reported here.

Case 3: Single grid computation with 81 design parameters
In this case we carry out the same computation reported in Case 1 with finer design space with 81 design parameters. In this case the optimization requires 155 iterations to converge. Figure 11.8 presents the optimization convergence history of this case. Figure 11.9 presents the convergence history of state and costate residuals and Figure 11.10 presents a comparison of the baseline and the optimized airfoils and surface pressure distributions of this computation.

Case 4: Multigrid computation on two grid levels with 81 design parameters
In this case the multigrid computations are carried out with finer design space. Initial iterations are started at the finest level (L1). 4 iterations are carried out in this level. Next the computations are carried out, for 3 iterations, in the coarser level (L2). Two V-cycles and total 45 optimization iterations are required for the optimization convergence. The convergence histories are presented in Figures 11.11 and

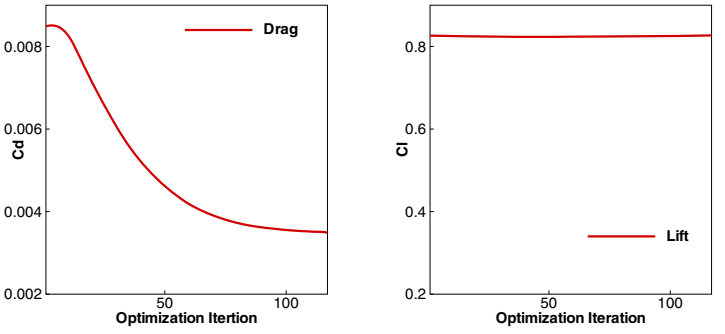


Fig. 11.8 Convergence history of the optimization iterations (Case 3)

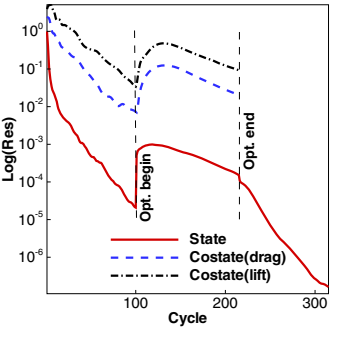


Fig. 11.9 Convergence history of state and costate residuals (Case 3)

11.12. A comparison of the baseline and the optimized airfoils and surface pressure distributions are presented in Figure 11.13. Figure 11.14 presents a comparison of camberlines, airfoils and surface pressure distributions obtained in case 3 (single grid) and in case 4 (multigrid) computations. As we see here as well, there is no noticeable difference in these quantities. The resulting force coefficients are presented in Table 11.1 where one notices very close optimized values. The drag reduction is about 59% and the lift is well maintained in all the cases. On this grid, 'well' converged state solution requires about 350 time-steps. The number of time-steps required on fine grid in this case is (200+200+300=) 700 which about 2 times that of forward simulation runs. The time steps required on coarse grid is less than 300 time steps which is much less than one forward simulation run. Hence, all together the total effort required for the optimization convergence is less than 3 times forward simulation runs. The number of optimization iterations is 45. In traditional gradient methods the number of optimization iterations required are also about the same. However, there the total computational effort is about 50-60 forward simulation runs, since each optimization iteration requires 'well' converged solution.

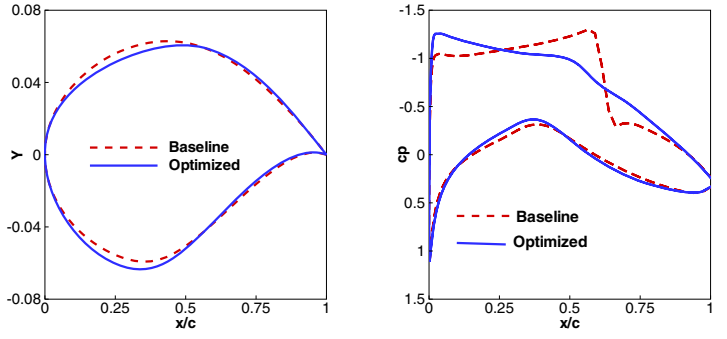


Fig. 11.10 Comparison of airfoils and surface pressure distributions (Case 3)

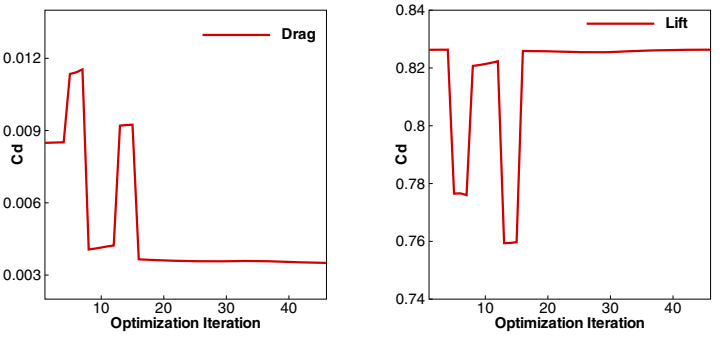


Fig. 11.11 Convergence history of the optimization iterations (Case 4)

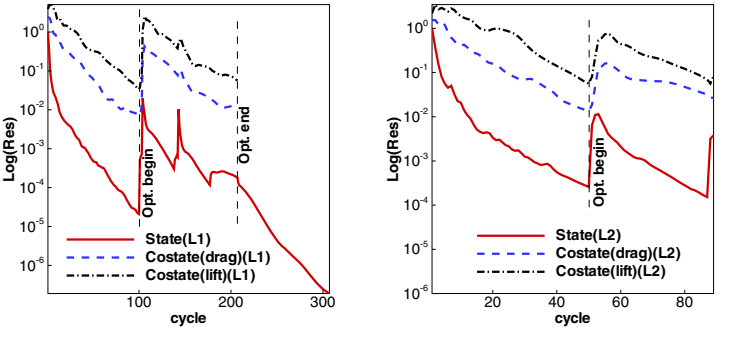


Fig. 11.12 Convergence history of state and costate residuals on level-1 (left), level-2 (middle) and level-3 (right) (Case 4)

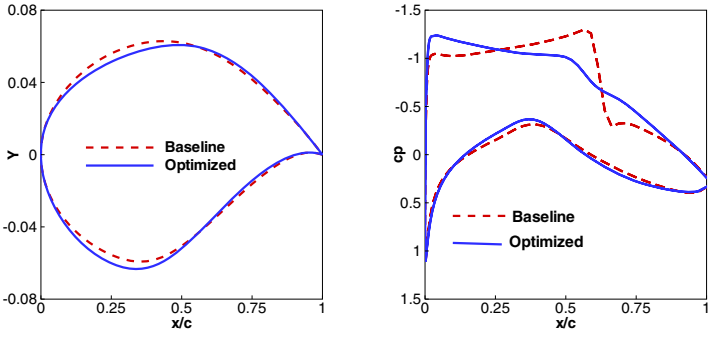


Fig. 11.13 Comparison of airfoils and surface pressure distributions (Case 4)

Table 11.1 Comparison of number of iterations and force coefficients for baseline and optimized airfoil using different multigrid iterations

Geometry	Iter	CD	ΔC_D	CL	ΔC_L
Baseline		0.848960E-02		0.826800E+00	
Opt.(Case 1, $q = 41$, sg)	225	0.349350E-02	58.85%	0.826833E+00	-0.004%
Opt.(Case 2, $q = 41$, mg)	80	0.348720E-02	58.92%	0.826782E+00	0.002%
Opt.(Case 3, $q = 81$, sg)	115	0.348986E-02	58.89%	0.826843E+00	-0.005%
Opt.(Case 4, $q = 81$, mg)	45	0.347574E-02	59.06%	0.826802E+00	-0.000%

11.3.2 Drag Reduction with Constant Lift on (321×57) *Grid*

In the next cases we carry out computations on finer computational grid of size (321×57). This is the grid on the finest level (denoted by L1). Next coarser level grid (denoted by L2) is of size (161×29) and the coarsest level grid (denoted by L3) is of size (81×15). On these grid levels the preconditioned pseudo-stationary equations, resulting from the necessary optimality conditions corresponding to the optimization subproblems of Section 11.2, are solved using Algorithm 1 of Chapter 9. We start the optimization iteration (i.e., w_0 and λ_0) with the solution obtained after 150 time steps of the state and costate equations on L1 level and 100 time steps of the state and costate equations on L2 and L3 levels. During a switch from 'n' to 'N' or from 'N' to 'n' the 'restart' facility is used to read the solution of last iteration on the same level. Since there is considerable change in geometry when the computations return to a particular level, few (35, in the current implementations) time-steps of state and costate solver are carried out to reduce the numerical error in the computation (see in the Figures of state and costate convergence histories). After the convergence of the optimization problem, another 200 time-steps are carried out (in L1) for the state equation to get more accurate values of the surface pressure and force coefficients (which are comparable to the values obtained by other methods).

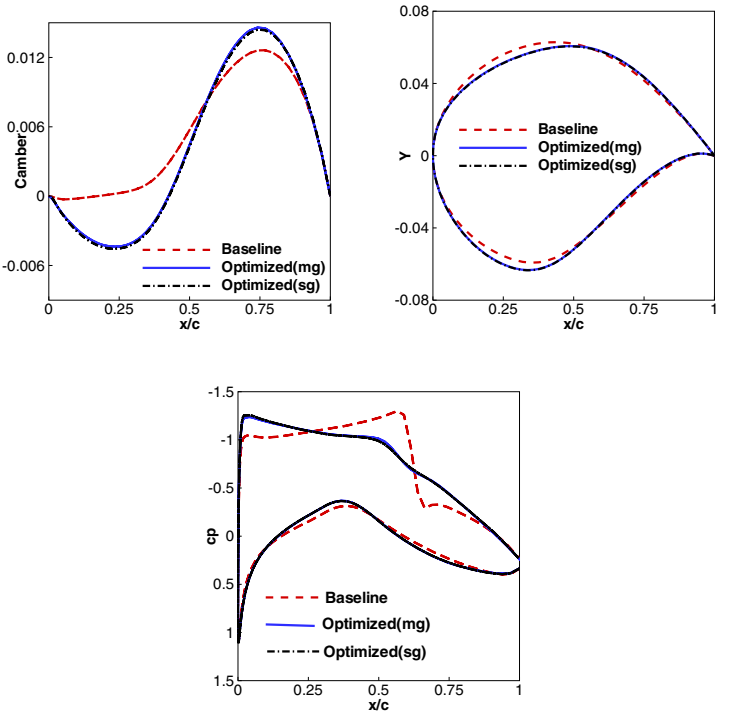


Fig. 11.14 Comparison of Camberlines, airfoils and surface pressure distributions of Case 3 and Case 4

Case 5: Single grid computation with 41 design parameters
In this case the design space is parameterized by 41 design parameters. The computations are carried out on finest grid level only. In this case the optimization requires 725 iterations to converge. Optimization convergence history is presented in Figure 11.15. Convergence history of the state and the costate residuals are presented in Figure 11.16. A comparison of the baseline and the optimized airfoils and surface pressure distributions are presented in Figure 11.17.

Case 6: Multigrid computation on two grid levels with 41 design parameters
In this case we carry out computations on two grid levels using multigrid algorithm. Number of design parameters on L1 level is 41 and that in L2 level is 21. We start the computation on the finest level. Each V-cycle consists of 5 iterations on the finest level and 24 iterations on the coarser level. Since the coarser grid involves less spurious drag on this grid level, more number of iterations are possible on coarser level in this case. 5 V-cycles and in total 325 iterations are required for the convergence of the optimization problem. The optimization convergence histories are presented in Figures 11.18 and 11.19. A comparison of the baseline and the optimized airfoils and surface pressure distributions are presented in Figure 11.20. Figure 11.21

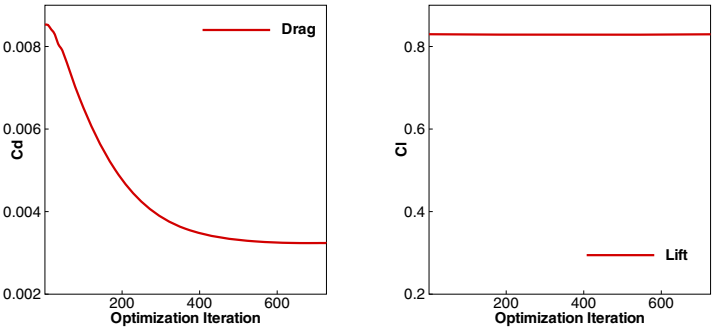


Fig. 11.15 Convergence history of the optimization iterations (Case 5)

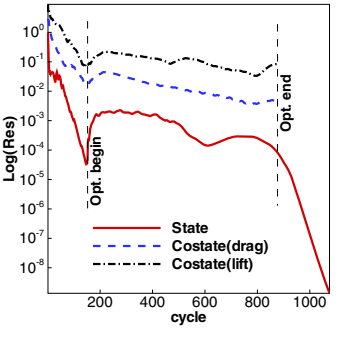


Fig. 11.16 Convergence history of state and costate residuals (Case 5)

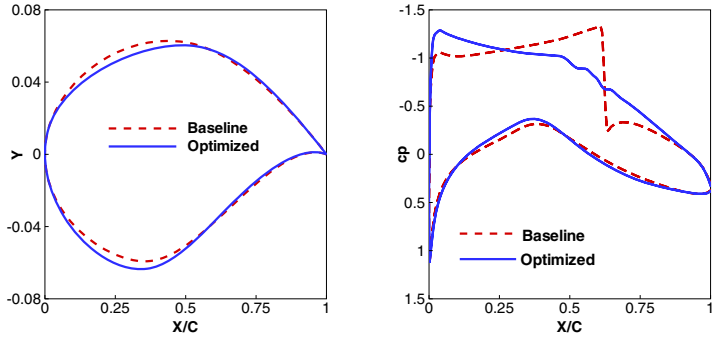


Fig. 11.17 Comparison of airfoils and surface pressure distributions (Case 5)

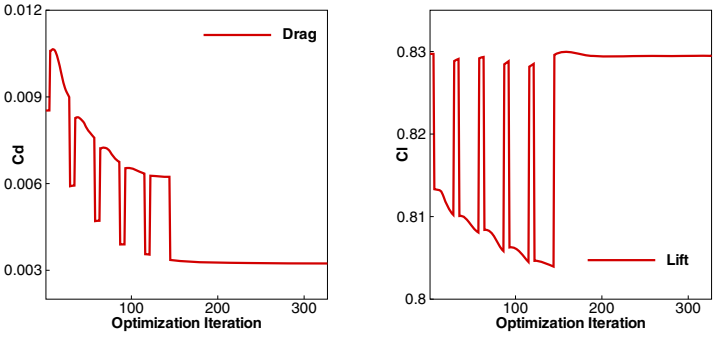


Fig. 11.18 Convergence history of the optimization iterations (Case 6)

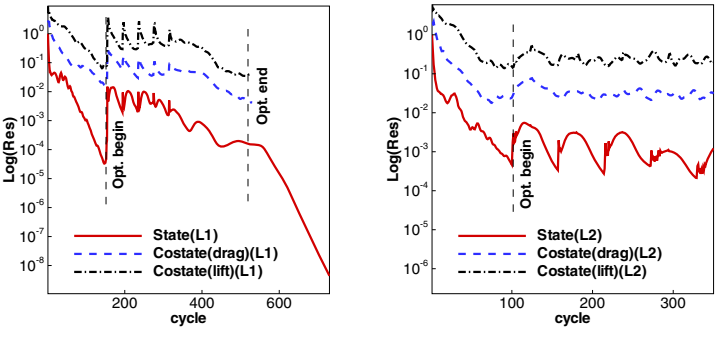


Fig. 11.19 Convergence history of state and costate residuals on level-1 (left) and level-2 (right) (Case 6)

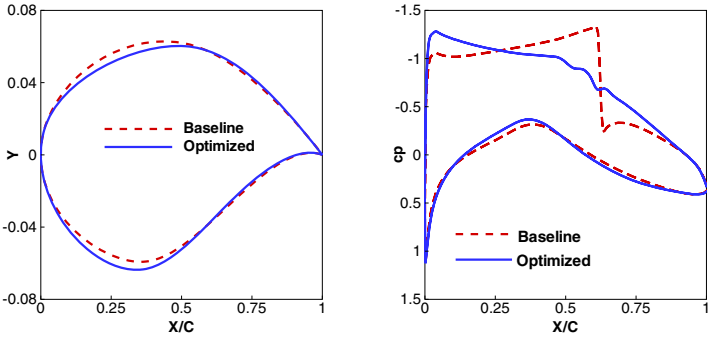


Fig. 11.20 Comparison of airfoils and surface pressure distributions (Case 6)

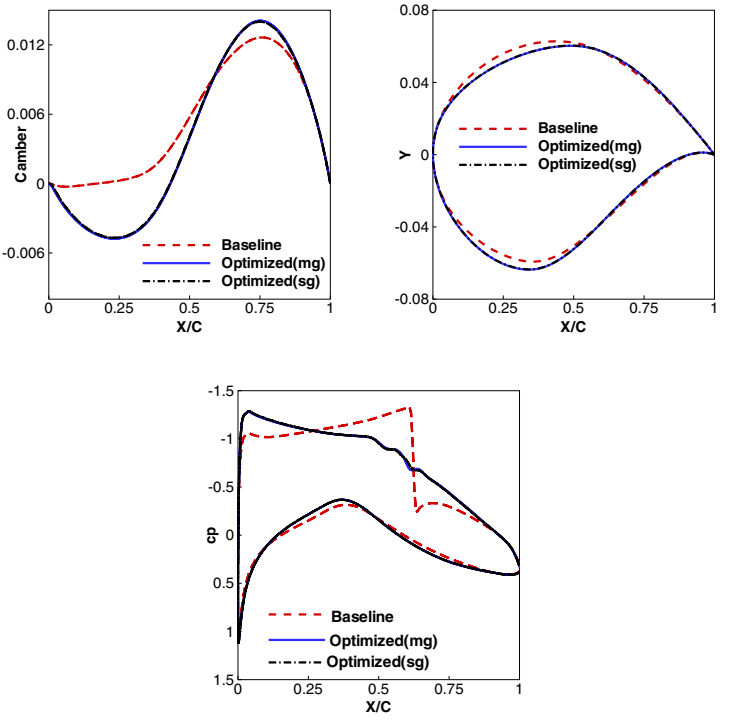


Fig. 11.21 Comparison of Camberlines, airfoils and surface pressure distributions of Case 5 and Case 6

presents a comparison of camberlines, airfoils and surface pressure distributions obtained in case 5 and in case 6 computations. The optimized solutions are almost the same.

Case 7: Single grid computation with 81 design parameters
In this case the design space is parameterized by 81 design parameters. The computations are carried out on finest grid level only. In this case the optimization requires 475 iterations to converge. Optimization convergence history is presented in Figure 11.22. Convergence history of the state and the costate residuals are presented in Figure 11.23. A comparison of the baseline and the optimized airfoils and surface pressure distributions are presented in Figure 11.24.

Case 8: Multigrid computation on two grid levels with 81 design parameters
In this case we carry out computations on two grid levels using multigrid algorithm. Number of design parameters on L1 level is 81 and that in L2 level is 41. We start the computation on the finest level. Each V-cycle consists of 5 iterations on the finest level and 11 iterations on the coarser level. 5 V-cycles and in total 155 iterations are required for the convergence of the optimization problem. The convergence histories

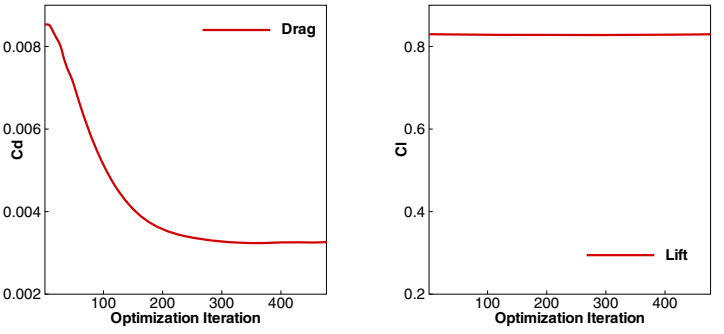


Fig. 11.22 Convergence history of the optimization iterations (Case 7)

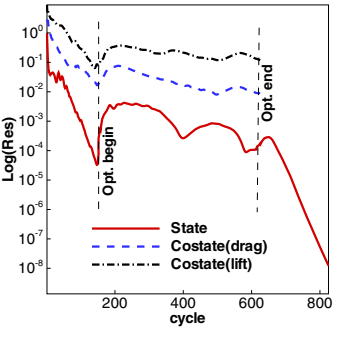


Fig. 11.23 Convergence history of state and costate residuals (Case 7)

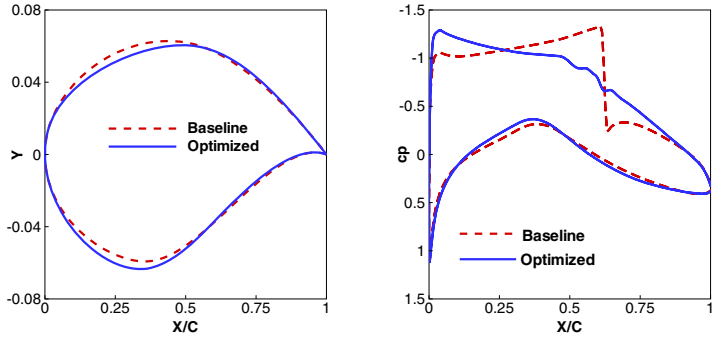


Fig. 11.24 Comparison of Camberlines, airfoils and surface pressure distributions (Case 7)

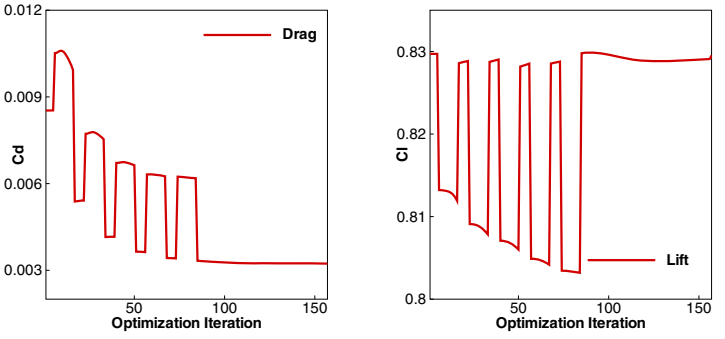


Fig. 11.25 Convergence history of the optimization iterations (Case 8)

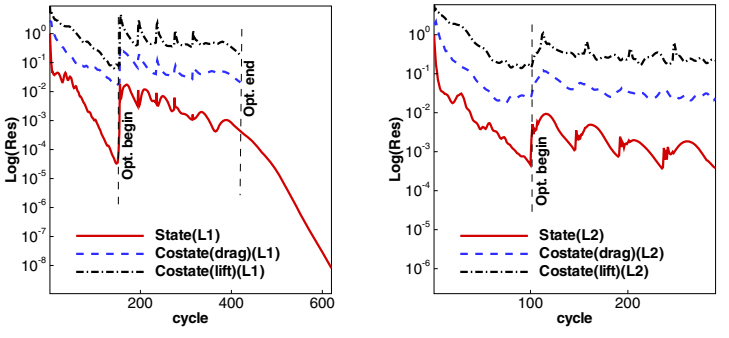


Fig. 11.26 Convergence history of state and costate residuals on level-1 (left) and level-2 (right) (Case 8)

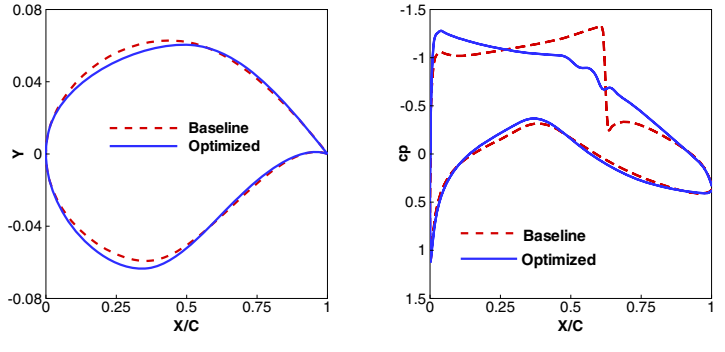


Fig. 11.27 Comparison of Camberlines, airfoils and surface pressure distributions (Case 8)

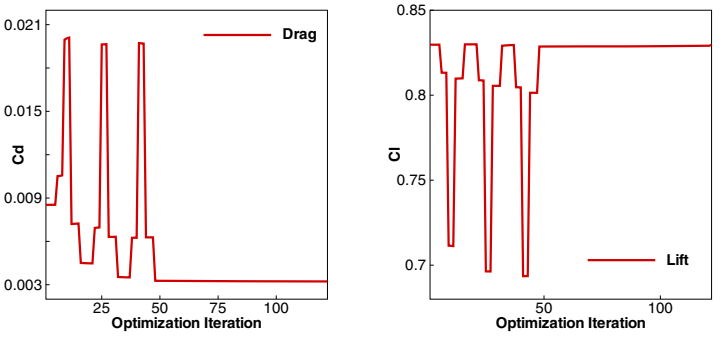


Fig. 11.28 Convergence history of the optimization iterations (Case 9)

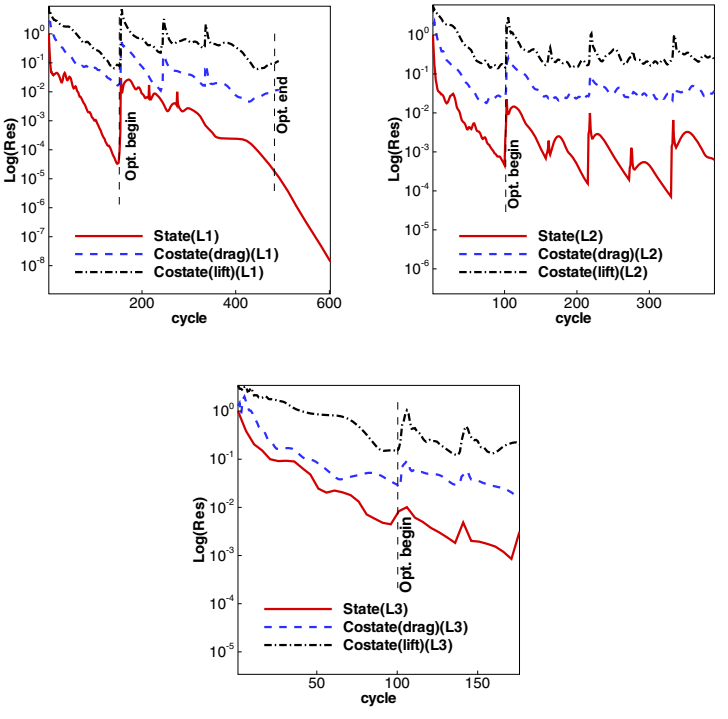


Fig. 11.29 Convergence history of state and costate residuals on level-1 (left), level-2 (middle) and level-3 (right) (Case 9)

are presented in Figures 11.25 and 11.26. A comparison of the baseline and the optimized airfoils and surface pressure distributions are presented in Figure 11.27.

Case 9: Multigrid computation on three grid levels with 81 design parameters
In this case the computations are carried out on three grid levels. The number of

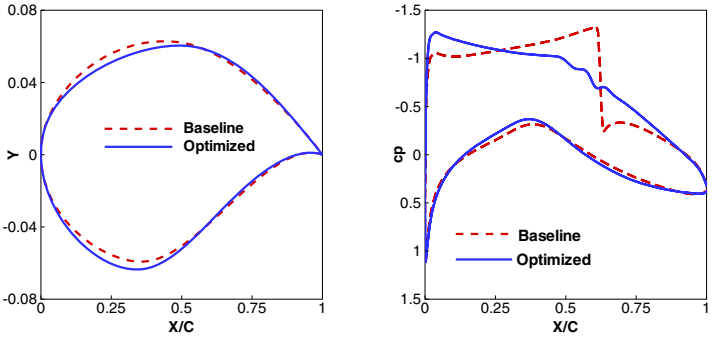


Fig. 11.30 Comparison of Camberlines, airfoils and surface pressure distributions (Case 9)

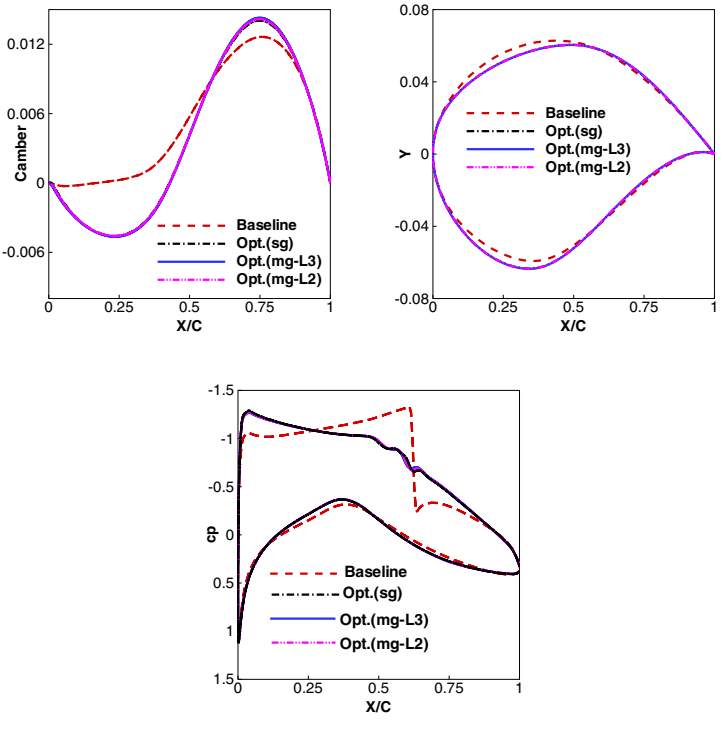


Fig. 11.31 Comparison of Camberlines, airfoils and surface pressure distributions of Case 7, Case 8 and Case 9

design parameters on finest level is 81, on next coarser level (L2) is 41 and on the coarsest level (L3) is 21. In this case during the 'restart' at L1 85 iterations and at L2 55 iterations are carried out for state and costate solver since there is larger change

Table 11.2 Comparison of number of iterations and force coefficients for baseline and optimized airfoil using different multigrid iterations

Geometry	Iter	CD	ΔC_D	CL	ΔC_L
Baseline		0.852910E-02		0.829500E+00	
Opt.(Case 5, $q = 41$, sg)	725	0.323478E-02	62.07%	0.829535E+00	-0.004%
Opt.(Case 6, $q = 41$, mg)	325	0.323486E-02	62.07%	0.829541E+00	-0.005%
Opt.(Case 7, $q = 81$, sg)	475	0.323639E-02	62.05%	0.829522E+00	-0.003%
Opt.(Case 8, $q = 81$, mg)	155	0.322975E-02	62.13%	0.829522E+00	-0.003%
Opt.(Case 9, $q = 81$, mg)	120	0.322674E-02	62.16%	0.829535E+00	-0.004%

in geometry when the computations return to these levels. In step (ii) of Algorithm 1 in this chapter, the reduced gradients are the sum of reduced gradients at levels L1 and L2. In the finest level 5 iterations are carried out. On the next coarser level (L2) 3 iterations as well as on the coarsest level (L3) 3 iterations are carried out. In the prolongation step from L3 to L2 4 iterations are carried out. The convergence of the optimization requires 3 V-cycles and in total 120 optimization iterations. The optimization convergence history is presented in Figure 11.28. The convergence history of the state and the costate iterations are presented in Figure 11.29. A comparison of airfoils and surface pressure distributions are presented in Figure 11.30. Figure 11.31 presents a comparison of camberlines, airfoils and the surface pressure distributions obtained in case 7, in case 8 and in case 9 computations. There is quite good agreement in all the optimized values. Table 11.2 presents a comparison of number of iterations and the force coefficients of the baseline and the optimized geometries of all the above cases. Here also we see very close values of the force coefficients in all the cases. The drag reduction is about 62% and the lift constraint is well maintained in all the cases.

One sufficiently converged forward solution needs about 450 time-steps. As we see, the total effort in the finest level is less than four forward simulation runs. Adding up the time-steps required in coarser and coarsest level, the total effort will be about 5 times that of forward simulation runs.

11.4 Conclusions

'Optimization-based' multigrid strategy is used in the context of Simultaneous pseudo-time-stepping methods for constrained aerodynamic shape optimization problems. Preconditioned pseudo-stationary state, costate and design equations are integrated simultaneously in time at different discretization levels. Coarse grid solution, which is less expensive to compute, is used to accelerate the convergence of the optimization problem in the fine grid. Due to similar structure of the optimization subproblems at different levels, same algorithm and the same software modules could be used, with minor modifications, to solve those subproblems. Multigrid strategy is quite effective when far away from the solution. The number of optimization iterations required in the one-shot method is very close to that required

by the 'black-box' gradient methods. The overall convergence is achieved by about 25%-35% effort of that required by single grid computations using the same method. The overall cost of computation in the current implementation is about 3-5 forward simulation runs, whereas the overall cost of computation is about 50-60 forward simulation runs in 'black-box' gradient methods.

Chapter 12
One-Shot Pseudo-Time-Stepping Method for Aerodynamic Shape Optimization Using the Navier-Stokes Equations

12.1 Introduction

In all the applications of pseudo-time-stepping method mentioned in Chapters 7-11, the state equations have been the Euler equations. In this chapter we extend the method to viscous compressible flow modeled by the Reynolds Averaged Navier-Stokes equations together with algebraic turbulence model of Baldwin and Lomax. While inviscid formulations are useful for the design in transonic cruise conditions, inclusion of viscous effects are essential for optimal design encompassing off-design conditions and high-lift configurations. The computational complexity in viscous design is at least an order of magnitude greater than that in inviscid design since the number of mesh points are to be increased by a factor of two or more to resolve the boundary layer. The convergence of the Navier-Stokes solver is much more slower than the Euler solver due to discrete stiffness and directional decoupling arising from the highly stretched boundary layer cells. Since we use inaccurate state and costate solutions, and hence inaccurate gradients, in our one-shot pseudo-time-stepping method, therefore slow convergence of the Navier-Stokes (forward and adjoint) solver may affect the convergence of this method. We investigate that numerically in this chapter.[1]

As far as the use of inexact state and costate solutions, and hence inexact gradients, are concerned, our one-shot approach has some similarity with the method of A. Jameson. He has proposed gradient smoothing. Instead of using the gradient information from the adjoint solution, the gradient is smoothed implicitly via second order (or fourth order) differential equations and the smoothed gradient is used to find the search direction. It turns out that this approach is tolerant to use inexact gradient so that neither flow solution nor the adjoint solution need to be fully converged. Extension of the continuous adjoint method for shape optimization problems in viscous compressible flow is carried out in [98, 96, 94].

The optimization problem can be written in abstract form (6.3), as in Chapter 6. Here, $\mathbf{c}(w,q) = 0$ represents the steady-state flow equations (in this case Reynolds averaged Navier-Stokes equations) together with boundary conditions.

[1] Materilas presented in this chapter can also be found in [66], reprinted with kind permission of American Institute of Aeronautics and Astronautics, Inc.

S.B. Hazra: Large-Scale PDE-Constrained Optimization in Applications, LNACM 49, pp. 175–188.
springerlink.com

The objective $I(w,q)$ is again the drag of an airfoil for the purposes of this chapter. The preconditioned pseudo-time embedded system that we consider is given by equation (7.1). The reduced Hessian update is based on most recent reduced gradient and most recent parameter update informations as discussed in Chapter 7 (case 2) and can be found in [66].

12.2 Detailed Equations of the Aerodynamic Shape Optimization Problem

In this section we explain briefly the state, costate and design equations represented in equations (8.7) for the shape optimization problem. The formulation of the adjoint and gradient equations for viscous optimization follows the development in references [98] and [96].

State equations: Since we are interested in the steady flow, a proper approach for numerical modeling is to integrate the unsteady Navier-Stokes equations in time until a steady state is reached. These equations in Cartesian coordinates (x,y) for two-dimensional flow can be written in integral form for the region $\mathscr{D}$ with boundaries $\mathscr{B}$ as

$$\frac{\partial w}{\partial t} + \frac{\partial f_i}{\partial x_i} = \frac{\partial f_{vi}}{\partial x_i} \text{ in } \mathscr{D}, \tag{12.1}$$

where

$$w := \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{bmatrix}, \qquad f_i := \begin{bmatrix} \rho u_i \\ \rho u_i u_1 + p\delta_{i1} \\ \rho u_i u_2 + p\delta_{i2} \\ \rho u_i H \end{bmatrix} \quad \text{and} \quad f_{vi} := \begin{bmatrix} 0 \\ \sigma_{ij}\delta_{j1} \\ \sigma_{ij}\delta_{j2} \\ u_j\sigma_{ij} + k\frac{\partial T}{\partial x_i} \end{bmatrix}.$$

For a perfect gas the pressure and total enthalpy is given by

$$p = (\gamma - 1)\rho \left\{ E - \frac{1}{2}(u^2 + v^2) \right\}, \quad H = E + \frac{p}{\rho},$$

respectively. The viscous stresses may be written as

$$\sigma_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \lambda \delta_{ij} \frac{\partial u_k}{\partial x_k},$$

where μ and λ are the first and second coefficients of viscosity. The coefficient of thermal conductivity and the temperature are computed as

$$k = \frac{c_p \mu}{Pr} \quad T = \frac{p}{R\rho},$$

where Pr is the Prandtl number, c_p is the specific heat at constant pressure and R is the universal gas constant.

For discussion of real applications using a discretization on a body conforming structured mesh, it is useful to consider a transformation to the computational coordinates (ξ_1, ξ_2) defined by the metrics

$$K_{ij} = \left[\frac{\partial x_i}{\partial \xi_j}\right], \; J = \det(K), \; K_{ij}^{-1} = \left[\frac{\partial \xi_i}{\partial x_j}\right].$$

The Navier-Stokes equations can then be written in computational space as

$$\frac{\partial (Jw)}{\partial t} + \frac{\partial (F_i - F_{vi})}{\partial \xi_i} = 0 \text{ in } \mathscr{D}, \tag{12.2}$$

where the inviscid and viscous flux contributions are now defined with respect to the computational cell faces by $F_i = S_{ij} f_j$ and $F_{vi} = S_{ij} f_{vj}$, and the quantity $S_{ij} = JK_{ij}^{-1}$ represents the projection of the ξ_i cell face along the x_j axis. In obtaining equations (12.2) we have made use of the property that

$$\frac{\partial S_{ij}}{\partial \xi_i} = 0$$

which represents the fact that the sum of the face areas over a closed volume is zero, as can be readily verified by a direct examination of the metric terms.

The boundary conditions used to solve these equations are the 'no slip' condition on the solid wall, and the farfield boundary is treated by considering the incoming and outgoing characteristics based on the one dimensional Riemann invariants.

Costate equations: Aerodynamic optimization is based on the determination of the effect of shape modifications on some performance measure which depends on the flow. For convenience, the coordinates ξ_i describing the fixed computational domain are chosen so that each boundary conforms to a constant value of one of these coordinates. Variations in the shape then result in corresponding variations in the mapping derivatives defined by K_{ij}.

Suppose that the performance is measured by a cost function

$$I = \int_{\mathscr{B}} \mathscr{M}(w,S)\, dB_\xi + \int_{\mathscr{D}} \mathscr{P}(w,S)\, dD_\xi, \tag{12.3}$$

containing both boundary and field contributions where dB_ξ and dD_ξ are the surface and volume elements in the computational domain. In general, $\mathscr{M}$ and $\mathscr{P}$ will depend on both the flow variables w and the metrics S defining the computational space. The design problem is now treated as a control problem where the boundary shape represents the control function, which is chosen to minimize I subject to the constraints defined by the flow equations (12.2). A shape change produces a variation in the flow solution δw and the metrics δS which in turn produce a variation in the cost function

$$\delta I = \int_{\mathscr{B}} \delta.\mathscr{M}(w,S)\, dB_\xi + \int_{\mathscr{D}} \delta \mathscr{P}(w,S)\, dD_\xi. \tag{12.4}$$

This can be split as

$$\delta I = \delta I_I + \delta I_{II}, \tag{12.5}$$

with

$$\begin{aligned} \delta \mathscr{M} &= [\mathscr{M}_w]_I \, \delta w + \delta \mathscr{M}_{II}, \\ \delta \mathscr{P} &= [\mathscr{P}_w]_I \, \delta w + \delta \mathscr{P}_{II}, \end{aligned} \tag{12.6}$$

where we continue to use the subscripts I and II to distinguish between the contributions associated with the variation of the flow solution δw and those associated with the metric variations δS. Thus $[\mathscr{M}_w]_I$ and $[\mathscr{P}_w]_I$ represent $\frac{\partial \mathscr{M}}{\partial w}$ and $\frac{\partial \mathscr{P}}{\partial w}$ with the metrics fixed, while $\delta \mathscr{M}_{II}$ and $\delta \mathscr{P}_{II}$ represent the contribution of the metric variations δS to $\delta \mathscr{M}$ and $\delta \mathscr{P}$.

In the steady state, the constraint equation (53) specifies the variation of the state vector δw by

$$\delta R = \frac{\partial}{\partial \xi_i} \delta \left(F_i - F_{vi} \right) = 0. \tag{12.7}$$

Here, also, δR, δF_i and δF_{vi} can be split into contributions associated with δw and δS using the notation

$$\begin{aligned} \delta R &= \delta R_I + \delta R_{II} \\ \delta F_i &= [F_{iw}]_I \, \delta w + \delta F_{iII} \\ \delta F_{vi} &= [F_{viw}]_I \, \delta w + \delta F_{viII}. \end{aligned} \tag{12.8}$$

The inviscid contributions are easily evaluated as

$$[F_{iw}]_I = S_{ij} \frac{\partial f_i}{\partial w}, \quad \delta F_{viII} = \delta S_{ij} f_j.$$

The details of the viscous contributions are complicated by the additional level of derivatives in the stress and heat flux terms.

Multiplying by a co-state vector ψ, which will play an analogous role to the Lagrange multiplier, and integrating over the domain produces

$$\int_{\mathscr{D}} \psi^T \frac{\partial}{\partial \xi_i} \delta \left(F_i - F_{vi} \right) d\mathscr{D}_\xi = 0. \tag{12.9}$$

Assuming that ψ is differentiable the terms with subscript I may be integrated by parts to give

$$\int_{\mathscr{B}} n_i \psi^T \delta \left(F_i - F_{vi} \right)_I d\mathscr{B}_\xi - \int_{\mathscr{D}} \frac{\partial \psi^T}{\partial \xi_i} \delta \left(F_i - F_{vi} \right)_I d\mathscr{D}_\xi + \int_{\mathscr{D}} \psi^T \delta R_{II} d\mathscr{D}_\xi = 0. \tag{12.10}$$

This equation results directly from taking the variation of the weak form of the flow equations, where ψ is taken to be an arbitrary differentiable test function. Since the

left hand expression equals zero, it may be subtracted from the variation in the cost function (12.4) to give

$$\delta I = \delta I_{II} - \int_{\mathscr{D}} \psi^T \delta R_{II} d\mathscr{D}_\xi - \int_{\mathscr{B}} \left[\delta \mathscr{M}_I - n_i \psi^T \delta \left(F_i - F_{vi}\right)_I\right] d\mathscr{B}_\xi + \int_{\mathscr{D}} \left[\delta \mathscr{P}_I + \frac{\partial \psi^T}{\partial \xi_i} \delta \left(F_i - F_{vi}\right)_I\right] d\mathscr{D}_\xi. \tag{12.11}$$

Now, since ψ is an arbitrary differentiable function, it may be chosen in such a way that δI no longer depends explicitly on the variation of the state vector δw. The gradient of the cost function can then be evaluated directly from the metric variations without having to recompute the variation δw resulting from the perturbation of each design variable.

Comparing equations (12.6) and (12.8), the variation δw may be eliminated from (12.11) by equating all field terms with subscript "I" to produce a differential adjoint system governing ψ

$$\frac{\partial \psi^T}{\partial \xi_i} \left[F_{i_w} - F_{vi_w}\right]_I + \left[\mathscr{P}_w\right]_I = 0 \quad \text{in } \mathscr{D}. \tag{12.12}$$

The corresponding adjoint boundary condition is produced by equating the subscript "I" boundary terms in equation (12.11) to produce

$$n_i \psi^T \left[F_{i_w} - F_{vi_w}\right]_I = \left[\mathscr{M}_w\right]_I \quad \text{on } \mathscr{B}. \tag{12.13}$$

The remaining terms from equation (12.11) then yield a simplified expression for the variation of the cost function which defines the gradient

$$\delta I = \delta I_{II} + \int_{\mathscr{D}} \psi^T \delta R_{II} d\mathscr{D}_\xi, \tag{12.14}$$

which consists purely of the terms containing variations in the metrics with the flow solution fixed. Hence an explicit formula for the gradient can be derived once the relationship between mesh perturbations and shape variations is defined.

Comparing equations (12.6) and (12.8), the variation δw may be eliminated from (12.11) by equating all field terms with subscript "I" to produce a differential adjoint system governing ψ

$$\frac{\partial \psi^T}{\partial \xi_i} \left[F_{i_w} - F_{vi_w}\right]_I + \mathscr{P}_w = 0 \quad \text{in } \mathscr{D}. \tag{12.15}$$

The corresponding adjoint boundary condition is produced by equating the subscript "I" boundary terms in equation (12.11) to produce

$$n_i \psi^T \left[F_{i_w} - F_{vi_w}\right]_I = \mathscr{M}_w \quad \text{on } \mathscr{B}. \tag{12.16}$$

The remaining terms from equation (12.11) then yield a simplified expression for the variation of the cost function which defines the gradient

$$\delta I = \int_{\mathscr{B}} \left\{ \delta \mathscr{M}_{II} - n_i \psi^T \left[\delta F_i - \delta F_{vi} \right]_{II} \right\} d\mathscr{B}_\xi + \int_{\mathscr{D}} \left\{ \delta \mathscr{P}_{II} + \frac{\partial \psi^T}{\partial \xi_i} \left[\delta F_i - \delta F_{vi} \right]_{II} \right\} d\mathscr{D}_\xi . \tag{12.17}$$

The details of the formula for the gradient depend on the way in which the boundary shape is parameterized as a function of the design variables, and the way in which the mesh is deformed as the boundary is modified. Using the relationship between the mesh deformation and the surface modification, the field integral is reduced to a surface integral by integrating along the coordinate lines emanating from the surface. Thus the expression for δI is finally reduced to the form

$$\delta I = \int_{\mathscr{B}} \mathscr{G} \delta q \, d\mathscr{B}_\xi$$

where q represents the design variables, and $\mathscr{G}$ is the gradient, which is a function defined over the boundary surface.

The boundary conditions satisfied by the flow equations restrict the form of the left hand side of the adjoint boundary condition (12.16). Consequently, the boundary contribution to the cost function $\mathscr{M}$ cannot be specified arbitrarily. Instead, it must be chosen from the class of functions which allow cancellation of all terms containing δw in the boundary integral of equation (12.11). On the other hand, there is no such restriction on the specification of the field contribution to the cost function $\mathscr{P}$, since these terms may always be absorbed into the adjoint field equation (12.15) as source terms.

It is convenient to develop the inviscid and viscous contributions to the adjoint equations separately. Also, for simplicity, it will be assumed that the portion of the boundary that undergoes shape modifications is restricted to the coordinate surface $\xi_2 = 0$. Then equations (12.11) and (12.13) may be simplified by incorporating the conditions

$$n_1 = 0, \quad n_2 = 1, \quad d\mathscr{B}_\xi = d\xi_1,$$

so that only the variations δF_2 and δF_{v2} need to be considered at the wall boundary.

The inviscid and viscous contributions are to be derived separately to get the final form of the adjoint equations. The inviscid contributions are derived in [91, 93]. The viscous contributions are derived in [98, 96] and can be found in [97] as well for 2D case. Determining the contributions from momentum and energy equations, the viscous adjoint field operator will read as follows:

$$\begin{aligned}
(\bar{L}\psi)_1 &= -\frac{p}{\rho^2} \frac{\partial}{\partial \xi_l} \left(S_{lj} \kappa \frac{\partial \theta}{\partial x_j} \right) \\
(\bar{L}\psi)_{i+1} &= \frac{\partial}{\partial \xi_l} \left\{ S_{lj} \left[\mu \left(\frac{\partial \phi_i}{\partial x_j} + \frac{\partial \phi_j}{\partial x_i} \right) + \lambda \delta_{ij} \frac{\partial \phi_k}{\partial x_k} \right] \right\} \\
&\quad + \frac{\partial}{\partial \xi_l} \left\{ S_{lj} \left[\mu \left(u_i \frac{\partial \theta}{\partial x_j} + u_j \frac{\partial \theta}{\partial x_i} \right) + \lambda \delta_{ij} u_k \frac{\partial \theta}{\partial x_k} \right] \right\} - \sigma_{ij} S_{lj} \frac{\partial \theta}{\partial \xi_l} \quad \text{for} \quad i = 1,2 \\
(\bar{L}\psi)_4 &= \frac{1}{\rho} \frac{\partial}{\partial \xi_l} \left(S_{lj} \kappa \frac{\partial \theta}{\partial x_j} \right).
\end{aligned}$$

The conservative viscous adjoint operator is obtained by the transformation

$$L = M^{-1^T} \bar{L}$$

where

$$M^{-1^T} = \begin{pmatrix} 1 & -\frac{u_1}{\rho} & -\frac{u_2}{\rho} & \frac{(\gamma-1)u_i u_i}{2} \\ 0 & \frac{1}{\rho} & 0 & -(\gamma-1)u_1 \\ 0 & 0 & \frac{1}{\rho} & -(\gamma-1)u_2 \\ 0 & 0 & 0 & (\gamma-1) \end{pmatrix}.$$

The cost function that we choose in the present optimization problem is drag reduction. Hence, the cost function, which corresponds equation (12.3), reads as

$$I(w,q) := C_D = \frac{1}{C_{ref}} \int_{\mathscr{B}} C_p \left(\frac{\partial y}{\partial \xi} \cos\alpha - \frac{\partial x}{\partial \xi} \sin\alpha \right) d\xi_1, \tag{12.18}$$

where the surface pressure coefficient is defined by

$$C_p := \frac{2(p - p_\infty)}{\gamma M_\infty^2 p_\infty}. \tag{12.19}$$

The boundary conditions for the adjoint equations on the solid body, corresponding to equation (12.16), are of Neumann-type and for the above mentioned cost function they are given by

$$n_{\xi_1} \psi_2 + n_{\xi_2} \psi_3 = -\frac{2}{\gamma M_\infty^2 p_\infty C_{\text{ref}}} (n_{\xi_1} \cos\alpha + n_{\xi_2} \sin\alpha), \quad \text{on } \mathscr{B}. \tag{12.20}$$

Design equation: For the design equation (8.7c), we need an expression for the derivative of the Lagrangian with respect to the geometry of the airfoil. However, in the actual computation instead of actual gradient (derivative of the Lagrangian) the reduced gradient (derivative of the cost function with respect to the geometry of the airfoil) is used. Hence for the above mentioned cost function, the reduced gradient is given by

$$\delta I = \frac{1}{C_{ref}} \int_{\mathscr{B}} C_p \left(\delta \left(\frac{\partial y}{\partial \xi} \right) \cos\alpha - \delta \left(\frac{\partial x}{\partial \xi} \right) \sin\alpha \right) d\xi_1. \tag{12.21}$$

Gradient smoothing: The reduced gradient obtained using inaccurate state and costate solutions are quite non-smooth, specially near the leading and trailing edges. In order to make sure that each new shape in the optimization sequence remains smooth, it proves essential to smooth the gradient and to replace $\mathscr{G}$ by its smoothed value $\bar{\mathscr{G}}$ in the descent process. This also acts as a preconditioner which allows the use of much larger steps. The gradient smoothing is equivalent to redefining the inner product in a Sobolev space as described in [94], and the steps in the smoothed gradient direction still guarantee descent towards the optimum. To apply second

order smoothing in the ξ_1 direction, for example, the smoothed gradient $\bar{\mathscr{G}}$ may be calculated from a discrete approximation to

$$\bar{\mathscr{G}} - \frac{\partial}{\partial \xi_1} \varepsilon \frac{\partial}{\partial \xi_1} \bar{\mathscr{G}} = \mathscr{G} \tag{12.22}$$

where ε is the smoothing parameter. For higher order smoothing, similar equation of higher order needs to be solved.

Implementation of Navier-Stokes design: In this paper we implement the one-shot approach of the design for Navier-Stokes equations. The method is compared with the continuous adjoint method of A. Jameson and termed as 'Original' in what follows. His design procedure can be summarized as follows:

1. Solve the flow equations (for 10 iterations) for ρ, u_1, u_2, p.
2. Solve the adjoint equations (for 10 iterations) for ψ subject to appropriate boundary conditions.
3. Evaluate $\mathscr{G}$.
4. Project $\mathscr{G}$ into an allowable subspace that satisfies any geometric constraints.
5. Smooth the gradient to get $\bar{\mathscr{G}}$.
6. Update the shape based on the direction of steepest descent.
7. Return to 1 until convergence is reached.

The design procedure of 'one-shot' pseudo-time-stepping method can be summarized as follows:

1. Solve the flow equations (for 2 to 4 iterations) for ρ, u_1, u_2, p.
2. Solve the adjoint equations (for 2 to 4 iterations) for ψ subject to appropriate boundary conditions.
3. Evaluate $\mathscr{G}$.
4. Project $\mathscr{G}$ into an allowable subspace that satisfies any geometric constraints.
5. Smooth the gradient to get $\bar{\mathscr{G}}$.
6. Approximate the reduced Hessian B.
7. Integrate the preconditioned design equation.
8. Update the shape.
9. Return to 1 until convergence is reached.

Remark: It is to note that in the implementations mentioned below, the geometry is not parameterized using (orthogonal) shape functions (such as Hicks-Henne function mentioned in earlier chapters), which is not an easy task and requires user's expertise to come up with an appropriate set of such functions for specific applications. One of the goals of the optimization software has been to free the designer from such mundane task. This has been achieved in SYN103 code. Instead of such parameterization, the discrete grid points on the geometry are used as design parameters, which are usually huge in numbers, specially for viscous computations. The gradient with respect to those large number of parameters are quite non-smooth, specially, at the leading and the trailing edges. Gradient smoothing helps in taking larger design steps by smoothing it.

12.3 Numerical Results and Discussion

The numerical method is applied to 2D test cases for drag reduction with constant lift and constant thickness. The constraints are treated in indirect way as mentioned in Chapter 8. The computational domain is discretized using 512×64 C-grid. On this grid pseudo-unsteady state, costate and design equations are solved. SYN103 code of A. Jameson is used for the computations. The code is modified for one-shot optimization. The constraint of constant lift is maintained by changing angle of incidence. All the points on the airfoil are used as design parameters which are 257 in numbers. All the computations are carried out on a Linux machine with Intel(R) Xeon(TM) processor, CPU 3.00GHz and 8MB RAM.

Case 1: RAE2822 airfoil

In this case the optimization method is applied to RAE2822 airfoil at Mach number 0.75 and Reynolds number $0.600E+07$. The constraint of constant lift coefficient is fixed at 0.65. The optimization is started after 80 iterations of state and 40 iterations of costate solver. The design equation is integrated after every 2 iterations of state and costate runs instead of after every iteration of state and costate run as it should be in order to call it one-shot method. Since the grid is very fine, the residual is not reduced to the minimum level required by the one-shot method. That is why we need 2 iterations of state and costate solver in each optimization update. The optimization requires 32 iterations to converge. After the convergence of the optimization another 80 iterations of state solver is carried out in order to get the force coefficients which are comparable with results obtained by other methods.

For the 'Original' method, the optimization is started after 80 iterations of state and costate solver. The optimization requires 8 iterations to converge. Table 12.1 presents the comparison of number of iterations, force coefficients and CPU time required in both the methods. As can be seen, optimized drag coefficients are same in both the methods. The lift coefficients are also almost the same but the angle of

Table 12.1 Comparison of number of iterations and force coefficients for baseline and optimized RAE2822 airfoil using different optimization iterations

Geometry	Opt. Itr	State Itr	Costate Itr	CD	CL	AL	CPU time
Baseline		300		0.0118	0.6500	1.93276	146.99 Sec
Original	8	230	150	0.0078	0.6498	2.16736	190.16 Sec
One-shot	32	222	102	0.0078	0.6501	2.12036	165.48 Sec

Table 12.2 Comparison of number of iterations and force coefficients for baseline and optimized TAI airfoil using different optimization iterations

Geometry	Opt. Itr	State Itr	Costate Itr	CD	CL	AL	CPU time
Baseline		500		0.0282	0.7504	2.28576	244.94 Sec
Original	16	310	230	0.0099	0.7508	2.40356	273.77 Sec
One-shot	38	328	188	0.0100	0.7506	2.38676	258.26 Sec

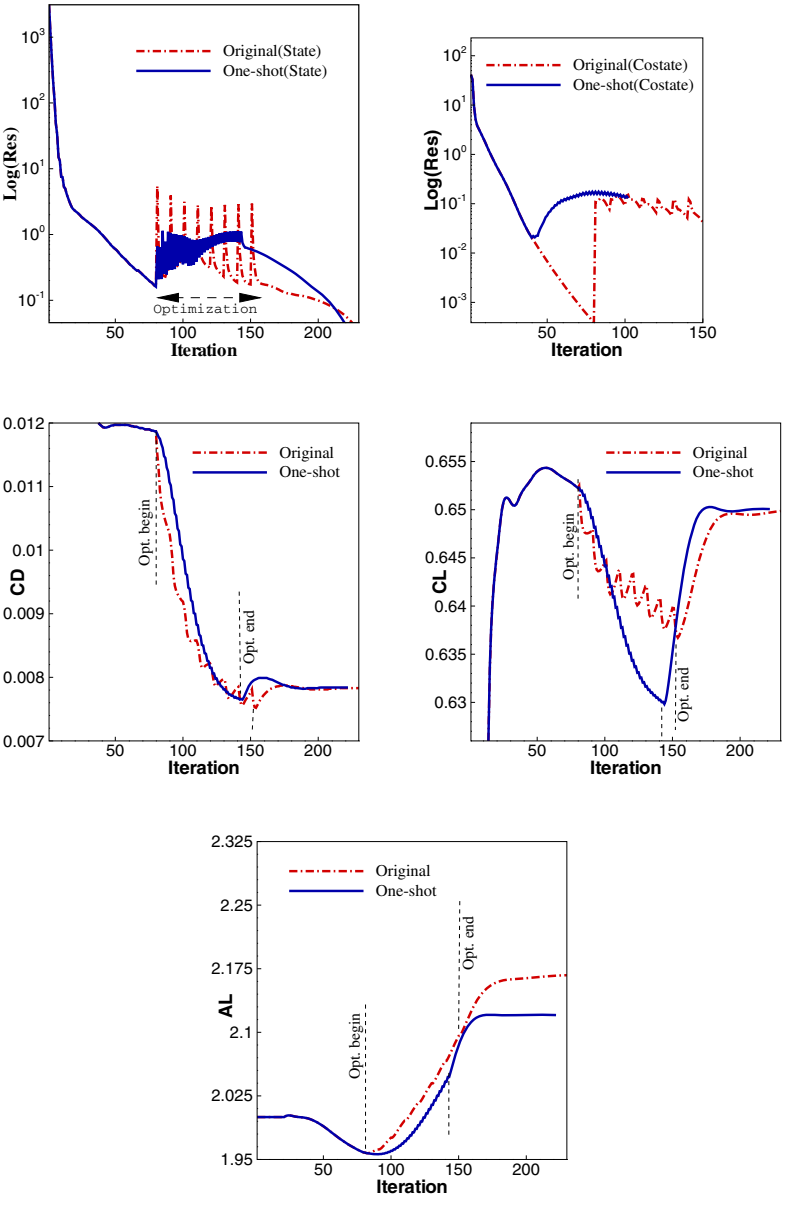
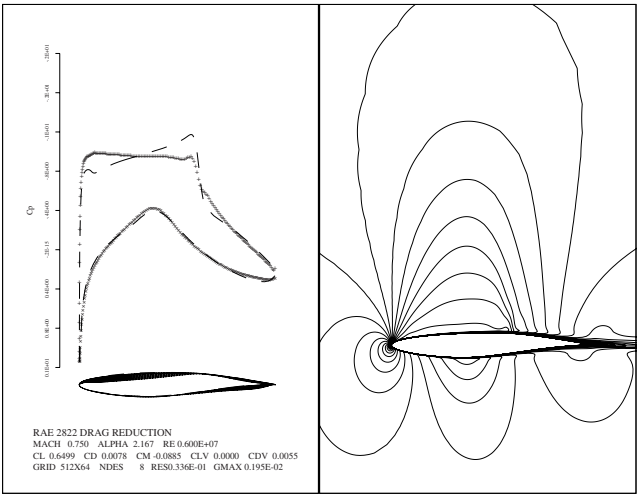
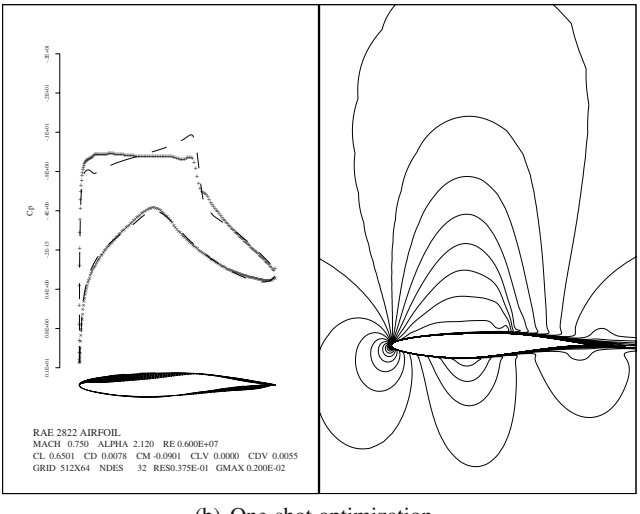


Fig. 12.1 Convergence history of the optimization iterations (Case 1: RAE2822 airfoil)

incidence obtained by one-shot method has smaller value. Also, one-shot method requires less CPU time to converge. The convergence histories of both the optimization methods are presented in Figure 12.1. The surface pressure distributions



(a) Original Optimization



(b) One-shot optimization

Fig. 12.2 Pressure distribution and Mach contours for the RAE2822 airfoil

and Mach contours are presented in Figure 12.2. Both the optimized airfoils, surface pressure distributions and Mach contours look quite similar.

Case 2: TAI airfoil

In this case the optimization method is applied to TAI airfoil at Mach number 0.65 and Reynolds number $0.600E+07$. The constraint of constant lift coefficient is fixed at 0.75. The optimization is started after 80 iterations of state and 40 iterations of costate solver. The design equation is integrated after every 4 iterations of state and costate runs. The optimization requires 38 iterations to converge. After the

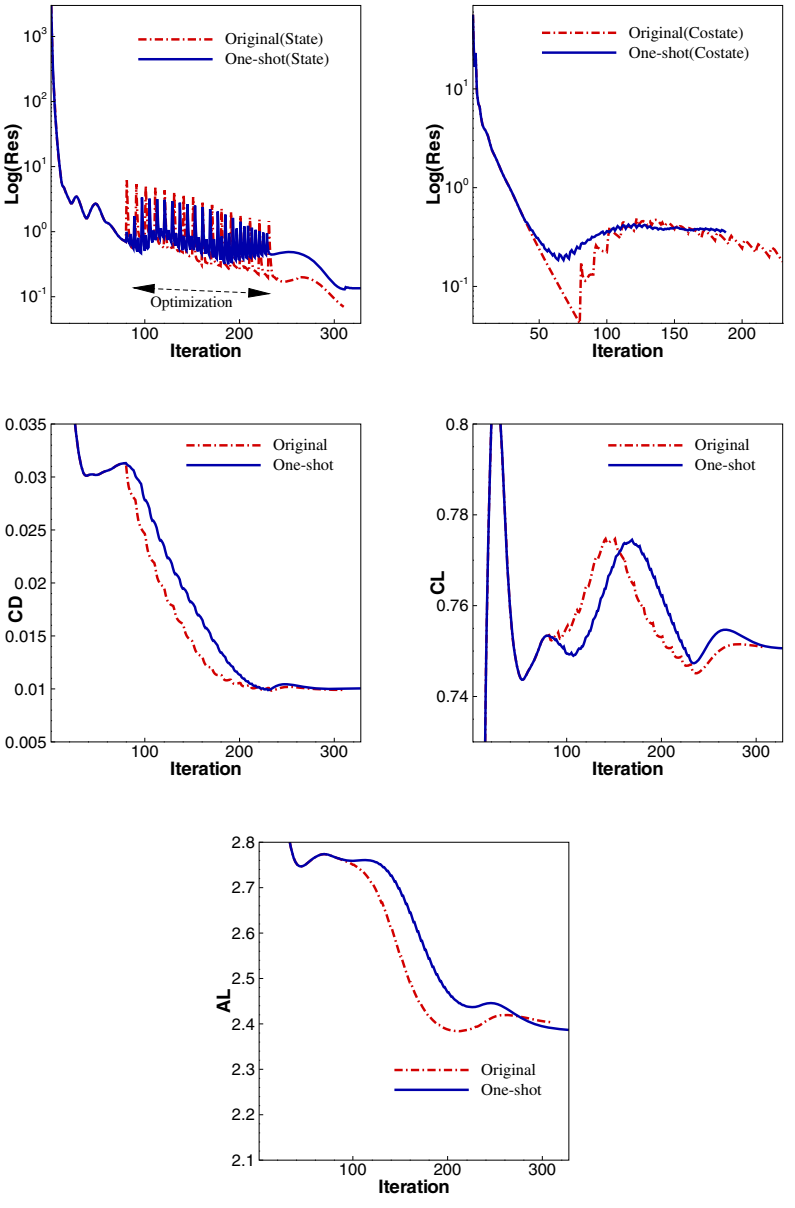
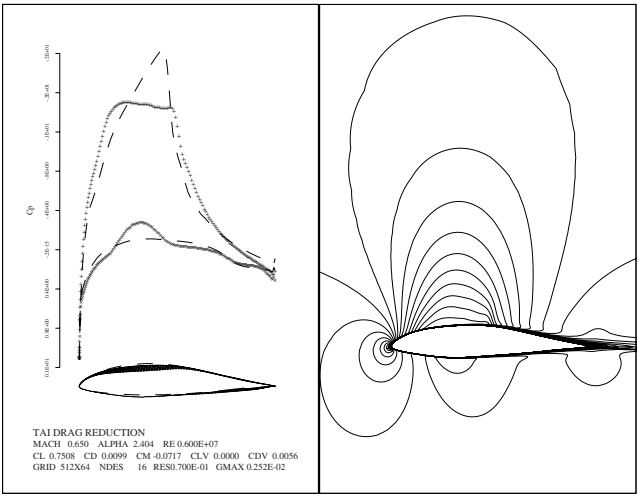
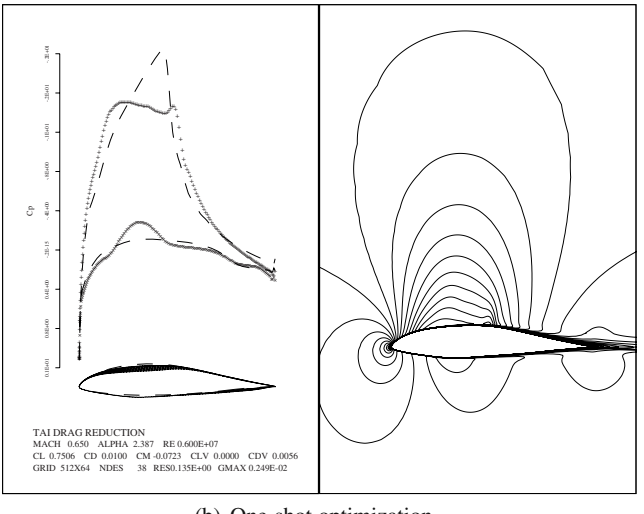


Fig. 12.3 Convergence history of the optimization iterations (Case 2: TAI airfoil)

convergence of the optimization another 100 iterations of state solver is carried out in order to get the force coefficients which are comparable with results obtained with other methods.



(a) Original Optimization



(b) One-shot optimization

Fig. 12.4 Pressure distribution and Mach contours for the TAI airfoil

For the 'Original' method, the optimization is started after 80 iterations of state and costate solver. The optimization requires 16 iterations to converge. Table 12.2 presents the comparison of number of iterations, force coefficients and the CPU time required for the convergence of the methods. In this case force coefficients are almost the same, angle of incidence obtained by one-shot method and the CPU time required by this method is little less than that resulted by the original method. The convergence history of the optimization method is presented in Figure 12.3. The

surface pressure distribution and Mach contour are presented in Figure 12.4. The optimized quantities obtained by both the methods are again quite similar.

12.4 Conclusions

One-shot pseudo-time-stepping method is applied successfully to shape optimization problems in aerodynamics using viscous compressible flow. The method works efficiently as in case of applications using inviscid compressible flow. In the convergence histories of the one-shot method, linear convergence with respect to the objective function is observed. The iteration step in the design space is so small that the process truly reflects a continuous behavior.

The optimized shapes from the two methods are very similar and have the same performance -in this respect it is important to note that there is no reason to believe there is a unique optimum shape since any shock-free airfoil should have the same performance as long as the skin friction remains the same. The two methods have roughly equal computational costs, depending on tuning data parameters such as step size, smoothing parameters, no of iterations in the flow and adjoint solutions, etc. Perhaps this is not so surprising because when the original method is run without fully converging the intermediate flow and adjoint solutions it can properly be regarded as a variant of a one shot method. The over all computational cost is less than 2 times the forward simulation runs. Further acceleration in convergence of the one-shot method can be achieved using the multigrid strategy discussed in the previous chapters. The method can be applied to problems of high-lift configuration as well as to multidisciplinary optimization in this problem class.

References

1. I (1967). International formulation committee. a formulation of the thermodynamic properties of ordinary water substance. Technical report, IFC Sekretariat, Düsseldorf, Germany (1967)
2. Akcelik, V., Biros, G., Ghattas, O.: Parallel multiscale gauss-newton-krylov methods for inverse wave propagation. In: IEEE/ACM SC 2002 Conference. IEEE/ACM (2002)
3. Anderson, D.A., Tannehill, J.C., pletcher, R.H.: Computational Fluid Mechanics and Heat Transfer. McGraw-Hill, New York (1984)
4. Anderson, W.K., Venkatakrishnan, V.: Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. In: AIAA paper 97-0643, 35th Aerospace Sciences Meeting and Exhibit, Reno, Nevada (January 1997)
5. Arian, E., Ta'asan, S.: Analysis of the hessian for aerodynamics optimization: Inviscid flow. Technical Report No.96-28, ICASE (1996)
6. Arian, E., Vatsa, N.V.: A preconditioning method for shape optimization governed by the euler equations. Technical Report No.98-14, ICASE (1998)
7. Baldwin, B., Lomax, H.: Thin layer approximation and algebraic model for separated turbulent flow. AIAA Paper 78-257 (1978)
8. Bastian, P.: Numerical computation of multiphase flows in porous media. Habilitationsschrift, Christian-Albrechts-Universität Kiel, Germany (1999)
9. Bastian, P., Helmig, R.: Efficient fully-coupled solution techniques for two-phase flow in porous media: parallel multigrid solution and large scale computations. Adv. Water Resour. 23, 199–216 (1999)
10. Batchelor, G.K.: An Introduction to Fluid Dynamics. Cambridge University Press, Cambridge (1967)
11. Battermann, A., Heinkenschloss, M.: Preconditioners for karush-kuhn-tucker systems arising in the optimal control of distributed systems. In: Kappel, F., Desch, W., Kunisch, K. (eds.) Optimal Control of PDE, pp. 15–32. Birkhäuser, Basel (1996)
12. Battermann, A., Sachs, E.W.: Block preconditioners for KKT systems in PDE-governed optimal control problems. In: Hoppe, R.H.W., Hoffmann, K.H., Schulz, V. (eds.) Fast Solution of Discretized Optimization Problems, pp. 1–18. Birkhäuser, Basel (2001)
13. Bear, J.: Hydraulics of Graundwater. McGraw-Hill, New York (1979)
14. Behrman, W.: An efficient gradient flow method for unconstrained optimization. PhD thesis, Department of Scientific Computing and Computational Mathematics, Stanford University (1998)
15. Biegler, L.T., Ghattas, O., Heinkenschloss, M., Keyes, D., van Bloemen Waanders, B.: Real-Time PDE-Constrained optimization. SIAM, Philadelphia (2007)

16. Biegler, L.T., Ghattas, O., Heinkenschloss, M., van Bloemen Waanders, B.: Large-Scale PDE-Constrained optimization. Springer, Berlin (2003)
17. Biros, G., Ghattas, O.: Parallel lagrange-newton-krylov-schur methods for PDE-constrained optimization. Part i: The krylov-schur solver. SIAM J. Sci. Comput. 27, 687–713 (2005)
18. Blazek, J.: Computational Fluid Dynamics: Principles and Applications. Elsevier Science, Oxford (2001)
19. Bock, H.G., Egartner, W., Kappis, W., Schulz, V.: Practical shape optimization for turbine and compressor blades. Optimization and Engineering 3, 395–414 (2002)
20. Bock, H.G.: Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen. Bonner Mathematische Schriften 183, Bonn(1987)
21. Boggs, P.T.: Sequential quadratic programming. In: Iserles, A. (ed.) Acta Numerica, pp. 1–51. Cambridge University Press, Cambridge (1995)
22. Brandt, A.: Multigrid techniques: 1984 guide with applications to fluid dynamics. GMD-Studien 85, Gesellschaft fuer Mathematik und Datenverarbeitung mbH, Bonn, Germany (1984)
23. Brezillon, J.: Application of the adjoint technique with the optimization framework synaps pointer pro. In: Kroll, N., Fassbender, J. (eds.) Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 89. Springer, Heidelberg (2002)
24. Brezillon, J., Gauger, N.R.: 2d and 3d aerodynamic shape optimization using the adjoint approach. AST Journal 8, 715–727 (2004)
25. Brooks, R.H., Corey, A.J.: Hydraulic properties of porous media. Colorado State University, Fort Collins (1964)
26. Burdine, N.: Relative permeability calculations from pore–size distribution data. Technical Report AIME, Petroleum Transactions (1953)
27. Chavent, G., Jaffre, J., Jegou, S., Liu, J.: A symbolic code generator for parameter estimation. In: Proceedings of the SIAM Workshop on Computational Differentiation, pp. 129–136. SIAM, Philadelphia (1996)
28. Chorin, A.J., Marsden, J.E.: A Mathematical Introduction to Fluid Mechanics. Springer, New York (1993)
29. Class, H.: Theorie und numerische Modellierung nichtisothermer Mehrphasenprozesse in NAPL–kontaminierten porösen Medien. PhD thesis, Institut für Wasserbau, Universität Stuttgart (2001)
30. Class, H., Helmig, R., Bastian, P.: Numerical simulation of nonisothermal multiphase multicomponent processes in porous media - 1. an efficient solution technique. Adv. Water Resour. 25, 533–550 (2002)
31. Cliff, E.M., Shenoy, A., Heinkenschloss, M.: Airfoil design by all-at-once method. 97-15, Department of Computational and Applied Mathematics, Rice University (1997)
32. Courant, R.: Variational methods for the solution of problems of equilibrium and vibrations. Bull. Amer. Math. Soc. 49, 1–23 (1943)
33. Deufelhard, P.: Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms. Springer, Berlin (2004)
34. Dieses, A.: Numerical methods for optimization problems in water flow and reactive solute transport processes of xenobiotics in soils. PhD thesis, University of Heidelberg, Report SFB Preprint 2001-07 (2001)
35. Dieses, A.E., Schlöder, J.P., Bock, H.G., Richter, O.: Parameter estimation for nonlinear transport and degradation processes of xenobiotica in soil. In: Keil, F., et al. (eds.) Scientific computing in Chemical Engineering II, vol. 2, pp. 290–297. Springer, Heidelberg (1999)

36. Ewing, R.E., Lazarov, R.D., Lin, Y.: Finite volume element approximations of nonlocal in time one-dimensional flows in porous media. Computing 64, 157–183 (2000)
37. Faerber, A.: Wärmetransport in der ungesättigten Bodenzone: Entwicklung einer thermischen in-situ Sanierungstechnologie. PhD thesis, Institut für Wasserbau, Universität Stuttgart (1997)
38. Fahl, M.: Trust-region methods for flow control based on reduced order modellin. PhD thesis, University of Trier (2000)
39. Fahl, M., Sachs, E.W.: Reduced order modelling approaches to PDE-constrained optimization based on proper orthogonal decomposition. In: Biegler, L., Ghattas, O., Heinkenschloss, M., van Bloemen Waanders, B. (eds.) Large-Scale PDE-Constrained Optimization, pp. 268–280. Springer, Heidelberg (2003)
40. Feng, D., Pulliam, H.T.: An all-at-once reduced SQP scheme for aerodynamic design and optimization. RIACS Tech. Rep. 95.19 (1995)
41. Finsterle, S.: Multiphase inverse modeling: An overview. In: Proceedings of the DOE Geothermal Program Review XVI, Berkeley, CA, April 1-2, 1998, pp. 33–39 (1998)
42. Fletcher, C.A.J.: Computational Techniques for Fluid Dynamics, Vol. I & II. Springer, Berlin (1988)
43. Fletcher, R.: Practical Methods of Optimization. John Wiley & Sons, New York (1987)
44. Fromman, O.: Synapspointerpro v2.50. Synaps Ingenieure Gesellschaft mbH, Bremen, Germany (2002)
45. Gauger, N.R.: Aerodynamic shape optimization using the adjoint euler equations. In: Proceedings of the GAMM workshop on Discrete Modelling and Discrete Algorithms in Continuoum Mechanics, pp. 87–96. Logos Verlag, Berlin (2001)
46. Gauger, N.R.: Das adjungiertenverfahren in der aerodynamischen formoptimierung. DLR-Report DLR-FB-2003-05, DLR, Germany (2003)
47. Gauger, N.R., Brezillon, J.: Aerodynamic shape optimization using adjoint method. J. Aero. Soc. of India 54, 247–254 (2002)
48. Ghattas, O., Bark, J.: Optimal control of two- and three-dimensional navier-stokes flows. Journal of Comp. Phys. 136, 231–244 (1997)
49. Giles, M.B., Pierce, N.A.: An introduction to the adjoint approach to design. Flow, Turbulence and Combustion 65, 393–415 (2000)
50. Gill, P.E., Jay, L.O., Leonard, M.W., Petzold, L.R., Sharma, V.: An SQP method for the optimal control of large scale dynamical systems. J. Comput. App. Math. 120, 197–213 (2000)
51. Gill, P.E., Murry, W., Wright, M.H.: Practical Optimization. Academic Press, London (1981)
52. Griewank, A.: Projected hessians for preconditioning in one-step one-shot design optimization. In: Large Scale Nonlinear Optimization. Kluwer Academic Publishers, Dordrecht (2005)
53. Hackbusch, W.: Multigrid Methods and Applications, vol. 4. Springer, Berlin (1985)
54. Hackbusch, W.: Iterative Solution of Large Sparse Systems of Equations. Springer, Heidelberg (1994)
55. Hadamard, J.: Memoire sur le probleme d'analyse relatif a l'equilibre des plaques elastiques encastrees. Memoeres presentes par divers savants a' l'Academie des Sciences de l'Institut National de France, Series 2 33(4), 1–128 (1908)
56. Hazra, S.B.: Shock Capturing in Computation of High Speed Inviscid and Viscous Flow Past Airfoils. PhD thesis, Department of Mathematics, Indian Institute of Technology, Kharagpur, India (1997)
57. Hazra, S.B.: Computation of high-speed flow using non-oscillatory scheme. ISNM 129, 465–474 (1999)

58. Hazra, S.B.: An efficient method for aerodynamic shape optimization. In: AIAA paper 2004-4628, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York (September 2004)
59. Hazra, S.B.: Reduced hessian updates in simultaneous pseudo-timestepping for aerodynamic shape optimization. In: 44th AIAA Aerospace Sciences Meeting and Exhibit AIAA paper 2006-933, Reno, Nevada (January 2006)
60. Hazra, S.B.: Direct treatment of state constraints in aerodynamic shape optimization using simultaneous pseudo-time-stepping. AIAA Journal 45, 1988–1997 (2007)
61. Hazra, S.B.: Multigrid one-shot method for aerodynamic shape optimization. SIAM J. Sci. Comput. 30, 1527–1547 (2008)
62. Hazra, S.B.: Multigrid one-shot method for state constrained aerodynamic shape optimization. SIAM J. Sci. Comput. 30, 3220–3248 (2008)
63. Hazra, S.B., Chakrabortty, S.K., Niyogi, P.: Study in non-oscillatory schemes for computational solution of euler equations. Journal of CFD 7, 163–176 (1998)
64. Hazra, S.B., Chakrabortty, S.K., Niyogi, P.: Compressible viscous flow past airfoils using non-oscillatory finite-volume schemes. Journal of CFD 8, 400–409 (1999)
65. Hazra, S.B., Gauger, N.: Simultaneous pseudo-timestepping for aerodynamic shape optimization. In: PAMM, vol. 5, pp. 743–744 (2005)
66. Hazra, S.B., Jameson, A.: One-shot pseudo-time method for aerodynamic shape optimization using the navier-stokes equations. In: 45th AIAA Aerospace Sciences Meeting and Exhibit AIAA paper 2007-1470, Reno, Nevada (January 2007)
67. Hazra, S.B., Schulz, V.: Simultaneous pseudo-timestepping for pde-model based optimization problems. BIT Numerical Mathematics 44, 457–472 (2004)
68. Hazra, S.B., Schulz, V.: Simultaneous pseudo-timestepping for aerodynamic shape optimization problems with state constraints. SIAM J. Sci. Comput. 28, 1078–1099 (2006)
69. Hazra, S.B., Schulz, V.: Simultaneous pseudo-timestepping for state constrained optimization problems in aerodynamics. In: Biegler, L., Ghattas, O., Heinkenschloss, M., Keyes, D., van Bloemen Waanders, B. (eds.) Real-Time PDE-Constrained Optimization, pp. 169–182. SIAM, Philadelphia (2007)
70. Hazra, S.B., Schulz, V., Brezillon, J.: Simultaneous pseudo-timestepping for 3d aerodynamic shape optimization. J. Numer. Math. 16, 139–161 (2008)
71. Hazra, S.B., Schulz, V., Brezillon, J., Gauger, N.R.: Aerodynamic shape optimization using simultaneous pseudo-timestepping. J. Comp. Phys. 204, 46–64 (2005)
72. Hazra, S.B., Class, H., Helmig, R., Schulz, V.: Forward and inverse problems in modeling of multiphase flow and transport through porous media. Computational Geosciences 8, 21–47 (2004)
73. Hazra, S.B., Schulz, V.: Numerical parameter identification in multiphase flow through porous media. J. of Comput. Visual. Sci. 5, 107–113 (2002)
74. Hazra, S.B., Schulz, V.: On efficient computation of the optimization problem arising in the inverse modeling of non-stationary multiphase multicomponent flow through porous media. Comput. Opt. and Appl. 31, 69–85 (2005)
75. Helmig, R.: Multiphase flow and transport processes in the subsurface - a contribution to the modeling of hydrosystems. Springer, Heidelberg (1997)
76. Helmig, R., Class, H., Faerber, A., Emmert, M.: Heat transport in the unsaturated zone - comparison of experimental results and numerical simulations. J. of Hydr. Reas. 36 (1998)
77. Helmig, R., Class, H., Huber, R., Sheta, H., Ewing, J., Hinkelmann, R., Jakobs, H., Bastian, P.: Architecture of the modular program system mufte_ug for simulating multiphase flow and transport processes in heterogeneous porous media. Mathematische Geologie 2 (1998)

78. Hicks, R.M., Henne, P.A.: Wing design by numerical optimization. Journal of Aircraft 15, 407–412 (1978)
79. Hintermueller, M., Kunisch, K.: Inverse problems for elastohydrodynamic models. ZAMM 81, 17–20 (2001)
80. Hinze, M., Pinnau, R., Ulbrich, M., Ulbrich, S.: Optimization with PDE Constraints. Springer, Heidelberg (2009)
81. Hirsch, C.: Numerical Computation of Internal and External Flows, Vol. I & II. John Wiley & Sons, Chichester (1988)
82. Holt, A., Marten, L.: Aerodynamics of Wings and Bodies. Dover Publications, Mineola (1965)
83. Hoppe, R.H.W., Petrova, S., Schulz, V.: A primal-dual newton-type interior-point method for topology optimization. J. Optm. Theory Appl. 114, 545–571 (2002)
84. Hörmander, L.: Pseudo-differential operators. Comm. Pure Appl. Math. 18, 501–517 (1965)
85. Hou, G.W., Taylor III, A.C., Mani, S.V., Newman, P.A.: Formulation for simultaneous aerodynamic analysis and design optimization. NASA Tech. Rep. NASA-CR-201036, NASA (1993)
86. Iollo, A., Kuruvila, G., Ta'asan, S.: Pseudo-time method for optimal shape design using euler equations. Technical Report No.95-59, ICASE (1995)
87. Ito, K., Kunisch, K., Gherman, I., Schulz, V.: Approximate nullspace iteration for KKT systems in model based optimization. Technical Report Forschungsbericht Nr.06-5, FB IV Mathematik, Universität Trier, Germany (2006)
88. Anderson JR., J.D.: Computational Fluid Dynamics. McGraw-Hill, New York (1995)
89. Jameson, A.: Solution of the Euler equations for two dimensional transonic flow by a multigrid method. Applied Mathematics and Computations 13, 327–356 (1983)
90. Jameson, A.: Aerodynamic design via control theory. Journal of Scientific Computing 3, 233–260 (1988)
91. Jameson, A.: Automatic design of transonic airfoils to reduce the shock induced pressure drag. In: Proceedings of the 31st Israel Annual Conference on Aviation and Aeronautics, Tel Aviv, February 1990, pp. 5–17 (1990)
92. Jameson, A.: Optimum aerodynamic design using CFD and control theory. In: AIAA paper 95-1729, AIAA 12th Computational Fluid Dynamics Conference, San Diego, CA (June 1995)
93. Jameson, A.: Optimum aerodynamic design using control theory. Computational Fluid Dynamics Review, 495–528 (1995)
94. Jameson, A.: Efficient aerodynamic shape optimization. In: 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference AIAA paper 2004-4369, Albany, New York, Augest 30-September 1 (2004)
95. Jameson, A., Kim, S.: Reduction of the adjoint gradient formula in the continuous limit. In: AIAA 41st Aerospace Sciences Meeting & Exhibit AIAA paper 2003-040, Reno, NV (January 2003)
96. Jameson, A., Martinelli, L., Pierce, N.A.: Optimum aerodynamic design using the Navier-Stokes equations. Theoret. Comput. Fluid Dynamics 10, 213–237 (1998)
97. Jameson, A., Nadarajah, S.: Studies of the continuous and discrete adjoint approaches to viscous automatic aerodynamic shape optimization. In: 15th AIAA Computational Fluid Dynamics Conference AIAA paper 2003-040, Anaheim, California, June 11-14 (2001)
98. Jameson, A., Pierce, N., Martinelli, L.: Optimum aerodynamic design using the Navier-Stokes equations. In: 35th Aerospace Sciences Meeting and Exhibit AIAA paper 97-0101, Reno, Nevada (January 1997)

99. Jameson, A., Reuther, J.: Control theory based airfoil design using the Euler equations. In: 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization AIAA paper 94-4272, Panama City Beach, FL (September 1994)
100. Jameson, A., Schmidt, W., Turkel, E.: Numerical solutions of the Euler equations by finite volume methods with Runge-Kutta time stepping schemes. In: AIAA paper 81-1259 (January 1981)
101. Kameswaran, S., Staus, G., Biegler, L.T.: Parameter estimation of core flood and reservoir models. Computers and Chemical Engg. 29, 1787–1800 (2005)
102. Kelley, C.T., Keyes, D.E.: Convergence analysis of pseudo-transient continuation. SIAM J. Num. Anal. 35, 508–523 (1998)
103. Kim, H.J., Sasaki, D., Obayahi, S., Nakahashi, K.: Aerodynamic optimization of supersonic transport wing using unstructured adjoint method. AIAA Journal 39, 1011–1020 (2001)
104. Knabner, P., Igler, B.: Structural identification of nonlinear coefficient functions in transport processes through porous media. In: Hans-Joachim, B., et al. (eds.) Lectures on Applied Mathematics. Springer, Berlin (1999)
105. Kohn, J.J., Nirenberg, L.: On the algebra of pseudo-differential operators. Comm. Pure Appl. Math. 18, 269–305 (1965)
106. Kool, J.B., Parker, J.C., Van Genuchten, M.T.: Parameter estimation for unsaturated flow and transport models - a review. J. of Hydrology 91, 255–293 (1987)
107. Kroll, N., Jain, R.K.: Solution of two-dimensional euler equations- experience with a finite volume code. DFVLR-Report DFVLR-IB-129-84/19, DFVLR, Germany (1984)
108. Kroll, N., Rossow, C.C., Becker, K., Thiele, F.: The megaflow - a numerical flow simulation system. In: Proceedings of the 21st ICAS Symposium, Melbourne, Australia, Paper 98-2.7.4 (1998)
109. Kroll, N., Rossow, C.C., Becker, K., Thiele, F.: The megaflow - a numerical flow simulation system. Aerosp. Sci. Technol. 4, 223–237 (2000)
110. Kunisch, K., Vexler, B.: On the choice of the cost functional for optimal vortex reduction for instationary flows. SIAM J. Control and Optimization (2007) (to appear)
111. Kunisch, K., Volkwein, S., Xie, L.: Hjb-pod based feedback design for the optimal control of evolution problems. SIAM J. on Applied Dynamical Systems 4, 701–722 (2004)
112. Kupfer, F.S.: An infinite dimensional convergence theory for reduced SQP methods in hilbert space. SIAM J. optimization 6, 126–163 (1996)
113. Kuruvila, G., Ta'asan, S., Salas, M.: Airfoil optimization by the one-shot method. In: AGARD-FDP-VKI, April 1994, VKI (1994)
114. Lenhard, R., Parker, J., Mishra, S.: On the correspondence between brooks–corey and van genuchten models. J. of Irrigation and Drainage Engg. 115, 744–751 (1989a)
115. Levenberg, K.: A method for the solution of certain nonlinear problems in least squares. Q. Appl. Math. 2, 164–168 (1944)
116. LeVeque, R.J.: Numerical Methods for Conservation Laws. Birkhäuser, Basel (1990)
117. Lewis, R.M., Nash, S.G.: Multigrid approach to the optimization of systems governed by differential equations. In: AIAA-paper AIAA-2000-4890, AIAA (2000)
118. Lewis, R.M., Nash, S.G.: Model problems for the multigrid optimization of systems governed by differential equations. SIAM J. Sci. Comput. 26(6), 1811–1837 (2005)
119. Libby, P.A.: Introduction to Turbulence. Taylor and Francis, Washington DC (1996)
120. Lions, J.L.: Optimal Control of Systems Governed by Partial Differential Equations. Springer, Heidelberg (1971)
121. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Prog. 45, 503–528 (1989)

122. Logashenko, D., Maar, B., Schulz, V., Wittum, G.: Optimal geometrical design of bingham parameter measurement devices. International Series of Numerical Mathematics (ISNM) 138, 167–183 (2001)
123. Marquardt, D.W.: An algorithm for least squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. 11, 431–441 (1963)
124. McCormick, S.F.: Multilevel Adaptive Methods for Partial Differential Equations. SIAM, Philadelphia (1989)
125. Meyer, R.E.: Introduction to Mathematical Fluid Dynamics. Wiley-Interscience, New York (1971)
126. Mohammadi, B., Pironneau, O.: Applied shape optimization for fluids. Oxford University Press, Oxford (2001)
127. Mualem, Y.: A new model for predicting the hydraulic conductivity of unsaturated porous media. Water Resour. Res. 12, 513–522 (1976)
128. Nash, S.G.: A multigrid approach to discretized optimization problems. Opt. methods and software 14, 99–116 (2000)
129. Nemec, M., Zingg, D.W.: From analysis to design of high-lift configurations using a newton-krylov algorithm. Technical report, ICASE (2002)
130. Niyogi, P., Chakrabartty, S.K., Laha, M.K.: Introduction to Computational Fluid Dynamics. Pearson Education, Singapore (2005)
131. Nocedal, J.: Updating quasi-newton matrices with limited storage. Math. of Comput. 35(151), 773–782 (1980)
132. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
133. Orozco, C.E., Ghattas, O.N.: Jacobian and hessian sparsity in simultaneous and nested sturctural optimization. AIAA paper 91-1093-CP (1991)
134. Orozco, C.E., Ghattas, O.N.: Optimal design of systems governed by nonlinear partial differential equations. In: 4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization AIAA paper 92-4836-CP, Cleveland, OH, September 21-23 (1992)
135. Orozco, C.E., Ghattas, O.N.: Sparse approach to simultaneous analysis design of geometrically nonlinear structures. AIAA Journal 30, 1877–1885 (1992)
136. Parker, J., Lenhard, R.: A model for hysteretic constitutive relations governing multiphase flow, 1. saturation–pressure relations. Water Resour. Res. 23, 2187–2196 (1987)
137. Petzold, L., Rosen, J.B., Gill, P.E., Jay, L.O., Park, K.: Numerical optimal control of parabolic PDES using dasopt. In: Biegler, et al. (eds.) Large Scale Optimization with Applications, Part II. Springer, Heidelberg (1997)
138. Pironneau, O.: On optimum profiles in stokes flow. J. Fluid Mech. 59, 117–128 (1973)
139. Pironneau, O.: On optimum design in fluid mechanics. J. Fluid Mech. 64, 97–110 (1974)
140. Pironneau, O.: Optimal shape design for elliptic systems. Springer, New York (1982)
141. Prager, W.: Introduction to Mechanics of Continua, Ginn, Boston (1961)
142. Reuther, J., Jameson, A.: Aerodynamic shape optimization of wing and wing-body configurations using control theory. In: AIAA 33rd Aerospace Sciences Meeting, AIAA paper 95-0123, Reno, Nevada (January 1995)
143. Reuther, J., Jameson, A., Farmer, J., Martinelli, L., Saunders, D.: Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. In: 34th Aerospace Sciences Meeting and Exhibit, AIAA paper 96-0094, Reno, Nevada (January 1996)
144. Schlichting, H.: Boundary Layer Theory. McGraw-Hill, New York (1979)

145. Schlöder, J.: Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung. Bonner Mathematische Schriften 187 (1988)
146. Schmidt, S.: Indifferenzkurven in der mehrzieloptimierung. Diplomarbeit, Fachbereich IV, Mathematik, Universität Trier, Germany (2006)
147. Schulz, V., Bardossy, A., Helmig, R.: Conditional statistical inverse modeling in groundwater flow by multigrid methods. Computational Geosciences (1999) (to appear)
148. Schulz, V., Bock, H.G., Steinbach, M.: Exploiting invariants in the numerical solution of multipoint boundary value problems for daes. SIAM J. Sci. Comput. 19, 440–467 (1998)
149. Schulz, V.H., Bock, H.G., Steinbach, M.C.: Exploiting invariants in the numerical solution of multipoint boundary value problems for dae. SIAM J. Sci. Comput. 19, 440–467 (1998)
150. Schulz, V.H.: Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots. PhD thesis, University of Heidelberg (1996)
151. Schulz, V.H.: Solving discretized optimization problems by partially reduced SQP methods. Computing and Visualization in Sciences 1(2), 83–96 (1998)
152. Schulz, V.H., Wittum, G.: Multigrid optimization methods for stationary parameter identification problems in groundwater flow. In: Hackbusch, W., Wittum, G. (eds.) Multigrid Methods V. Lecture Notes in Computational Science and Engineering 3, pp. 276–288. Springer, Heidelberg (1998)
153. Schulz, V.H. (guest-editor): Special issue on SQP-based direct discretization methods for practical optimal control problems. Journal of Computational and Applied Mathematics 120(1-2) (2000)
154. von Schwerin, M., Deutschmann, O., Schulz, V.: Process optimization of reactive systems by partially reduced sqp methods. Computers & Chemical Engineering 24, 89–97 (2000)
155. Sheta, H.: Einfluss der Hysterese bei Infiltrations– und Ausbreitungsvorgängen in der gesättigten und ungesättigten Bodenzone. PhD thesis, Institut für Wasserbau, Universität Stuttgart (1999)
156. Ta'asan, S.: Pseudo-time methods for constrained optimization problems governed by pde. Technical Report No.95-32, ICASE (1995)
157. Ta'asan, S.: Multigrid one-shot methods and design strategy. Lecture Notes on Optimization 4. VKI (1997)
158. Ta'asan, S., Kuruvila, G., Salas, M.D.: Aerodynamic design and optimization in one shot. Technical Report AIAA 92-0025, AIAA 30th Aerospace Sciences Meeting and Exhibit, Reno, Nevada (January 1992)
159. Thevenin, D., Janiga, G.: Optimization and Computational Fluid Dynamics. Springer, Heidelberg (2008)
160. Waanders, B.v.B., Bartlett, R., Long, K., Boggs, P., Salinger, A.: Large scale non-linear programming for PDE constrained optimization. SAND 2002-3198, Sandia National Laboratories, Albuquerque, NM and Livermore, CA (October 2002)
161. van Genuchten, M.T.: A closed form equation for predicting the hydraulic conductivity of unsaturated soils. Soil Sci. Soc. Am. J. 44, 892–898 (1980)
162. Weinerfelt, P., Enoksson, O.: Numerical methods for aerodynamic optimization. CFD Journal 9, 758–765 (2001)
163. Wesseling, P.: Principles of Computational Fluid Dynamics. Springer, Berlin (2001)
164. Wilcox, D.C.: Turbulence Modeling for CFD. DCW Industries Inc., La Canada (1993)
165. Wright, T.G.: Eigtool. Oxford University, Oxford, `http://www.comlab.ox.ac.uk/pseudospectra/eigtool/`

166. Xu, C., Follmann, P.M., Biegler, L.T., John, M.S.: Numerical simulation and optimization of a direct methanol fuel cell. Computers and Chemical Engg. 29, 1849–1860 (2005)
167. Xue, M., Wang, D., Gao, J., Brewster, K., Droegemeier, K.K.: The advanced regional prediction system (arps), strom-scale numerical weather prediction and data assimilation. Meteorology and Atmospheric Physics 82, 139–170 (2003)
168. Yeh, W.G.: Review of parameter estimation procedures in groundwater hydrology: The inverse problem. Water Resour. Res. 22, 95–108 (1986)

Index