SIMULTANEOUS ANALYSIS AND DESIGN IN PDE-CONSTRAINED OPTIMIZATION

A DISSERTATION
SUBMITTED TO THE INSTITUTE FOR
COMPUTATIONAL AND MATHEMATICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Youngsoo Choi
December 2012

This dissertation is online at: http://purl.stanford.edu/yd710td7711

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Walter Murray, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Charbel Farhat

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marco Pavone

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael Saunders

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

New methods for solving certain types of PDE-constrained optimization problems are presented in this thesis. The approach taken is to augment state-of-the-art PDE methods. The PDE variables and the optimization variables are solved for simultaneously. This impacts the PDE method by changing the core from solving a system of nonlinear equations to that of finding a nonlinear saddle point. This in turn alters the character of the linear equations that are solved at each iteration.

In the problem we address, the objective function has to match a given target state. Both volume and boundary controls are considered in order to match the target. Regularization is added to the objective function to aid stability and to facilitate computing the solution of the linear systems that arise within the algorithm. Solving such linear systems has been the focus of much research, with many methods being proposed. How to do this simultaneously and efficiently for the specific systems that arise in the PDE-constrained optimization problems of interest is the main focus of our work.

The new methods have been implemented by modifying the cutting-edge software AERO-S. Numerical results are presented for a variety of problems including a flapping wing, a robotic control problem, and a thermal control problem.

Acknowledgements

I am thankful to be thankful. There are many to whom I would like to express gratitude. First and foremost, I am thankful to God for keeping me from going in a wrong direction and disciplining me because He loves me so much. He has been my strength and protection. He also has provided me with many good people who are essential for my life to flourish.

Professor Walter Murray is a wonderful person to have as my adviser. I am very fortunate to have him not only because he is a distinguished scholar in numerical optimization, but also because he is exceptionally friendly and supportive. Whenever I have come to see him in his home, I have been treated as a family member.

Professor Michael Saunders is a great scholar in numerical linear algebra. He has been extremely helpful. Whenever I had a question on numerical linear algebra, he did not hesitate to help me. Plus, if it were not for his revisions, this thesis would not have been easy to read.

Professor Charbel Farhat is a passionate, intelligent, and distinguished scholar. I am grateful that he allows me to work with other colleagues in his group and to use his cutting-edge code, AERO-S. I am also looking forward to working with him in my postdoctoral career.

Professor Marco Pavone is an excellent scholar in control theory. He introduced me to the subject of control theory and made me realize how powerful it is. I am looking forward to working with him more in the future.

Without Dr. Philip Avery, it would have been impossible to generate the results I needed for this thesis. He has been an essential help for my research. He taught me C++ and all the necessary parts in AERO-S that I needed to know in order to implement a PDE-constrained optimization routine. I am in debt to him in many ways.

Dr. Evelin Sullivan has corrected this thesis. She has transformed my thesis from a manuscript that was full of grammatical errors to a marvelously readable manuscript. I am sure my writing skills have tremendously improved (although they are far from perfect yet) because of the time we spent together to go over my thesis.

Artist Helen Rose Baker is a good friend of mine whom I met when I came to America in 2001 for the first time. She became my friend when I did not have friends in the United States, and she has been my friend and my mentor. She has even provided food when I was hungry.

I am thankful to the following friends and colleagues for their friendship: Leonard Larry Majuri, Rosaline Siu, Paula Akiya, Kelly Thomas, Robin Zuhlke, Jaehyoung Yoo; friends in ICME, Korean Christian Fellowship at Stanford, New Vision Church, Cornell University, Montgomery County Community College, Ambler Mennonite Church, and SaRang Community Church.

My parents have been endless supporters since I was born. They have filled my life with encouraging words. My dad is my role model who teaches me the value of truth, justice, and honesty. My mom provides limitless love, sacrifice, and patience. I learned all of these qualities from her.

My dear wife Giyoung Park has been continuously supportive no matter how hard circumstances around her have been. She has provided mental as well as physical help with her sacrificial love. Her presence gave love, peace, and comfort to me. Moreover, she has given me two lovely daughters, Yejin and Yein, and raised them with care and love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Because of advances in computer technology and numerical methods, it is now possible to simulate complicated physical models, predict their response, and use the results for further design. Many physical laws are expressed in partial differential equations (PDEs). In thermodynamics, for example, Newton's law of cooling states that *the rate of heat loss of a body is proportional to the temperature difference between the body and its surroundings.* Fourier's law, which explains heat conduction, is also expressed as a PDE. In electromagnetism, Maxwell's equations consist of a set of PDEs. Laws of chemistry, biological laws, geophysical laws, and law of conservations can also be expressed as PDEs. In addition, PDEs can be used as mathematical models for a financial market. For example, the Black-Scholes equation describes prices in the options market.

In spite of these broad applications of PDEs, if one wants to use them in a more practical way, it is often necessary to use a mathematical optimization technique in which the PDE is a constraint. PDE-constrained optimization tries to optimize some functional quantity of a system that is governed by a PDE. Many applications of PDE-constrained optimization exist. Three representative examples are inverse problems, shape optimization, and optimal controls. In inverse problems, one tries to find properties of a physical system based on a given measurement. For example, the Mayo Clinic is interested in finding properties of human arteries by examining the response of arteries to ultrasound excitation and using those properties to diagnose arterial disease. This can be done by solving a PDE-constrained optimization problem. In shape optimization, one tries to optimize some functional quantity (e.g., drag or lift of a wing) by varying shape of an object (e.g., a wing or a car). In optimal controls, one tries to find an optimal control of a system in order to achieve a goal. A goal may be to match a target state or to optimize a physical quantity.

As a specific instance of PDE-constrained optimization, we focus on the PDE-constrained optimal control problem. Optimal control is a powerful tool that can be used in a vast number of ways. In engineering problems, for instance, one may want to control a system. For example, one may want to control a car so that it reaches a destination either as fast as possible or with minimum

fuel consumption. Another famous example is the Space-Shuttle trajectory control. The space mission is limited by fuel consumption. Thus, finding a minimum-fuel trajectory and corresponding control for a spacecraft is important. Other examples are found in chemical process industries, such as hydrocarbon fuels, chemical products, pulp and paper products, agrochemicals, and man-made fibers. Controlling the process can optimize profits. Optimal control can also be used in traffic control. One may want to control the traffic signals in a city in order to minimize the duration of rush hour given a fixed amount of traffic. To summarize, applications of optimal control are varied and important.

Optimal control problems with ordinary differential equations and constraints have a much longer history than the multidimensional one. There are two ways of solving such optimal control problems. The first one is an indirect method, in which the calculus of variations is used to form the first-order optimality condition. One ends up solving a Hamiltonian dynamical system, which is a two-point boundary value problem. A software package that implements the indirect method is BNDSCO [62]. However, the Hamiltonian system is often hard to solve. Advances in numerical optimization over the past two decades have resulted in a direct method being the dominant approach. In direct methods, states and controls as well as a cost functional are discretized with some approximated functions (e.g., polynomials in the finite element method). Then the coefficients of approximated functions are treated as optimization variables. The resultant problem becomes a DE-constrained optimization (i.e., Differential Equation-constrained optimization). Direct methods are so popular that a great deal of software has been developed. Programs written in Fortran are ASTOS [36], DIRCOL [85], DITAN [84], OTIS [44], and SOCS [8]. Programs written in MATLAB include DIDO [72], GPOPS [70], PROPT [73], and RIOTS [77]. Except for SOCS and RIOTS, all the software mentioned above uses SNOPT [38] as a nonlinear programming (NLP) solver. PROPT is a MATLAB optimal control routine that utilizes other NLP solvers, such as KNITRO, CONOPT [20], and CPLEX. SOCS was developed by the Boeing company, which developed its own NLP solver.

In spite of the abundance of optimal control software that can solve a DE-constrained optimization, further development in both numerical algorithms and software is required in order to perform optimal control on more complicated physical systems, such as Navier-Stokes equations for fluid flow and nonlinear elasticity for structure and solid mechanics. Sophisticated PDE solvers do exist, and they can be used in developing a numerical method for PDE-constrained optimization. This thesis focuses on developing and implementing an optimal control routine in the cutting-edge PDE solver AERO-S [24]. AERO-S was first developed at the University of Colorado and is now maintained and being further developed by the Farhat Research Group at Stanford University. AERO-S uses the finite element method. The PDEs that AERO-S can solve include linear and nonlinear elasticity equations, the Helmholtz equation, and the Laplace equation. AERO-S consists of many numerical algorithms such as FETI (space-domain parallel decomposition algorithm), GNAT (nonlinear ROM solver), and PITA (time-domain parallel decomposition algorithm). Many linear solvers, such as

MUMPS and SPOOLES, are available in AERO-S.

The simultaneous analysis and design (SAND) approach in solving a PDE-constrained optimization problem can be adapted to a sophisticated numerical method for a PDE. The SAND method is mainly studied in this thesis and implemented in AERO-S in order to take advantage of many of the numerical methods available in AERO-S.

An efficient preconditioner is necessary for a SAND method to be successful. Many powerful preconditioners have been developed in the literature. However, there has been no preconditioner that improves as the value of the regularization parameter becomes smaller. This is undesirable in some applications. For example, if one wants to control a robot that plays a violin, one needs precise control of the robot's arm movement. Another example is a robot operating on a human patient. The precision of the robot's motion directly affects the well-being of the patient. In these applications, the smaller the value of a regularization parameter, the better the performance. In this thesis, several novel preconditioners that work better for a smaller regularization parameter are proposed and developed.

1.1 Thesis accomplishments and outline

The major contributions of this thesis are follows:

- Implemented PDE-constrained optimal control routines in AERO-S with four types of PDE constraints.
- Integrated the generalized α method in SAND sequentially for a nonlinear dynamic PDE-constrained optimization routine in AERO-S.
- Developed preconditioners that work well for a small regularization parameter.
- Introduced a "useful" exact representation of the Schur complement for a special type of KKT systems and used FETI-DP in order to solve them.

This thesis considers PDEs that reflect thermal and structural systems. Chapter 2 describes and derives four types among those partial differential equations (i.e., linear static, nonlinear static, linear dynamic, and nonlinear dynamic PDEs). For the dynamic problems, time integrators are explained. Chapter 3 formulates PDE-constrained optimal control problems in which the four types of PDEs described in Chapter 2 are used as constraints. Chapter 4 briefly goes over methods for PDE-constrained optimization (the nested analysis and design (NAND) method and the simultaneous analysis and design (SAND) method). Chapter 5 presents a Krylov iterative method (GMRES) that is used in SAND and introduces a set of preconditioners that work well for a small regularization parameter. Chapter 5 also describes an exact representation of the Schur complement for a special type of KKT systems, which can be used as either a preconditioner in an iterative method or a

solver in the range-space method. Chapter 6 explains the linesearch method used to achieve global convergence of sequential quadratic programming methods and several merit functions implemented in the optimal control routine. Chapter 7 presents numerical results. The numerical results include square heat plate static and dynamic controls, large deflection of a plate, balance of an inverted pendulum, control of five-link rigid and flexible biped robots, and control of a flapping wing. Chapter 8 provides a summary of this thesis and suggests directions for future research. Finally, spectral analysis of the preconditioned systems with the preconditioners introduced in Chapter 5 is given in Appendix A.

Chapter 2

Partial differential equations

Partial differential equations (PDEs) can be applied to many areas. Elasticity, plasticity, quantum mechanics, fluid flow, electrodynamics, acoustics, and heat transfer are a few examples. Solving PDEs means that one finds states (e.g., temperature, displacement, or velocity), given some boundary conditions and internal and external loads (e.g., applied body or surface forces, gravity, or heat source). There are four types of PDE:

- Linear Static
- Nonlinear Static
- Linear Dynamic
- Nonlinear Dynamic

Several numerical methods for solving these PDEs exist, among them the finite element (FE), the finite volume (FV), the finite difference (FD), and the spectral method. The FE method is frequently used in computational heat transfer, structural, and solid mechanics, in which a Lagrangian mesh is convenient, while the FV and FD methods are popular in computational fluid analysis, in which an Eulerian mesh is often adapted. In this thesis, only thermal and structural applications are considered, so the finite element method is its focus. However, the numerical optimization methods introduced for PDE-constrained optimization problems not are limited to the FE method. They can be extended to other numerical PDE solving methods (e.g., FV or FD methods).

Two PDE formulations of solid mechanics are possible with the Lagrangian mesh: the total Lagrangian formulation and the updated Lagrangian formulation. In the total Lagrangian formulation, derivatives and integrals are taken with respect to the Lagrangian coordinates (fixed coordinates), while the updated Lagrangian uses Eulerian coordinates (updated coordinates). The total Lagrangian formulation uses fixed coordinates (e.g., initial configuration), whereas the updated Lagrangian formulation uses some reference coordinate, which is updated regularly. If the reference

coordinate is updated so that it coincides with the current configuration, this is called the Eulerian formulation. Since the updated Lagrangian needs to update the coordinate, it seems to require more work than the total Lagrangian. However, one may prefer to use the updated Lagrangian because it can handle rotational degrees of freedom more easily than the total Lagrangian, in which some difficulties may occur when the angle of rotation reaches 360 degrees. For this reason, one may use a mixed formulation, in which translational degrees of freedom are treated in the total Lagrangian formulation and rotational degrees of freedom are treated in an updated Lagrangian or Eulerian. In this thesis, whenever rotational degrees of freedom are present (e.g., in the case of beam and shell elements), the mixed formulation is adopted.

Dynamic PDEs, in general, require a time integrator, which makes the solution proceed from the initial to the final time step. There are two kinds of time integrators: explicit and implicit. The explicit time integrator uses only previous states in the PDE residual, so the update of the current state is explicit and computationally cheap. On the other hand, in the implicit time integrator, the update formula involves both current and previous states, so one needs to solve a system of equations in order to proceed in time. Thus, it is computationally more expensive than the explicit time integrator. However, in order to get stable and accurate enough solutions, an explicit scheme requires much smaller time steps (and hence many more than an implicit one). An example of an explicit scheme is the forward Euler method. Examples of implicit time integrators include the backward Euler method, the midpoint rule, the generalized α method, and the composite rule. In the numerical optimization methods developed in this thesis (i.e., the full SAND), all or some of the temporal state variables are used as optimization variables. Thus it is important to have as small a number of time steps as possible in order to prevent storage from being a problem. This is why only implicit time integrators are used in this thesis.

For linear dynamic PDEs, the midpoint rule is sufficient to obtain stable, accurate solutions. However, in nonlinear dynamic PDEs, the midpoint rule may result in an unstable solution. Thus, more sophisticated time integrators need to be applied, such as generalized-α or composite rules (e.g., the TRBDF2 method). In this thesis, only the generalized-α method is considered for nonlinear dynamic PDE-constrained optimization. However, there is no reason the composite time integrators would not work in optimization. The generalized-α method is explained in Sections 2.3 and 2.4 for linear and nonlinear PDEs.

2.1 Linear static PDE

Linear static PDE may be applied to some problems in heat conduction. It also seems to solve problems in the theory of linear elasticity in solid mechanics if small deformation is assumed as well as many other problems. At its core, a system of linear equations needs to be solved in order to

obtain the solution of discretized linear PDE, that is,

$$Ax = b.$$

In this section, a linear system of equations is derived to obtain a linear static heat equation. Specifically, the finite element formulation is derived. Since many interesting and practical examples of structural analysis are nonlinear, the formulation of a structural PDE is postponed until Section 2.2 where both linear and nonlinear formulations are presented. A more detailed derivation of an FE formulation is found in the textbook by Hughes [48] or by Quarteroni [69]. The finite element formulation can be derived through two consecutive procedures: first, converting a strong form of the PDE to a variational form; second, approximating the variational form with finite element basis functions.

2.1.1 Linear heat equation

The linear static heat equation is expressed as an elliptic PDE. Let Υ be a solution space. The strong form of the PDE is defined to find temperature $y \in \Upsilon$ in a domain Ω for a given heat source u:

$$-\nabla \cdot (\kappa \nabla y) = u \text{ on } \Omega, \tag{2.1}$$

subject to $y = g$ on Γ_g and $n \cdot \nabla y = h$ on Γ_h, where $\Gamma_g \subset \bar{\Omega}$ and $\Gamma_h \subset \bar{\Omega}$ with Laplacian operator ∇^2; κ is the conductivity matrix. The variational form (i.e., weak form) is obtained by introducing a weighting function $w \in H^1$ that vanishes on Γ_g. H^1 denotes a Sobolev space, in which the first derivative of a function is square integrable:

$$\int_\Omega \|\nabla w\|^2 dx < \infty. \tag{2.2}$$

It is convenient to define a function space that w belongs to. For example, $w \in Ł = \{v(x) | v(x) \in H^1, v(x) = 0 \text{ for } x \in \Gamma_g\}$. Due to (2.1), and setting the conductivity matrix to identity, it is true that

$$0 = -\int_\Omega w(\nabla^2 y + u) dx. \tag{2.3}$$

Because w vanishes on Γ_g, integration by parts and the divergence theorem leads to a weak form,

$$0 = \int_\Omega (\nabla w \cdot \nabla y) dx - \int_\Omega (wu) dx - \int_{\Gamma_h} (wh) dx. \tag{2.4}$$

The variable y is a solution to the weak form (2.4) if the weak form is satisfied for any $w \in Ł$. One can prove that a solution to the weak form is a solution to the strong form and vice versa [48]. One thing to note here is that both the test function w and the trial function y do not need to be

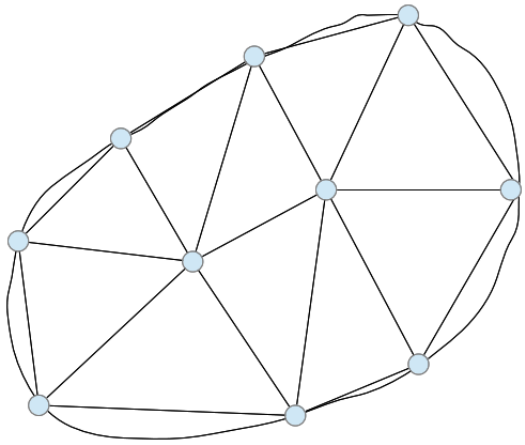


Figure 2.1: The figure depicts the domain Ω and triangular discretization with nodes at each vertex.

in H^1 in order to have an integral in the weak form. They only need to be continuous (e.g., C^0). To get the strong form one can relax the condition on w and y of being in C^0 by imposing the additional continuous condition $[\![\nabla y]\!] = 0$, where $[\![\cdot]\!]$ denotes the jump function. The strong form then becomes

$$\begin{aligned} -\nabla \cdot (\kappa \nabla y) &= u \text{ on } \Omega, \\ [\![\nabla y]\!] &= 0 \text{ on } \Gamma_i, \end{aligned} \tag{2.5}$$

subject to $y = g$ on Γ_g and $n \cdot \nabla y = h$ on Γ_h, where Γ_i are the points of discontinuity. In order to actually solve heat equation (2.1) numerically, the geometric domain needs to be discretized by introducing nodes. The set of nodes is introduced in a conforming way, as depicted in Figure 2.1. The finite element method replaces y and w with finite dimensional approximate functions y^h and w^h. It is convenient to decompose y^h into the homogeneous part $v^h \in Ł^h$ and the Dirichlet part, g^h (i.e. $y^h = v^h + g^h$). $Ł^h$ is a finite dimensional function space that corresponds to the space Ł. That is,

$$Ł^h = \{v^h(x) | v^h(x) \in H^1, v^h(x_i) = 0 \text{ for any node } x_i \in \Gamma_g\}. \tag{2.6}$$

Note that y_h satisfies the Dirichlet boundary conditions by construction. It is possible to express w^h, $y^h \in Ł^h$, and g^h as a linear combination of some basis functions called shape functions.

$$\begin{aligned} w^h(x) &= \sum_{i \in \eta - \eta_g} w_i N_i(x), \\ v^h(x) &= \sum_{j \in \eta - \eta_g} v_j N_j(x), \\ g^h(x) &= \sum_{j \in \eta_g} g_j N_j(x), \end{aligned} \tag{2.7}$$

where N_i is the shape function associated with node i; w_i and v_j are some coefficients, η is a set of all the nodes, and η_g is a set of nodes in Γ_g; w^h needs to satisfy the Dirichlet boundary condition exactly on the nodes that belong to Γ_g. Each y_j needs to be determined by satisfying the following finite dimensional weak form for any $w^h \in \mathrm{L}^h$:

$$0 = \int_\Omega (\nabla w^h \cdot \nabla (v^h + g^h)) dx - \int_\Omega (w^h u) dx - \int_{\Gamma_h} (w^h h) dx. \tag{2.8}$$

Plugging the w^h, v^h, and g^h expressions into (2.7), the finite dimensional weak form (2.8) becomes

$$\begin{aligned} 0 = \sum_{i \in \eta - \eta_g} w_i [& \sum_{j \in \eta - \eta_g} v_j \int_\Omega \nabla N_i(x) \cdot \nabla N_j(x) dx \\ & + \sum_{k \in \eta_g} g_k \int_\Omega \nabla N_i(x) \cdot \nabla N_k(x) dx \\ & - \int_\Omega N_i(x) u dx - \int_{\Gamma_h} N_i(x) h dx]. \end{aligned} \tag{2.9}$$

This needs to be satisfied for any w_i. Equivalently, the parts in the square brackets need to sum up to zero. That is,

$$\sum_{j \in \eta - \eta_g} v_j \int_\Omega \nabla N_i(x) \cdot \nabla N_j(x) dx = - \sum_{k \in \eta_g} g_k \int_\Omega \nabla N_i(x) \cdot \nabla N_k(x) dx + \int_\Omega N_i(x) u dx + \int_{\Gamma_h} N_i(x) h dx. \tag{2.10}$$

for each i. Now, it is obvious that the finite element formulation for linear heat conduction ends up with a system of linear equations, $Kv^h = b$, where $K_{ij} = \int_\Omega \nabla N_i(x) \cdot \nabla N_j(x) dx$ and $b_i = -\sum_{k \in \eta_g} g_k \int_\Omega \nabla N_i(x) \cdot \nabla N_k(x) dx + \int_\Omega N_i(x) u dx + \int_{\Gamma_h} N_i(x) h dx$. K is called the stiffness matrix in the finite element method. Although the functions u and h are known in the PDE solver, if they are discretized with the same shape functions $N_j(x)$ (i.e. $u = \sum_{j \in \eta - \eta_g} u_j N_j(x)$ and $h = \sum_{k \in \eta_h} h_k N_k(x)$), then b_i can be decomposed into

$$\begin{aligned} b_i &= - \sum_{k \in \eta_g} g_k \int_\Omega \nabla N_i(x) \cdot \nabla N_k(x) dx + \int_\Omega N_i(x) u dx + \int_{\Gamma_h} N_i(x) h dx, \\ &= - \sum_{k \in \eta_g} g_k \int_\Omega \nabla N_i(x) \cdot \nabla N_k(x) dx + \sum_{j \in \eta - \eta_g} u_j \int_\Omega N_i(x) N_j(x) dx + \sum_{k \in \eta_h} h_k \int_{\Gamma_h} N_i(x) N_k(x) dx, \\ &= V u^h - K_c y_c^h, \end{aligned} \tag{2.11}$$

where $V_{ij} = \int_\Omega N_i(x) N_j(x)$ is a volume matrix and an element of K_c is determined by

$$(K_c)_{ik} = \begin{cases} \int_\Omega \nabla N_i(x) \cdot \nabla N_k(x) dx & \text{if } k \in \eta_g \\ \int_\Omega N_i(x) N_k(x) dx & \text{if } k \in \eta_h \end{cases}, \tag{2.12}$$

and similarly an element of y_c^h is determined by

$$(y_c^h)_k = \begin{cases} g_k & \text{if } k \in \eta_g \\ h_k & \text{if } k \in \eta_h \end{cases} . \tag{2.13}$$

Finally, the discretized linear static heat PDE becomes $Kv^h = Vu^h - K_c y_c^h$. When implementing finite element code, one can also consider a nodal heat source instead of the distributed heat source u_h. This can be accomplished by setting the nodal heat source variable $f^h = Vu^h$. Then the system of equations for linear static PDE becomes $Kv^h = f^h - K_c y_c^h$.

2.2 Static structural PDE

Both linear and nonlinear static structural problems can be explained within continuum mechanics. An important partial differential equation one needs to deal with in continuum mechanics is the equilibrium equation. It is convenient to express the PDE in indicial notation (or Einstein notation) due to the presence of fourth-order tensors in constitutive laws and the partial derivatives of second-order tensors. In indicial notation, y_i represents partial derivative with respect to the i-th component of the coordinate. If the index is repeated twice, then this indicates summation on that index. For example, $y_i w_i$ means $\sum_i y_i w_i$ and $\sigma_{ij,j}$ means $\sum_j \sigma_{ij,j}$, etc. In indicial notation, a strong form in continuum mechanics consists of an equilibrium equation,

$$\sigma_{ij,j} + b_i = 0, \tag{2.14}$$

subject to displacement boundary condition $y_i = d_i$ on Γ_{d_i} and traction boundary condition $n_j \sigma_{ij} = t_i$ on Γ_{t_i}, where σ_{ij} is the (i,j) component of the stress tensor. The $\sigma_{ij,j}$ term is a partial derivative with respect to the j-th component of the coordinate, and b_i is i-th component of the body force. As in the linear heat equation (2.1.1), the weak form of (2.14) can be obtained by multiplying 2.1.1 by the test function w and integrating over the domain, that is,

$$\int_\Omega w_i(\sigma_{ij,j} + b_i)dx. \tag{2.15}$$

Integration by parts and the divergence theorem leads to the weak form,

$$\int_\Omega w_{i,j}\sigma_{ij}dx - \int_\Omega w_i b_i dx - \sum_{j=1}^{n_{sd}} \int_{\Gamma_{t_j}} t_i w_i dx = 0, \tag{2.16}$$

where n_{sd} is the number of space dimensions (i.e., $n_{sd} = 2$ for 2D and 3 for 3D). The boundary integrals over Γ_{d_i}s vanish because test function w is zero on the displacement boundary. In order to make sense out of the integrals in weak form (2.16), σ_{ij} needs to be C^{-1} and $w \in C^0$, where C^{-1}

denotes a set of piecewise continuous functions. For example, if $f \in C^{-1}$ in 1D, then

$$\int_a^b f_{,x}(x)dx = f(b) - f(a) + \sum_i [\![f(x_i)]\!].$$

If $f \in C^{-1}$ is in 2D or in 3D, then the discontinuity is assumed to occur along a line or surface, respectively. If $\sigma_{ij} \in C^0$ and $C^{-1} \in C^0$, then the strong form corresponding to the weak form (2.16) is identical to the strong form (2.14) with an extra boundary condition, that is, the traction continuity condition,

$$[\![n_j \sigma_{ij}]\!] = 0 \text{ on } \Gamma_{int}, \tag{2.17}$$

where Γ_{int} is the union of all surfaces (or lines in 2D) on which the stresses are discontinuous in the domain. As for the linear heat equation (2.1.1), the finite element method uses spatial discretization of the domain by introducing nodes and elements. Then test function w and displacement y are approximated as linear combinations of basis functions called shape functions, introducing finite variables w^h and y^h. It is convenient to decompose displacement vector y^h into displacement boundary parts d^h and non-boundary parts v^h (i.e. $y^h = v^h + d^h$). Then w^h, v^h, and d^h are expressed as

$$\begin{aligned} w^h_{iI}(x) &= w_{iI} N_I(x), \\ v^h_{iI}(x) &= v_{iI} N_I(x), \\ d^h_{iI}(x) &= d_{iI} N_I(x), \end{aligned} \tag{2.18}$$

where index i is the degree of freedom number and I the global node number. The global node number I for w and v belongs to $\eta - \eta_{d_i}$, where η is a set of all the nodes and η_{d_i} a set of nodes in Γ_{d_i}. The global node number I for d is restricted to η_{d_i}. Substituting (2.18) into the weak form (2.16), the discretized weak form becomes

$$w^h_{iI} \left(\int_\Omega N_{I,j} \sigma_{ij}(y^h) dx - \int_\Omega N_I b_i dx - \sum_{j=1}^{n_{sd}} \int_{\Gamma_{t_j}} t_i N_I dx \right) = 0. \tag{2.19}$$

The discretized weak form needs to be satisfied for any w^h_{iI}, and ends up being

$$\int_\Omega N_{I,j} \sigma_{ij}(y^h) dx - \int_\Omega N_I b_i dx - \sum_{j=1}^{n_{sd}} \int_{\Gamma_{t_j}} t_i N_I dx = 0, \tag{2.20}$$

for any $I \in \eta - \eta_{d_i}$. The weak form in (2.19) can be thought of as a virtual work where w^h_{iI} is virtual displacement and rest of the terms are force. Thus, internal force f_{int} and external force f_{ext} can

be defined as

$$\begin{aligned} f_{iI}^{int}(v^h,d^h) &= \int_\Omega N_{I,j}\sigma_{ij}(y^h)dx, \\ f_{iI}^{ext} &= \int_\Omega N_I b_i dx + \sum_{j=1}^{n_{sd}} \int_{\Gamma_{t_j}} t_i N_I dx. \end{aligned} \tag{2.21}$$

Then the discretized equilibrium for static continuum mechanics reduces to $f_{int}(y^h) = f_{ext}$. The external force f_{ext} consists of two parts, namely, the body force part f_{body} and the traction boundary parts $f_{traction}$. The body force part f_{body} can be expressed as $f_{body} = Vb^h$, where $V_{ij} = \int_\Omega N_i(x)N_j(x)$ is a volume matrix and b^h is a finite element approximation of b, that is, $b_{iI}^h(x) = b_{iI}N_I(x)$. The internal force can be divided into two parts since $y^h = v^h + d^h$. The notation $f_{int}(v^h,d^h)$ is used to emphasize that internal force depends on discretized displacement v^h and d^h. Whether the PDE (2.14) is linear or nonlinear is determined by a strain-displacement relation and a constitutive law that defines the relation between stress tensor and deformation gradient (or strain tensor). Let x denote a point in the current configuration and X a corresponding point in the initial configuration. The deformation gradient is then defined to be $F_{ij} = \frac{\partial x_i}{\partial X_j}$. Note that the current and initial points are related to displacement as $x_i = X_i + y_i$. Thus the deformation gradient is directly related to the displacement gradient as follows:

$$\begin{aligned} F_{ij} &= \frac{\partial X_i}{\partial X_j} + \frac{\partial y_i}{\partial X_j}, \\ &= \delta_{ij} + \frac{\partial y_i}{\partial X_j}, \end{aligned} \tag{2.22}$$

where δ_{ij} is the Kronecker delta. There are two kinds of strain tensor: infinitesimal strain (or the Cauchy strain tensor) for small deformation and finite strain (or a large strain tensor) for large deformation. The Cauchy strain tensor is defined as

$$\epsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}), \tag{2.23}$$

which can be rewritten in terms of deformation gradient as $\epsilon = \frac{1}{2}(F^T + F) + I$ due to relation (2.22). One of the large strain tensors, the Green-Lagrangian strain tensor E, is defined as

$$E = \frac{1}{2}(F^T F - I), \tag{2.24}$$

which can be rewritten in terms of displacement $E_{ij} = \frac{1}{2}(y_{i,j} + y_{j,i} + y_{k,i}y_{k,j})$ due to the relation (2.22); $y_{k,i}y_{k,j}$ represents summation on index k. Likewise, there are stress measures for infinitesimal and large deformation. For infinitesimal deformation, the Cauchy stress ϵ defines three orthogonal normal stresses and six orthogonal shear stresses relative to the current configuration, which can be expresses as a second-order tensor. On the other hand, one of the stress measures for large deformation, the second Piola-Kirchhoff stress tensor S, relates forces in the reference configuration

to area in the reference configuration, which is expressed as

$$S = JF^{-1} \cdot \sigma \cdot F^{-T}, \tag{2.25}$$

where $J = \det F$ is the determinant of the deformation gradient. In general, stress tensors σ or S are expressed as a function of the deformation gradient F and F's rates in a constitutive law, that is,

$$\sigma(t) = f(F(\bar{t}), \dot{F}(\bar{t}), \text{ etc., } \bar{t} \leqslant t), \tag{2.26}$$

where the stress tensor may depend on the history of deformation in time. For example, the generalized Hooke's law defines the linear constitutive law between Cauchy stress σ and Cauchy strain tensor ϵ,

$$\sigma_{ij} = c_{ijkl}\epsilon_{kl}, \tag{2.27}$$

where c_{ijkl} are some elastic coefficients. Therefore, Hooke's law is only valid for small deformation. For large deformation, the simplest hyper-elastic material model, the Saint Venant-Kirchhoff model relates the second Piola-Kirchhoff (PK2) stress tensor S to the Green-Lagrangian strain tensor E as

$$S = \lambda \operatorname{tr}(E) I + 2\mu E, \tag{2.28}$$

where λ and μ are lame constants and I the identity matrix. Thus, internal force f_{int} is linear in displacement if the generalized Hooke's law is used and is nonlinear if, for example, the Saint Venant-Kirchhoff model is used. In conclusion, the discretized equilibrium equation of a static structural PDE can be expressed as

$$f_{int}(v^h, d^h) = Vb^h + f_{traction}. \tag{2.29}$$

For more detailed formulation of linear elasticity finite element models, the textbook by [48] is recommended. For nonlinear formulation, the textbook by Belytschko, Liu, and Moran [6] is an excellent source.

2.3 Linear dynamic PDE

In this section, linear thermal and structural dynamic formulation is considered. The midpoint rule is an unconditionally stable implicit time integrator for a linear dynamic problem. Thus, it is enough to consider only midpoint rules for both thermal and structural linear dynamic problems.

2.3.1 Linear dynamic heat equation

The linear dynamic heat equation is expressed as a parabolic PDE,

$$\frac{\partial d}{\partial t} - \nabla^2 d = F(t), \tag{2.30}$$

where d is temperature and $F(t)$ internal heat source. The corresponding finite element discretization is

$$Mv + Kd = F, \tag{2.31}$$

where M and K are the mass and stiffness matrix, respectively, and d and F are temperature and applied heat source, respectively; v is the first temporal derivative, that is, $v = \dot{d}$. The midpoint rule for the linear heat transfer starts by setting the initial states,

$$\begin{aligned} d_0 &= d, \\ v_0 &= M^{-1}(F(0) - Kd), \end{aligned} \tag{2.32}$$

where $F(0)$ is the initial external heat source. The midpoint rule for the heat transfer problem satisfies the following equilibrium equation at midpoint:

$$F(t_{n+\frac{1}{2}}) = Mv_{n+\frac{1}{2}} + Kd_{n+\frac{1}{2}}. \tag{2.33}$$

The update rule for the midpoint rule is,

$$d_{n+1} = d_n + \frac{\Delta t}{2}(v_n + v_{n+1}). \tag{2.34}$$

The midpoint states are linearly interpolated between two end points as

$$\begin{aligned} d_{n+\frac{1}{2}} &= \frac{1}{2}(d_{n+1} + d_n), \\ v_{n+\frac{1}{2}} &= \frac{1}{2}(v_{n+1} + v_n), \\ t_{n+\frac{1}{2}} &= \frac{1}{2}(t_{n+1} + t_n), \end{aligned} \tag{2.35}$$

in which d_n and v_n are displacement and velocity at time t_n, respectively; $n = 0,1,\ldots,N-1$ where N is the number of time steps and Δt is the time step. Using (2.34) and (2.35), the first temporal derivate midpoint can be expressed in terms of midpoint and current temperature,

$$v_{n+\frac{1}{2}} = \frac{2}{\Delta t}(d_{n+\frac{1}{2}} + d_n). \tag{2.36}$$

Substituting these into (2.33) leads to

$$F(t_{n+\frac{1}{2}}) = \frac{2}{\Delta t} M(d_{n+\frac{1}{2}} - d_n) + K d_{n+\frac{1}{2}}. \tag{2.37}$$

Combining $d_{n+\frac{1}{2}}$ terms in one side and multiplying $\frac{\Delta t}{2}$ both sides give

$$\bar{K} d_{n+\frac{1}{2}} = \frac{\Delta t}{2} F(t_{n+\frac{1}{2}}) + M d_n, \tag{2.38}$$

where $\bar{K} = M + \frac{\Delta t}{2} K$. Once $d_{n+\frac{1}{2}}$ is obtained by solving (2.38), d_{n+1} and v_{n+1} are updated to

$$\begin{aligned} d_{n+1} &= 2d_{n+\frac{1}{2}} - d_n \\ v_{n+1} &= \frac{1}{\Delta t}(d_{n+1} - d_n). \end{aligned} \tag{2.39}$$

It is possible to obtain a compact form, in which no intermediate states are shown. For example,

$$\begin{bmatrix} \bar{K} & 0 \\ 0 & \bar{K} \end{bmatrix} \begin{pmatrix} d_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} \Delta t F_{n+\frac{1}{2}} \\ 2F_{n+\frac{1}{2}} \end{pmatrix} + \begin{bmatrix} M - \frac{\Delta t}{2} K & 0 \\ -2K & -\bar{K} \end{bmatrix} \begin{pmatrix} d_n \\ v_n \end{pmatrix}. \tag{2.40}$$

This compact form will be used in the linear heat transfer dynamic optimal control problem. Setting external force terms in the compact form (2.40), to zero, it can be written as $X_{n+1} = A_h X_n$, where the amplification matrix A_h is defined as

$$A_h = \begin{bmatrix} \bar{K} & 0 \\ 0 & \frac{1}{\Delta t}\bar{K} \end{bmatrix}^{-1} \begin{bmatrix} M - \frac{\Delta t}{2} K & 0 \\ -2K & -\frac{1}{\Delta t}\bar{K} \end{bmatrix}, \tag{2.41}$$

and $X_n = (d_n, \Delta t v_n)^T$. The amplification matrix A_h is used to study the accuracy of the time integrator. The stability and numerical dissipation depends on the eigenvalues of A_h. Thus it is important to study the behavior of A_h.

2.3.2 Linear structural dynamic PDE

The discretized equilibrium equation for linear structural dynamic is

$$Ma + Cv + Kd = F, \tag{2.42}$$

where M, C, and K are mass, damping, and stiffness matrix, respectively; d, v, a, and F are displacement, velocity, acceleration, and external force, respectively. The midpoint rule for a linear

structural dynamic problem starts with initial states,

$$\begin{aligned} d_0 &= d \\ v_0 &= v, \end{aligned} \tag{2.43}$$

where $F(0)$ is an external force at initial time and d and v are given initial conditions. The midpoint rule satisfies the following equilibrium equation at midpoints:

$$Ma_{n+\frac{1}{2}} + Cv_{n+\frac{1}{2}} + Kd_{n+\frac{1}{2}} = F(t_{n+\frac{1}{2}}), \tag{2.44}$$

$a_{n+\frac{1}{2}}$, $a_{n+\frac{1}{2}}$, and $d_{n+\frac{1}{2}}$ are the acceleration, velocity, and displacement at the midpoint. The midpoint rule determines those midpoint states (i.e. $a_{n+\frac{1}{2}}$, $v_{n+\frac{1}{2}}$, and $d_{n+\frac{1}{2}}$) by linear interpolation between two end states. For example,

$$\begin{aligned} d_{n+\frac{1}{2}} &= \frac{1}{2}(d_{n+1} + d_n) \\ v_{n+\frac{1}{2}} &= \frac{1}{2}(v_{n+1} + v_n) \\ a_{n+\frac{1}{2}} &= \frac{1}{2}(a_{n+1} + a_n) \\ t_{n+\frac{1}{2}} &= \frac{1}{2}(t_{n+1} + t_n). \end{aligned} \tag{2.45}$$

The update formula from t_n to t_{n+1} is identical to the Newmark method, that is,

$$\begin{aligned} d_{n+1} &= d_n + \Delta t v_n + \frac{\Delta t^2}{4}(a_n + a_{n+1}) \\ v_{n+1} &= v_n + \frac{\Delta t}{2}(a_n + a_{n+1}). \end{aligned} \tag{2.46}$$

Using (2.45) and (2.46), it is possible to express $a_{n+\frac{1}{2}}$ and $v_{n+\frac{1}{2}}$ in terms of only midpoint displacement and previous states (e.g., $d_{n+\frac{1}{2}}$, d_n, v_n, and a_n). For example,

$$\begin{aligned} a_{n+\frac{1}{2}} &= \frac{4}{\Delta t^2}[d_{n+\frac{1}{2}} - d_n] - \frac{2}{\Delta t} v_n \\ v_{n+\frac{1}{2}} &= \frac{2}{\Delta t}[d_{n+\frac{1}{2}} - d_n]. \end{aligned} \tag{2.47}$$

Substituting these expressions for the equilibrium (2.44), the equilibrium equation is expressed without intermediate velocities and accelerations and only in terms of intermediate displacement and previous states.

$$M\left(\frac{4}{\Delta t^2}[d_{n+\frac{1}{2}} - d_n] - \frac{2}{\Delta t} v_n\right) + C\left(\frac{2}{\Delta t}[d_{n+\frac{1}{2}} - d_n]\right) + Kd_{n+\frac{1}{2}} - F(t_{n+\frac{1}{2}}) = 0. \tag{2.48}$$

Combining $d_{n+\frac{1}{2}}$ terms on one side and multiplying both sides by $\frac{\Delta t^2}{4}$, 2.48 becomes

$$\hat{K} d_{n+\frac{1}{2}} = \tilde{K} d_n + \frac{\Delta t}{2} M v_n + \frac{\Delta t^2}{4} F(t_{n+\frac{1}{2}}), \tag{2.49}$$

where

$$\begin{aligned} \tilde{K} &= M + \frac{\Delta t}{2} C \\ \hat{K} &= \tilde{K} + \frac{\Delta t^2}{4} K. \end{aligned} \tag{2.50}$$

Once $d_{n+\frac{1}{2}}$ is obtained by solving (2.49) then $v_{n+\frac{1}{2}}$ is updated as

$$v_{n+\frac{1}{2}} = \frac{2}{\Delta t}[d_{n+\frac{1}{2}} - d_n], \tag{2.51}$$

and d_{n+1} and v_{n+1} are updated as

$$\begin{aligned} d_{n+1} &= 2 d_{n+\frac{1}{2}} - d_n \\ v_{n+1} &= 2 v_{n+\frac{1}{2}} - v_n, \end{aligned} \tag{2.52}$$

It is possible to obtain a compact form of the midpoint rule for the linear structural dynamic as in the linear thermal dynamic problem. That is,

$$\begin{bmatrix} \hat{K} & 0 \\ 0 & \hat{K} \end{bmatrix} \begin{pmatrix} d_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} \frac{\Delta t^2}{2} F_{n+\frac{1}{2}} \\ \Delta t F_{n+\frac{1}{2}} \end{pmatrix} + \begin{bmatrix} \tilde{K} - \frac{\Delta t^2}{4} K & \Delta t M \\ -\Delta t K & 2M - \hat{K} \end{bmatrix} \begin{pmatrix} d_n \\ v_n \end{pmatrix}. \tag{2.53}$$

This compact form will be used as a linear structure dynamic optimal control formulation below. Setting external force terms in the compact form (2.53) to zero, it can be written as $X_{n+1} = A_s X_n$, where the amplification matrix A_s is defined as

$$A_s = \begin{bmatrix} \hat{K} & 0 \\ 0 & \frac{1}{\Delta t}\hat{K} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{K} - \frac{\Delta t^2}{4} K & M \\ -\Delta t K & \frac{2}{\Delta t} M - \frac{1}{\Delta t}\hat{K} \end{bmatrix}, \tag{2.54}$$

and $X_n = (d_n, \Delta t v_n)^T$. The amplification matrix A_s is used to study the accuracy of the time integrator. Since the stability and numerical dissipation depends on the eigenvalues of A_s, it is important to study the behavior of A_s.

2.4 Nonlinear dynamic PDE

Many interesting simulations are described by "nonlinear dynamic PDEs," which can handle large deformation and transient analysis. Nonlinear dynamic PDEs are the most complicated one among four PDE types simply because they have to deal with both space and time domains as well as

nonlinearity. The main issue here is the time integrator and how it deals with nonlinearity. There are two main types of time integrators: the generalized alpha method and the composite rule (e.g., the TRPDF2). The generalized alpha method is explained here since it is chosen to be used in the PDE-constrained optimization routine. However, the choice was completely arbitrary. In the future, it would be interesting to study how the PDE-constrained optimization routine performs with the TRPDF2. One attractive feature of the generalized alpha method is that the midpoint rule, which is described in the linear dynamic PDE Section 2.3, is a particular case.

The generalized alpha method was first introduced by Chung and Hulbert in their 1993 paper[16]. It starts with computing initial states with given initial displacement, d^0, and velocity, v^0.

$$\begin{aligned} d_0 &= d^0 \\ v_0 &= v^0 \\ a_0 &= M^{-1}[F(0) - Cv - f_{int}(d_0)], \end{aligned} \tag{2.55}$$

where M and C are the mass and damping matrix, respectively. $F(0)$ is external force at initial time, and $f_{int}(d)$ is an internal force corresponding to a displacement d. The generalized alpha method satisfies the following equilibrium equation:

$$Ma_{n+1-\alpha_m} + Cv_{n+1-\alpha_f} + f_{int}(d_{n+1-\alpha_f}) = F(t_{n+1-\alpha_f}), \tag{2.56}$$

where $a_{n+1-\alpha_m}$ is an acceleration at time $t_{n+1-\alpha_m}$ and $v_{n+1-\alpha_f}$ is interpreted similarly. The generalized alpha method determines intermediate states (i.e., $a_{n+1-\alpha_m}$, $v_{n+1-\alpha_f}$, and $d_{n+1-\alpha_f}$) by linear interpolation between two end states. For example,

$$\begin{aligned} d_{n+1-\alpha_f} &= (1-\alpha_f)d_{n+1} + \alpha_f d_n \\ v_{n+1-\alpha_f} &= (1-\alpha_f)v_{n+1} + \alpha_f v_n \\ a_{n+1-\alpha_m} &= (1-\alpha_m)a_{n+1} + \alpha_m a_n \\ t_{n+1-\alpha_f} &= (1-\alpha_f)t_{n+1} + \alpha_f t_n. \end{aligned} \tag{2.57}$$

The update formula from t_n to t_{n+1} is identical to the Newmark method, that is,

$$\begin{aligned} d_{n+1} &= d_n + \Delta t v_n + \Delta t^2 \left[(\frac{1}{2} - \beta)a_n + \beta a_{n+1} \right] \\ v_{n+1} &= v_n + \Delta t[(1-\gamma)a_n + \gamma a_{n+1}]. \end{aligned} \tag{2.58}$$

Using (2.57) and (2.58), it is possible to express $a_{n+1-\alpha_m}$ and $v_{n+1-\alpha_f}$ in terms of only intermediate

displacement and previous states (e.g., $d_{n+1-\alpha_f}$, d_n, v_n, and a_n). For example,

$$\begin{aligned} a_{n+1-\alpha_m} &= \frac{(1-\alpha_m)}{\Delta t^2\beta(1-\alpha_f)}[d_{n+1-\alpha_f}-d_n] - \frac{(1-\alpha_m)}{\Delta t\beta}v_n + \left[\frac{(\alpha_m-1)}{2\beta}+1\right]a_n \\ v_{n+1-\alpha_f} &= \frac{\gamma}{\Delta t\beta}[d_{n+1-\alpha_f}-d_n] + \left[1-\frac{\gamma(1-\alpha_f)}{\beta}\right]v_n + \frac{(1-\alpha_f)\Delta t(2\beta-\gamma)}{2\beta}a_n. \end{aligned} \tag{2.59}$$

Substituting these for the equilibrium (2.56), the equilibrium equation is expressed without intermediate velocities and accelerations and only in terms of intermediate displacement and previous states.

$$\begin{aligned} M\left(\frac{(1-\alpha_m)}{\Delta t^2\beta(1-\alpha_f)}[d_{n+1-\alpha_f}-d_n] - \frac{(1-\alpha_m)}{\Delta t\beta}v_n + \left[\frac{(\alpha_m-1)}{2\beta}+1\right]a_n\right) \\ +C\left(\frac{\gamma}{\Delta t\beta}[d_{n+1-\alpha_f}-d_n] + \left[1-\frac{\gamma(1-\alpha_f)}{\beta}\right]v_n + \frac{(1-\alpha_f)\Delta t(2\beta-\gamma)}{2\beta}a_n\right) \\ +f_{int}(d_{n+1-\alpha_f}) - F(t_{n+1-\alpha_f}) = 0. \end{aligned} \tag{2.60}$$

Applying Newton's method in order to find $d_{n+1-\alpha_f}$ that satisfies (2.60) at the i-th Newton's iteration, the following linearized equation needs to be solved in order to obtain the i-th incremental step, $\delta d^i_{n+1-\alpha_f}$,

$$\begin{aligned} K^i_e\delta d^i_{n+1-\alpha_f} &= \Delta t^2\beta[F(t_{n+1-\alpha_f}) - f_{int}(d^i_{n+1-\alpha_f})] - K_a(d^i_{n+1-\alpha_f}-d_n) \\ &+ [\Delta t(1-\alpha_m)M - \Delta t^2(\beta-(1-\alpha_f)\gamma)C]v_n \\ &+ \left[\Delta t^2\left(\frac{1-\alpha_m}{2}-\beta\right)M - \Delta t^3\frac{(1-\alpha_f)(2\beta-\gamma)}{2}C\right]a_n, \end{aligned} \tag{2.61}$$

where

$$\begin{aligned} K_a &= \left[\frac{1-\alpha_m}{1-\alpha_f}M + \Delta t\gamma C\right] \\ K^i_e &= K_a + \Delta t^2\beta K^i_{n+1-\alpha_f}, \end{aligned} \tag{2.62}$$

and the tangential stiffness matrix $K^i_{n+1-\alpha_f}$ is the Jacobian of f_{int} at $d^i_{n+1-\alpha_f}$. Once $\delta d^i_{n+1-\alpha_f}$ is obtained, $d^{i+1}_{n+1-\alpha_f}$ is updated to

$$d^{i+1}_{n+1-\alpha_f} = d^i_{n+1-\alpha_f} + \delta d^i_{n+1-\alpha_f}. \tag{2.63}$$

At the end of Newton's method, if everything goes well, it returns $d_{n+1-\alpha_f}$ that satisfies (2.60). Then, $v_{n+1-\alpha_f}$ are updated as

$$v_{n+1-\alpha_f} = \frac{\gamma}{\Delta t\beta}[d_{n+1-\alpha_f}-d_n] + \left(1-\frac{(1-\alpha_f)\gamma}{\beta}\right)v_n + \Delta t\frac{(1-\alpha_f)(2\beta-\gamma)}{2\beta}a_n, \tag{2.64}$$

and d_{n+1}, v_{n+1}, and a_{n+1} are updated as

$$\begin{aligned} d_{n+1} &= \frac{1}{(1-\alpha_f)}[d_{n+1-\alpha_f} - \alpha_f d_n] \\ v_{n+1} &= \frac{1}{(1-\alpha_f)}[v_{n+1-\alpha_f} - \alpha_f v_n] \\ a_{n+1} &= \frac{d_{n+1} - d_n}{\Delta t^2 \beta} - \frac{v_n}{\Delta \beta} + \frac{2\beta - 1}{2\beta} a_n. \end{aligned} \tag{2.65}$$

Chung and Hulbert recommended how to set generalized alpha parameters, α_f, α_m, β, and γ. Let ρ_∞ be a user-defined spectral radius in a high frequency limit with a value between 1 and 0.5. Then the generalized alpha parameters are set as

$$\begin{aligned} \alpha_f &= \frac{\rho_\infty}{\rho_\infty + 1} \\ \alpha_m &= \frac{2\rho_\infty - 1}{\rho_\infty + 1} \\ \beta &= \frac{1}{4}(1 + \alpha_f - \alpha_m)^2 \\ \gamma &= \frac{1}{2} + \alpha_f - \alpha_m. \end{aligned} \tag{2.66}$$

If more numerical dissipation is desired, then a smaller ρ_∞ value must be used. The midpoint rule is realized when $\rho_\infty = 1.0$.

2.5 Rotational Degrees of Freedom

For some structural elements, such as beam and shell elements, rotational as well as translational degrees of freedom are present. Rotational degrees of freedom are handled with a rotation matrix. The rotation matrix R is characterized by two properties,

$$\begin{aligned} \det(R) &= 1, \\ R^T R &= I, \end{aligned} \tag{2.67}$$

where the second property states that it is an orthogonal matrix. Every rotation $R(q,\theta)$ is defined by an axis (e.g., vector q) and angle (e.g., scalar θ) of rotation. If one applies a rotation matrix R to an axis q, then $Ru = u$ must be true, which is equivalent to saying q is an eigenvector of R with eigenvalue 1. The angle is determined by taking a vector p that is perpendicular to q and measuring the angle between p and Rp. Because only the direction of q (i.e., the direction of the axis of rotation) is important, one may make a compact expression of rotation by setting the magnitude of q equal to θ (i.e. $\|q\| = \theta$). Then one can denote the rotation matrix to be $R(q)$. It is clear now that one must multiply two rotation matrices in order to update a state of rotation. This implies that special care

must be taken whenever rotational degrees of freedom need to be updated via addition of a rotation. For example, update rule (2.63) within a Newton iteration should be modified for rotational degrees of freedom. If one wants to update a rotation $R(q^i_{n+1-\alpha_f})$ by adding $R(\delta q^i_{n+1-\alpha_f})$, this is done by multiplication $R(\delta q^i_{n+1-\alpha_f})R(q^i_{n+1-\alpha_f})$. Another place in the generalized alpha method where attention needs to be paid to summation is in the definition of intermediate states (2.57). There, the sum of two rotational degrees of freedom must be taken as

$$R(q_{n+1-\alpha}) = R((1-\alpha_f)q_{n+1})R(\alpha_f q_n). \tag{2.68}$$

Note that $\alpha_f q_n$ does not change the axis of rotation but only the angle of rotation. It is convenient to consider the increment whenever two rotational degrees of freedoms need to be subtracted. For example, update formula (2.58) for displacement can be expressed as

$$\Delta q_n = \Delta t v_n + \Delta t^2 \left[(\frac{1}{2} - \beta)a_n + \beta a_{n+1} \right] \tag{2.69}$$

where Δq_n represents the increment from q_n to q_{n+1}.

Chapter 3

PDE-constrained optimization

PDE-constrained optimization solves an optimization problem,

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & F(y,u) \\ \text{subject to} \quad & C(y,u) = 0, \end{aligned} \tag{3.1}$$

where $C(y,u) = 0$ is a PDE constraint. The y vector is the state variable. State variables are unknown variables in forward PDE problem. For example, temperatures in thermal PDE and displacements or velocities in structural PDE are state variables. The u is either a design or a control variable depending on the characteristics of optimization. If one wants to solve an optimal control problem, u is a control variable, whereas it is a design variable if one wants to solve an optimal design problem. It may also be some kind of parameter describing either material properties or dynamics of systems for some inverse problem. In this thesis, PDE-constrained optimal control is considered as an example of PDE-constrained optimization. In particular, thermal or structural PDE-constrained optimal control is studied here although the method developed in this thesis can also be applied to general PDE-constrained optimization. Among many possible objective functions in a PDE-constrained optimal control problem, a target state is assumed to be given. Thus, optimal control aims to find a state that is close to the target and a control that realizes that state. For example, the static optimal heat control problem with unit conductivity is formulated as

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & F(y,u) := \frac{1}{2}\int_\Omega (y-\bar{y})^2 dx + \frac{\phi}{2}\int_\Omega u^2 dx \\ \text{subject to} \quad & -\nabla^2 y = u \text{ on } \Omega \\ & y = y_c \text{ on } \Gamma_g \end{aligned} \tag{3.2}$$

Two ways of solving PDE-constrained optimization are: 1) *optimize-and-discretize* and 2) *discretize-and-optimize*. In the *optimize-and-discretize* approach, one obtains a continuous optimality condition

and then discretizes it, whereas in the *discretize-and-optimize* approach, one discretizes an objective function and constraints first and then obtains the discretized optimality condition. The two approaches do not produce identical solutions in general [17, 42] The advantage of the *optimize-and-discretize* approach is that one can use different meshes for different parts of continuous optimal control. Thus it is better to use *optimize-and-discretize* in shape optimization, in which one should consider grid movements and therefore needs to obtain derivatives of the mesh with respect to the design parameters. On the other hand, the *discretize-and-optimize* approach produces a consistent functional gradient, while gradients from the *optimize-and-discretize* approach may not be consistent, and hence may generate an ascent direction instead of a descent direction. In the method introduced here, it is important to have a consistent gradient. Thus the *discretize-and-optimize* approach is taken.

In the *discretize-and-optimize* approach, objective as well as constraint functions are discretized. Sections 3.1 and 3.4 go over discretization of objective functions for static and dynamic problems, respectively. Then formulations and necessary optimality conditions of linear static (Section 3.2), linear dynamic (Sections 3.5 and 3.6), nonlinear static (Section 3.3), and nonlinear dynamic PDE (Section 3.7) are explained.

3.1 Discretization of objective function for static problem

In the *discretize-and-optimize* approach, the objective function $F(y,u)$ in (3.2) must be discretized. For a linear heat equation, as in Chapter 2, finite element discretization of y, $\bar{y}$, and u are taken to be

$$y = \sum_{j\in\eta-\eta_g} y_j N_j(x), \quad \bar{y} = \sum_{j\in\eta-\eta_g} \bar{y}_j N_j(x), \quad u = \sum_{j\in\eta-\eta_g} u_j N_j(x). \tag{3.3}$$

Substituting these into the objective function in (3.2), the first part of $F(y,u)$ becomes

$$\begin{aligned} \frac{1}{2}\int_\Omega (y-\bar{y})^2 dx &= \frac{1}{2}\int_\Omega \left(\sum_{j\in\eta-\eta_g} y_j N_j(x) - \sum_{j\in\eta-\eta_g} \bar{y}_j N_j(x) \right)^2 dx, \\ &= \frac{1}{2} \sum_{j\in\eta-\eta_g} \sum_{k\in\eta-\eta_g} (y_j-\bar{y}_j)(y_k-\bar{y}_k) \int_\Omega N_j(x) N_k(x) dx, \\ &= \frac{1}{2}(y^h-\bar{y}^h)^T V (y^h-\bar{y}^h)^T, \end{aligned} \tag{3.4}$$

where $V = \int_\Omega N_j(x)N_k(x)dx$ is a volume matrix that represents the volume of Ω, and y^h and $\bar{y}^h$ are finite versions of y and $\bar{y}$. From now on, the superscript h in the finite version will be dropped unless there is potential for confusion. The second part of $F(y,u)$ after discretization becomes similar to

(3.4), that is,

$$\frac{\phi}{2}\int_\Omega u^2 dx = \frac{\phi}{2}{u^h}^T V u^h, \tag{3.5}$$

where u^h is the discretized version of u. By dropping the superscript h, in u^h, the objective function becomes

$$F(y,u) = \frac{1}{2}(y-\bar{y})^T V (y-\bar{y}) + \frac{\phi}{2}u^T V u. \tag{3.6}$$

The volume matrix V above is symmetric because $\int_\Omega N_i(x)N_j(x)dx = \int_\Omega N_j(x)N_i(x)dx$. Moreover, it is positive definite. For example,

$$\begin{aligned} u^T V u &= \sum_{i=1}^{n}\sum_{j=1}^{n} u_i u_j \int_\Omega N_i(x)N_j(x)dx, \\ &= \int_\Omega \left(\sum_{i=1}^{n} u_i N_i(x)\right)^2 dx, \\ &\geqslant 0. \end{aligned} \tag{3.7}$$

Thus, the objective function in (3.6) is bounded below by zero, and hence the optimization problem is well-posed. Note that the objective function can be rewritten as $F(y,u) = \frac{1}{2}\|y-\bar{y}\|_V^2 + \frac{\phi}{2}\|u\|_V^2$, using the norm induced by positive definiteness, $\|\cdot\|_V$. Although the same symmetric positive definite matrix, V, happens to be used in this specific example (i.e., the optimal distributed heat control problem), separate notation will be used for the two norms in the objective function because they may be different for other types of control problems (e.g., optimal boundary control). Plus, it is possible to modify the symmetric positive definite matrix V in order to speed up the simulation. For example, one may use a lumped volume matrix, which is a diagonal matrix. Thus, the objective function $F(y,u)$ can be expressed as

$$F(y,u) = \frac{1}{2}\|y-\bar{y}\|_V^2 + \frac{\phi}{2}\|u\|_G^2. \tag{3.8}$$

Note that for this particular problem, the objective function is quadratic in the optimization variables (y,u). Moreover, it is convex due to the positive definiteness of V and G.

3.2 Linear static PDE-constrained optimal control

Combining the discretized objective function (3.8)) with the finite element discretization of the linear static PDE in Chapter 2 (i.e., $Kv + K_c y_c = Vu$), if the forcing term u is a control variable, then the

discretized optimal control becomes

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & F(y,u) := \frac{1}{2}\|y-\bar{y}\|_V^2 + \frac{\phi}{2}\|u\|_G^2 \\ \text{subject to} \quad & Ky + K_c y_c = Vu. \end{aligned} \tag{3.9}$$

On the other hand, if one wants to consider a boundary control problem, then the optimal control formulation becomes

$$\begin{aligned} \underset{y,y_c}{\text{minimize}} \quad & F(y,y_c) := \frac{1}{2}\|y-\bar{y}\|_V^2 + \frac{\phi}{2}\|y_c\|_G^2 \\ \text{subject to} \quad & Ky + K_c y_c = Vu. \end{aligned} \tag{3.10}$$

Because discretized PDE constraints are linear in optimization variables (e.g., either (y,u) in force control or (y,y_c) in boundary control), the optimization problem becomes convex quadratic programming. In order to solve the discretized optimal control problems (3.9) or (3.10), one must consider necessary optimality conditions, which might be obtained by a Lagrangian. For example, the Lagrangian for (3.9) is

$$L(y,u,\lambda) = \frac{1}{2}\|y-\bar{y}\|_V^2 + \frac{\phi}{2}\|u\|_G^2 + \lambda^T(Ky + K_c y_c - Vu), \tag{3.11}$$

where λ is a Lagrange multiplier. The first order necessary optimality condition for linear static PDE-constrained forcing control is that there should be λ^* such that

$$\begin{bmatrix} V & & K^T \\ & \phi G & -V^T \\ K & -V & \end{bmatrix} \begin{pmatrix} y^* \\ u^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} V\bar{y} \\ 0 \\ -K_c y_c \end{pmatrix}, \tag{3.12}$$

is satisfied at the optimal solution (y^*, u^*, λ^*). Similarly, the first order necessary optimality condition for linear static PDE-constrained boundary control is that there should be λ^* such that

$$\begin{bmatrix} V & & K^T \\ & \phi G & K_c^T \\ K & K_c & \end{bmatrix} \begin{pmatrix} y^* \\ y_c^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} V\bar{y} \\ 0 \\ Vu \end{pmatrix}, \tag{3.13}$$

is satisfied at the optimal solution (y^*, y_c^*, λ^*). The Karush-Kuhn-Tucker (KKT) system (also called a saddle point system) in (3.12) and (3.13) is a symmetric indefinite matrix. In this thesis, solving this indefinite system is used as a building block. A sequential quadratic programming (SQP) method is used to solve the PDE-constrained optimization problem. At each iteration of this method a QP is solved in which the linear constraints are a linearization of nonlinear constraints. The QP subproblems have similar structure to QPs (3.9) and (3.10). It is important to have an efficient way of solving these QPs since how they are solved will affect the total performance of the SQP

algorithm significantly. There are many numerical approaches to solving this indefinite system of equations. An excellent survey paper on those methods is by Benzi, et al. [7]. The KKT system is solved using a Krylov iterative method. In Chapter 5, several existing and new iterative methods and preconditioners for solving KKT systems are presented.

3.3 Nonlinear static PDE-constrained optimal control

Combining the discretized objective function (3.8) with the finite element discretization of the nonlinear static PDE presented in Chapter 2 (i.e., $f_{int}(v^h,d^h) = Vb^h$ in the absence of traction boundary condition, using the Saint Venant-Kirchhoff constitutive law), if the body force term b is a control variable, then the discretized optimal control becomes

$$\begin{aligned} \underset{v,b}{\text{minimize}} \quad & F(v,b) := \frac{1}{2}\|v-\bar{v}\|_V^2 + \frac{\phi}{2}\|b\|_G^2 \\ \text{subject to} \quad & f_{int}(v,d) = Vb, \end{aligned} \tag{3.14}$$

where the superscript h for finite element discretization is dropped. Although the objective function is convex quadratic in v and b, the constraint is nonlinear. Thus a nonlinear optimization solver needs to be applied. At the k-th major iteration of the SQP algorithm, the following QP is solved:

$$\begin{aligned} \underset{\Delta v,\Delta b}{\text{minimize}} \quad & F(\Delta v,\Delta b) := \frac{1}{2}\|v^{(k)}+\Delta v-\bar{v}\|_V^2 + \frac{\phi}{2}\|b^{(k)}+\Delta b\|_G^2 \\ \text{subject to} \quad & K(v^{(k)},d)\Delta v - V\Delta b = Vb - f_{int}(v^{(k)},d), \end{aligned} \tag{3.15}$$

where $v^{(k)}$ and $b^{(k)}$ are the k-th current state of displacement and the body force control, respectively, and the tangential stiffness matrix $K(v^{(k)},d)$ is the Jacobian of f_{int} evaluated at $(v^{(k)},d)$. In a traditional SQP method, the objective Hessian of the QP subproblem is quasi-Newton approximation of the Hessian of the Lagrangian. here we use the exact second derivative for the objective function and ignore the Hessian of the constraints. Convergence is still observed by the rate of convergence may be impaired. In this sense, the QP formulation (3.15) is analogous to the Gauss-Newton method, where least square problem is solved with second order derivative terms omitted in the traditional Newton method. It is known that ignoring second order derivative terms in a quadratic objective function will degrade the local convergence rate from super-linear (or quadratic) to linear or may not even lead to convergence at all especially when nonlinear constraints are present. However, one may choose to ignore the second order derivative terms because they are not readily available from the PDE solver, and implementing them analytically is tedious. One may use finite difference or automatic differentiation in order to approximate or obtain second order derivatives of nonlinear constraints, but these are usually slow processes. Plus, the numerical results show that the convergence is fast enough at least with a large number of control variables although it is linear

(e.g., see Figure 7.7). Two things may result in a high enough linear convergence rate: First, the Lagrange multipliers are small and therefore the contribution of second order derivates of nonlinear constraints to the Hessian of the Lagrangian is small. Second, a SQP method tends to give a search direction that is nearly in the tangent space of nonlinear constraints (or the Null space of linearized constraints). Thus, if the nonlinear constraints are not highly nonlinear (or curvature information is not important), then the impact of second order derivative terms in nonlinear constraints is not significant. The traditional SQP is explained in Section 4.2.

In order to solve (3.15), one needs to find a point $(\Delta v^*, \Delta b^*, \lambda^*)$ that satisfies the necessary optimality condition,

$$\begin{bmatrix} V & & K(v^{(k)},d)^T \\ & G & -V^T \\ K(v^{(k)},d) & -V & \end{bmatrix} \begin{pmatrix} \Delta v^* \\ \Delta b^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} V(\bar{v} - v^{(k)}) \\ -Gb^{(k)} \\ Vb - f_{int}(v^{(k)},d) \end{pmatrix}, \tag{3.16}$$

which is also called the KKT system for the QP subproblem. Having an efficient solver for the linear system of equation (3.16) is important. Since the KKT system tends to be sparse and large due to finite discretization, an iterative method is adopted. Iterative methods for solving the KKT system are explained in Chapter 5.

3.4 Discretization of objective function for dynamic problem

Consider a transient optimal control problem in which one wants to find a state that is closest to the target $\bar{y}(t,x)$ for the whole time $t \in [t_0,t_f]$, where t_0 and t_f are initial and final times, respectively. Then, the continuous objective function $J(y,u)$ for the transient optimal control problem can be expressed as

$$J(y,u) := \frac{1}{2}\int_{t_0}^{t_f}\int_{\Omega}(y(t,x) - \bar{y}(t,x))^2 \, dx \, dt + \frac{\phi}{2}\int_{t_0}^{t_f}\int_{\Omega} u(t,x)^2 \, dx \, dt, \tag{3.17}$$

where y and u are state and control variables, respectively. In the *discretize-and-optimize* approach, the objective function $J(y,u)$ in (3.17)

must be discretized. The inner integrals in (3.14) are spatial integrals. Applying spatial discretizations as in Section 3.1, the semi-discretized objective function $\tilde{J}(y,u)$ is obtained:

$$\tilde{J}(y,u) := \frac{1}{2}\int_{t_0}^{t_f} \|y(t,\cdot) - \bar{y}(t,\cdot)\|_V^2 \, dt + \frac{\phi}{2}\int_{t_0}^{t_f} \|u(t,x)\|_G^2 \, dt. \tag{3.18}$$

An important point must be made here. For static problems, states do not involve time derivative quantities (e.g., displacement and acceleration). However, for dynamic problems, states must include time derivative quantities, especially high-order time integrator schemes such as the midpoint rule

and the generalized alpha method although one can avoid including time derivative terms in the state if the backward Euler time integrator scheme is used. Thus both state y and target state $\bar{y}$ may include time derivative information. However, having time derivative target states is not necessary (although one can always obtain them approximately via the finite difference method). One can either set elements in V matrix that correspond to time derivative states to zero or make them very small so that the difference between target time derivative states are negligible in the objective function.

There are many numerical approximations for one-dimensional integrals from the simplest one (e.g., Simpson's rule) to more sophisticated ones (e.g., the Gaussian quadrature). The goal of the objective function is to express some kind of scalar quantity of difference between state and target states in both time and space domains. As long as this goal is met, it does not matter what kind of discretization in the time domain is used. Thus, the Riemann-type approximation of the integral is chosen, where the integral is approximated by the sum of the product of each time domain interval with the corresponding function value. Thus, the fully discretized objective function $\bar{J}(y,u)$ is given as

$$\bar{J}(y,u) := \frac{1}{2}\sum_{n=0}^{N} \Delta t \left(\|y_n - \bar{y}_n\|_V^2\right) + \frac{\phi}{2}\sum_{n=0}^{N} \Delta t \left(\|u_n\|_G^2\right), \tag{3.19}$$

where $y_n = y(t_n)$ and $u_n = u(t_n)$ are spatial discretized states and controls at time $t = t_n$. The time domain (t_0,t_f) is supposed to be discretized uniformly with time interval $\Delta t = \frac{t_f - t_0}{N}$; hence the time t_n is expressed as $t_n = t_0 + n\Delta t$. Finally, the optimization variables are discretized states $y = (y_0, y_1, \ldots, y_N)^T$ and controls $u = (u_0, u_1, \ldots, u_N)^T$ both in time and space. It is possible to rewrite the objective function (3.19) in matrix form, $\frac{\Delta t}{2}(y-\bar{y})^T V_s (y-\bar{y}) + \frac{\phi}{2}\Delta t u^T G_s u$, where

$$V_s = \begin{bmatrix} V & 0 & \cdots & \cdots & 0 \\ 0 & V & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & V \end{bmatrix} \tag{3.20}$$

if a target state is given for all of the time domain (t_0,t_f). In shorthand, V_s can be expressed as $V_s = blkdiag(V,\ldots,V)$. Similarly, $G_s = blkdiag(G,\ldots,G)$. It might be the case that one only wants to match final target state y_N. In this case, the objective function becomes

$$\bar{J}_f(y,u) := \frac{\Delta t}{2}\left(\|y_N - \bar{y}_N\|_V^2\right) + \frac{\phi}{2}\sum_{n=0}^{N} \Delta t \left(\|u_n\|_G^2\right), \tag{3.21}$$

and in compact form, $\frac{\Delta t}{2}(y-\bar{y})^T V_{sf} (y-\bar{y}) + \frac{\phi}{2}\Delta t u^T G_s u$, where $V_{sf} = blkdiag(0,\ldots,0,V)$.

3.5 Linear dynamic PDE-constrained optimal heat control

For the linear dynamic optimal heat control, combining the discretized objective function (3.19) or (3.21) with the finite element discretization of linear dynamic PDEs discussed in Chapter 2 (i.e., equations (2.40) and (2.32)), the discretized formulation for a given initial temperature state d^0 is

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & \bar{J}(y,u) := \frac{1}{2}\sum_{n=0}^{N} \Delta t \|y_n - \bar{y}_n\|_V^2 + \frac{\phi}{4}\Delta t \|u_0\|_G^2 + \frac{\phi}{2}\sum_{n=1}^{N} \Delta t \|u_{n-\frac{1}{2}}\|_G^2 \\ \text{subject to} \quad & d_0 = d^0 \\ & Mv_0 = (u_0 - Kd_0) \\ & D_h y_{n+1} = U_{n+\frac{1}{2}} u_{n+\frac{1}{2}} + R_h y_n \qquad \text{for } n = 0,\ldots,N-1, \end{aligned} \tag{3.22}$$

where $D_h = \begin{bmatrix} \bar{K} & \\ & \frac{1}{\Delta t}\bar{K} \end{bmatrix}$, $R_h = \begin{bmatrix} M - \frac{\Delta t}{2}K & 0 \\ -2K & -\frac{1}{\Delta t}\bar{K} \end{bmatrix}$, $U_{n+\frac{1}{2}} = \begin{bmatrix} \Delta t I \\ 2I \end{bmatrix}$, and $\bar{K} = M + \frac{\Delta t}{2}K$. The state at time t_n is composed of temperature and its temporal derivative, that is, $y_n = (d_n^T, \Delta t v_n^T)^T$. Note that temporal derivative of temperature, v_n, gets multiplied by the time step Δt in order to scale temperature and its derivative to be of the same order. This is because the temporal derivative of the temperature is an order of $\frac{1}{\Delta t}$ bigger than temperature in magnitude. Similarly, the target state at time t_n is composed of $\bar{y}_n = (\bar{d}_n, \Delta t \bar{v}_n)$. Finally, the heat source control vector $u = (u_0, u_{\frac{1}{2}}^T, u_{1+\frac{1}{2}}^T, \ldots, u_{N-\frac{1}{2}}^T)^T$. The formulation (3.22) can be compactly expressed as

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & \bar{J}(y,u) := \frac{\Delta t}{2}(y-\bar{y})^T V_s (y-\bar{y}) + \frac{\phi}{2}\Delta t \left(u^T G_s u\right) \\ \text{subject to} \quad & Qy = Uu + d, \end{aligned} \tag{3.23}$$

where $G_s = blkdiag(G/2, G, \ldots, G)$ and Q, U, and d are defined as

$$Q = \begin{bmatrix} R_0 & 0 & \cdots & \cdots & 0 \\ -R_h & D_h & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & -R_h & D_h \end{bmatrix}, \qquad d = \begin{pmatrix} d^0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{3.24}$$

and $U = blkdiag(U_0, U_{\frac{1}{2}}, \ldots, U_{N-\frac{1}{2}})$, where $R_0 = \begin{bmatrix} I & \\ K & M \end{bmatrix}$, $U_0 = \begin{bmatrix} 0 \\ I \end{bmatrix}$, and d^0 is a given initial temperature state. Note that the formulation (3.23) is a QP. The necessary optimality condition

states that a solution (y^*,u^*,λ^*) to (3.23) must satisfy

$$\begin{bmatrix} V_s & & Q^T \\ & \phi G & -U^T \\ Q & -U & \end{bmatrix} \begin{pmatrix} y^* \\ u^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} V_s \bar{y} \\ 0 \\ d \end{pmatrix}, \tag{3.25}$$

which is the KKT system. In order to solve the KKT system, a Krylov subspace-based iterative method such as GMRES is chosen. In such an iterative method, only matrix-vector multiplications are required if no preconditioner is applied, which is the ideal case. However, most of the time, for the best performance, one needs a preconditioner. Preconditioners are designed to resemble the original matrix but to be easier to invert or solve for (e.g., the Jacobi preconditioner takes the diagonal of the original matrix). Thus, it is important to examine the number and ways of solving block matrices of of the KKT matrix. V_s and G are used as a positive definite norm in the objective function; thus the KKT matrix can be modified to be a simple positive definite matrix (e.g., a positive diagonal matrix). The only potentially computationally demanding blocks are Q and Q^T. If one needs to solve for $Qp_1 = w_1$, then one solve with M and $2N$ solves with $\bar{K}$ are required, which is also true for $Qp_2 = w_2$. Thus, if a preconditioner requires solving with Q and Q^T, two solves with M and $4N$ solves with $\bar{K}$ are required. Of course, one can approximate M and $\bar{K}$ since they are used within a preconditioner. However, since performance of a preconditioner depends on the approximation done for M and $\bar{K}$, it is important to keep track of the counts and types of computations that are required to apply a preconditioner.

3.6 Linear dynamic PDE-constrained optimal structure control

For the dynamic optimal strurcture force control, combining the discretized objective function (3.19) or (3.21) with finite element discretization of linear dynamic PDEs in Chapter 2 (i.e., equation (2.53)), the discretized formulation becomes for given initial displacement d^0 and velocity v^0,

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & \bar{J}(y,u) := \frac{1}{2} \sum_{n=0}^{N} \Delta t \|y_n - \bar{y}_n\|_V^2 + \frac{\phi}{2} \sum_{n=1}^{N} \Delta t \|u_{n-\frac{1}{2}}\|_G^2 \\ \text{subject to} \quad & d_0 = d^0 \\ & v_0 = v^0 \\ & D_s y_{n+1} = U_{n+\frac{1}{2}} u_{n+\frac{1}{2}} + R_s y_n \qquad \text{for } n = 0, \ldots, N-1, \end{aligned} \tag{3.26}$$

where $D_s = \begin{bmatrix} \hat{K} & \\ & \frac{1}{\Delta t}\hat{K} \end{bmatrix}$, $R_s = \begin{bmatrix} \tilde{K} - \frac{\Delta t^2}{4}K & M \\ -\Delta t K & \frac{2}{\Delta t}M - \frac{1}{\Delta t}\hat{K} \end{bmatrix}$, $U_{n+\frac{1}{2}} = \begin{bmatrix} \frac{\Delta t^2}{2}I \\ \Delta t I \end{bmatrix}$, $\tilde{K} = M + \frac{\Delta t}{2}C$, and $\hat{K} = \tilde{K} + \frac{\Delta t^2}{4}K$. The state at time t_n is composed of displacement and velocity, that is, $y_n = (d_n^T, \Delta t v_n^T)^T$. Note that velocity v_n gets multiplied by the time step Δt in order to scale displacement and velocity to be of the same order. This is because velocity is an order of $\frac{1}{\Delta t}$ bigger than displacement in magnitude. Similarly, the target state at time t_n is composed of $\bar{y}_n = (\bar{d}_n, \Delta t \bar{v}_n)$. Finally, the force control vector $u = (u_{\frac{1}{2}}^T, u_{1+\frac{1}{2}}^T, \ldots, u_{N-\frac{1}{2}}^T)^T$. Note that u_0 is not needed in the midpoint rule for structural problem because unlike in the thermal transient problem the initial velocity v^0 must be given in order for the time integrator to proceed. The formulation (3.26) can be compactly expressed as

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & \bar{J}(y,u) := \frac{\Delta t}{2}(y-\bar{y})^T V_s (y-\bar{y}) + \frac{\phi}{2}\Delta t \left(u^T G_s u\right) \\ \text{subject to} \quad & Qy = Uu + d, \end{aligned} \tag{3.27}$$

where $G_s = blkdiag(G, \ldots, G)$ and Q, U, and d are defined as

$$Q = \begin{bmatrix} I & 0 & \cdots & \cdots & 0 \\ -R_s & D_s & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & -R_s & D_s \end{bmatrix}, \qquad d = \begin{pmatrix} d^0 \\ v^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{3.28}$$

and $U = blkdiag(U_0, U_{\frac{1}{2}}, \ldots, U_{N-\frac{1}{2}})$, where $U_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and d^0 and v^0 are given initial displacement and velocity states, respectively. Note that formulation (3.27) is QP. The necessary optimality condition states that a solution (y^*,u^*,λ^*) to (3.27) must satisfy

$$\begin{bmatrix} V_s & & Q^T \\ & \phi G & -U^T \\ Q & -U & \end{bmatrix} \begin{pmatrix} y \\ u \\ \lambda \end{pmatrix} = \begin{pmatrix} V_s \bar{y} \\ 0 \\ d \end{pmatrix}. \tag{3.29}$$

As in the linear dynamic PDE-constrained heat control discussed in 3.5, it is important to keep track of types and counts of solves needed to solve $Qp_1 = w_1$ or $Q^T p_2 = w_2$. For a given Q, $2N$ solves of $\hat{K}$ are needed, which is also true for Q^T. Thus, if one needs to solve both $Qp_1 = w_1$ and $Q^T p_2 = w_2$, then $4N$ solves of $\hat{K}$ are needed in total.

3.7 Nonlinear dynamic PDE-constrained optimal control

Given current displacement, velocity, and acceleration states, d_n, v_n, and a_n, the generalized α method proceeds with six equations (e.g., equations (2.56), (2.57), and (2.58)) and six unknowns (i.e., d_{n+1}, v_{n+1}, a_{n+1}, $d_{n+1-\alpha_f}$, $v_{n+1-\alpha_f}$, and $a_{n+1-\alpha_m}$). The PDE solver (e.g., AERO-S) treats the generalized α method by solving for $d_{n+1-\alpha_f}$ by Newton's method (i.e. equation (2.61)), then obtains other unknowns d_{n+1}, v_{n+1}, a_{n+1}, $v_{n+1-\alpha_f}$, and $a_{n+1-\alpha_m}$. This approach is beneficial in the sense that it deals with a smaller system of equations (i.e., the size of the system is n_s if $d_n \in \mathbb{R}^{n_s}$). However, in the full SAND method, it is necessary to express the generalized α method in a compact form, where all the unknowns are solved simultaneously. In this setting, dealing with six unknown vector variables simultaneously (i.e., the size of the system is $6m$) is too daunting. Fortunately it is possible to eliminate three equations and three intermediate unknowns (e.g., $d_{n+1-\alpha_f}$, $v_{n+1-\alpha_f}$, and $a_{n+1-\alpha_m}$) and express the time integrator with only end points (e.g., d_{n+1}, v_{n+1}, and a_{n+1}). This is done by finding an expression for $d_{n+1-\alpha_f}$ from (2.60), that is,

$$\begin{aligned} d_{n+1-\alpha_f} &= \Delta t^2 \beta K_a^{-1} \left[u(t_{n+1-\alpha_f}) - f_{int}((1-\alpha_f) d_{n+1} + \alpha_f d_n) \right] \\ &+ d_n \\ &+ \left[\Delta t (1-\alpha_m) K_a^{-1} M - \Delta t^2 (\beta - (1-\alpha_f)\gamma) K_a^{-1} C \right] v_n \\ &+ \left[\Delta t^2 \left(\frac{1-\alpha_m}{2} - \beta \right) K_a^{-1} M - \Delta t^3 \frac{(1-\alpha_f)(2\beta-\gamma)}{2} K_a^{-1} C \right] a_n, \end{aligned} \tag{3.30}$$

and plugging it into the first intermediate interpolation equation of (2.57) in order to remove $d_{n+1-\alpha_f}$, that is,

$$\begin{aligned} K_a (d_{n+1} - d_n) &= \Delta t^2 \frac{\beta}{1-\alpha_f} \left[u(t_{n+1-\alpha_f}) - f_{int}((1-\alpha_f) d_{n+1} + \alpha_f d_n) \right] \\ &+ \left[\Delta t \frac{(1-\alpha_m)}{(1-\alpha_f)} M - \Delta t^2 \left(\frac{\beta}{1-\alpha_f} - \gamma \right) C \right] v_n \\ &+ \left[\Delta t^2 \left(\frac{1-\alpha_m}{2(1-\alpha_f)} - \frac{\beta}{1-\alpha_f} \right) M - \Delta t^3 \frac{(2\beta-\gamma)}{2} C \right] a_n, \end{aligned} \tag{3.31}$$

where the notation u is used for the forcing term instead of F in order to emphasize that the term is used as a control variable. Combining this equation with two updating equations of the Newmark method (2.58), which consists of only end points, completes the elimination of intermediate variables in the generalized α method.

There are two ways of formulating nonlinear dynamic optimal structure control problems. The first one is a simultaneous approach, where all the time steps of states and controls are solved simultaneously. The discretized formulation of nonlinear dynamic optimal control with objective

function (3.19) becomes for given initial displacement d^0 and velocity v^0,

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & \bar{J}(y,u) := \frac{1}{2}\sum_{n=0}^{N} \Delta t \|y_n - \bar{y}_n\|_V^2 + \frac{\phi}{4}\Delta t \|u_0\|_G^2 + \frac{\phi}{2}\sum_{n=1}^{N} \Delta t \|u_{n-\frac{1}{2}}\|_G^2 \\ \text{subject to} \quad & d_0 = d^0 \\ & v_0 = v^0 \\ & Ma_0 + Cv_0 + f_{int}(d_0) = u_0 \\ & g_\alpha(y_{n+1},y_n,u_{n+1-\alpha_f}) = 0 \qquad \text{for } n = 0,\ldots,N-1, \end{aligned} \tag{3.32}$$

where g_α stands for the generalized α update function, which is represented by the three equations (3.31) and (2.58). The state at time t_n is decomposed into displacement, velocity, and acceleration, i.e., $y_n = (d_n^T, v_n^T, a_n^T)^T$. Finally, the control is $u = (u_0^T, u_{1-\alpha_f}^T, \ldots, u_{N-\alpha_f}^T)^T$. Note that the KKT system corresponding to the formulation (3.32) is in $\mathbb{R}^{(6n_s+n_c)(N+1)\times(6n_s+n_c)(N+1)}$, where n_s is the size of displacement and n_c the size of control at a stationary time t_n. Considering the degrees of discretization both in space and time, the size is daunting. The second formulation is to solve smaller optimization problems sequentially. For example, initial states y_0 and control u_0 are completely determined by the following optimization problem:

$$\begin{aligned} \underset{a_0,u_0}{\text{minimize}} \quad & \frac{\Delta t}{2}\|a_0 - \bar{a}_0\|_{V_a}^2 + \frac{\phi}{4}\Delta t \|u_0\|_G^2 \\ \text{subject to} \quad & Ma_0 + Cv_0 + f_{int}(d_0) = u_0, \end{aligned} \tag{3.33}$$

where d_0 and v_0 are dropped in the optimization variables because they can be predetermined by the constraints $d_0 = d^0$ and $v_0 = v^0$. Once y_0 and u_0 are found, one can sequentially solve the following optimization problem:

$$\begin{aligned} \underset{y_n,u_{n-\alpha_f}}{\text{minimize}} \quad & \frac{\Delta t}{2}\|y_n - \bar{y}_n\|_V^2 + \frac{\phi}{2}\Delta t \|u_{n-\alpha_f}\|_G^2 \\ \text{subject to} \quad & g_\alpha(y_n,y_{n-1},u_{n-\alpha_f}) = 0 \end{aligned} \tag{3.34}$$

for $n = 1,\ldots,N$. Note that the KKT system corresponding to the formulation (3.33) is in $\mathbb{R}^{(2n_s+n_c)\times(2n_s+n_c)}$ and the one corresponding to the formulation (3.34) is in $\mathbb{R}^{(6n_s+n_c)\times(6n_s+n_c)}$. Although these KKT systems can still be large depending on how fine the discretization is, they are much smaller than the KKT system corresponding to the formulation (3.32), which is $\mathbb{R}^{(6n_s+n_c)(N+1)\times(6n_s+n_c)(N+1)}$. The two approaches of formulations (i.e., the simultaneous one (3.32) and the sequential ones (3.33) and (3.34)) are not equivalent. This can be easily seen by comparing the necessary optimality conditions.

For example, one of the necessary optimality condition for the formulation (3.32) is

$$\Delta t V(y_n - \bar{y}_n) + \left(\frac{\partial g_\alpha(y_n, y_{n-1}, u_{n-\alpha_f})}{\partial y_n}\right)^T \lambda_n + \left(\frac{\partial g_\alpha(y_{n+1}, y_n, u_{n+1-\alpha_f})}{\partial y_n}\right)^T \lambda_{n+1} = 0, \tag{3.35}$$

while the corresponding condition for the sequential formulation does not have the last term since $g_\alpha(y_{n+1}, y_n, u_{n+1-\alpha_f})$ is not needed in the formulation (3.34). However, if y_n is close to $\bar{y}_n$ (i.e., the state solution is close to the target state), then the Lagrange multipliers λ_n and λ_{n+1} are supposed to be small, so the two formulations will result in similar solutions. Unfortunately, this implies that the two formulations are totally different if the objective function (3.21) is used. To see this issue in a different way, if the objective function (3.21) the initial optimization problem one needs to solve becomes

$$\begin{aligned} \underset{a_0, u_0}{\text{minimize}} \quad & \frac{\phi}{4}\Delta t \|u_0\|_G^2 \\ \text{subject to} \quad & Ma_0 + Cv_0 + f_{int}(d_0) = u_0, \end{aligned} \tag{3.36}$$

whose control solution is zero, where a_0 is determined by $Ma_0 + Cv_0 + f_{int}(d_0) = 0$. For the subsequent optimizations, the control solutions at all the time steps except for the last one, $t_N = t_f$, are zero. This solution is undesirable and unaccepted because the solution states are identical to the initial states (d^0, v^0, a_0) throughout the whole time domain, and the last sequence of the optimization problem attempts a big jump from the initial states to the final target states ($\bar{d}_N$, $\bar{v}_N$, $\bar{a}_N$). Since it is not hard to provide target states for all the time steps (e.g., via some kind of interpolations between initial conditions and terminal target states), the sequential formulation is implemented.

Note that the constraints are nonlinear in y_n, especially in displacement states because of the internal force $f_{int}(\cdot)$ that appears in both formulations (3.33) and (3.34). However, the initial optimization problem (3.33) is QP, whose necessary optimality condition is expressed as the KKT system:

$$\begin{bmatrix} \Delta t V_a & & M^T \\ & \frac{\phi}{2}\Delta t G & -I \\ M & -I & \end{bmatrix} \begin{pmatrix} a_0 \\ u_0 \\ \lambda \end{pmatrix} = \begin{pmatrix} \Delta t V_a \bar{a}_0 \\ 0 \\ -Cv_0 - f_{int}(d_0) \end{pmatrix}. \tag{3.37}$$

Once a_0 is obtained, it is fed into the next sequential nonlinear optimization (3.34) with $n = 1$. Another option is to not solve the initial problem (3.33) at all and instead to feed zero initial acceleration (i.e $a_0 = 0$) into the next optimization problem. This implies that one chooses to apply no initial external or body force on the system and allows the system to move freely or stay static if the initial velocity is zero. The consequence of the resulting lack of force may be a lag in target motion. This option may be chosen if a small amount of lag is allowed. For the subsequent nonlinear programming, it is necessary to apply a nonlinear solver. Sequential quadratic programming (SQP) is used as in the nonlinear static cases. The SQP method provides the search direction $(\Delta y_n, \Delta u_n)$

by solving the following QP subproblem:

$$\begin{aligned} \underset{\Delta y_n, \Delta u_{n-\alpha_f}}{\text{minimize}} \quad & \frac{\Delta t}{2}\|\Delta y_n + y_n - \bar{y}_n\|_V^2 + \frac{\phi}{2}\Delta t\|u_{n-\alpha_f} + \Delta u_{n-\alpha_f}\|_G^2 \\ \text{subject to} \quad & \text{linearized version of } g_\alpha(y_n, y_{n-1}, u_{n-\alpha_f}) = 0. \end{aligned} \tag{3.38}$$

Note that the only nonlinear term in g_α is internal force $f_{int}((1-\alpha_f)d_n + \alpha_f d_{n-1})$ in equation (3.31). Linearizing it gives

$$\begin{aligned} (K_a + \Delta t^2\beta K_{n-\alpha_f})\Delta d_n - \frac{\Delta t^2\beta}{1-\alpha_f}\Delta u_{n-\alpha_f} = & \frac{\Delta t^2\beta}{1-\alpha_f}\left[u_{n-\alpha_f} - f_{int}((1-\alpha_f)d_n + \alpha_f d_{n-1})\right] \\ & - K_a(d_n - d_{n-1}) \\ & + \left[\Delta t\frac{(1-\alpha_m)}{(1-\alpha_f)}M - \Delta t^2(\frac{\beta}{1-\alpha_f} - \gamma)C\right]v_{n-1} \\ & + \left[\Delta t^2\left(\frac{1-\alpha_m}{2(1-\alpha_f)} - \frac{\beta}{1-\alpha_f}\right)M - \Delta t^3\frac{(2\beta-\gamma)}{2}C\right]a_{n-1}, \end{aligned} \tag{3.39}$$

where $K_{n-\alpha_f}$ is Jacobian (or tangential stiffness matrix) of internal force f_{int} at $d_{n-\alpha_f}$. Expressing the Newmark updates with incremental states Δy_n gives

$$\begin{aligned} \Delta t^2\Delta a_n - \frac{1}{\beta}\Delta d_n &= \frac{1}{\beta}(d_n - d_{n-1}) - \frac{\Delta t}{\beta}v_{n-1} + \frac{2\beta-1}{2\beta}\Delta t^2 a_{n-1} - \Delta t^2 a_n \\ -\Delta t^2\gamma\Delta a_n + \Delta t\Delta v_n &= \Delta t(v_{n-1} - v_n) + \Delta t^2[(1-\gamma)a_{n-1} + \gamma a_n], \end{aligned} \tag{3.40}$$

where unknowns in QP (3.38) are grouped in the lefthand side and the rest of the terms in the righthand side. Combining (3.39) and (3.40) forms the linearized version of $g_\alpha(y_n, y_{n-1}, u_{n-\alpha_f})$, which can be expressed in a compact form as $Q_n\Delta y_n + U\Delta u_{n-\alpha_f} = r_n$, where

$$Q_n = \begin{bmatrix} K_a + \Delta t^2\beta K_{n-\alpha_f} & 0 & 0 \\ -\frac{1}{\beta}I & 0 & \Delta t^2 I \\ 0 & \Delta t I & -\Delta t^2\gamma I \end{bmatrix}, \quad U = \begin{bmatrix} -\frac{\Delta t^2\beta}{1-\alpha_f}I \\ 0 \\ 0 \end{bmatrix}, \tag{3.41}$$

and residual

$$r_n = \begin{pmatrix} \frac{\Delta t^2\beta}{1-\alpha_f}\left[u_{n-\alpha_f} - f_{int}((1-\alpha_f)d_n + \alpha_f d_{n-1})\right] \\ -K_a(d_n - d_{n-1}) \\ + \left[\Delta t\frac{(1-\alpha_m)}{(1-\alpha_f)}M - \Delta t^2(\frac{\beta}{1-\alpha_f} - \gamma)C\right]v_{n-1} \\ + \left[\Delta t^2\left(\frac{1-\alpha_m}{2(1-\alpha_f)} - \frac{\beta}{1-\alpha_f}\right)M - \Delta t^3\frac{(2\beta-\gamma)}{2}C\right]a_{n-1} \\ \frac{1}{\beta}(d_n - d_{n-1}) - \frac{\Delta t}{\beta}v_{n-1} + \frac{2\beta-1}{2\beta}\Delta t^2 a_{n-1} - \Delta t^2 a_n \\ \Delta t(v_{n-1} - v_n) + \Delta t^2[(1-\gamma)a_{n-1} + \gamma a_n] \end{pmatrix}. \tag{3.42}$$

The necessary optimality condition for (3.38) is

$$\begin{bmatrix} \Delta tV & & Q_n^T \\ & \phi\Delta tG & U^T \\ Q_n & U & \end{bmatrix} \begin{pmatrix} \Delta y_n \\ \Delta U_{n-\alpha_f} \\ \lambda \end{pmatrix} = \begin{pmatrix} \Delta tV(\bar{y}_n - y_n) \\ -\phi\Delta tGu_{n-\alpha_f} \\ r_n \end{pmatrix}. \tag{3.43}$$

Note that the part where the most computational effort is required to solve $Q_n p_1 = q_1$ and $Q_n^T p_2 = q_2$ is to solve for $K_a + \Delta t^2 \beta K_{n-\alpha_f}$. The rest of the parts are simple algebraic operations (e.g., scalar products).

Chapter 4

Methods for PDE-constrained optimization

To solve PDE-constrained optimization, two methodologies are: NAND (Nested Analysis and Design) and SAND (Simultaneous Analysis and Design). SAND is focused on in this thesis. SAND has another name, the "all-at-once" approach. In the "all-at-once" approach, the numerical methods are again divided into two categories: 1. full space (full SAND) and 2. reduced space methods (reduced SAND). If sequential quadratic programming (SQP) method is used to solve the problem, the full space method is called full SQP (fSQP), while the reduced space method is called reduced SQP (rSQP). The full space method directly solves a saddle-point necessary optimality system, for example, by using an iterative method, such as a Krylov iterative method [9, 10, 68] [79, 45]. Reduced space methods have many variants, but the main idea is to introduce decomposition of the optimization variables (e.g., state and control variables) and reduce the size of optimization to the number of control variables [82, 54]. Research on solving the saddle-point system efficiently is very important because the system appears as both the main problem in linear PDE-constrained optimization and as a QP sub-problem in nonlinear PDE-constrained optimization.

NAND is preferable regarding storage issues because it does not have as many variables as SAND does. Plus, one can use available optimization and PDE solvers separately as long as the PDE solver can provide sensitivity information to the optimization solver. Thus, it has been the method of choice for a long time and it is still often preferred Some disadvantages of NAND are that it usually requires many iterations to converge and that it may be hard to get sensitivity information within the PDE solver either because it is tedious to compute or because it is computationally expensive if one attempts to get the information via a finite difference scheme. Reduced SAND forms a reduced KKT system of equations and solves smaller system at each iteration than Full SAND, but it requires even more storage than NAND. Unless one forms the reduced KKT system explicitly and factorizes it

exactly, reduced SAND usually solves a reduced KKT system approximately either by using a quasi-Newton approach or by omitting some terms, so it takes many iterations to converge. For example, for QP, it takes more than one iteration to converge. On the other hand, although the full space method is likely to be ill-conditioned and indefinite in nature, it exhibits better scalability than the reduced space method if a good preconditioner is used. For example, if an optimal preconditioner is used, then one can expect to converge with a constant number of iterations within a Krylov iterative method regardless of the size of the problem. Recent advances in numerical linear algebra have shifted more attention to full SAND. One reason is that it is easier to see many more ways to develop a preconditioner for full SAND than for reduced SAND. However, in spite of all the development and improvements in full SAND, much further research needs to be done. For example, if the PDE is time-dependent, the storage requirement becomes so huge that it is not clear how to implement full SAND in transient analysis for the whole time domain. Plus more effort on developing new efficient preconditioners is needed, especially for a small regularization parameter. In this chapter, NAND and reduced and full SAND are explained.

4.1 Nested analysis and design

In nested analysis and design, only control variables, u, are optimization variables, and state variables y are considered to be an implicit function of control variables through the PDE equations. If there are no additional constraints, then an optimization solver needs to solve the unconstrained optimization problem. That is, NAND solves

$$\underset{u}{\text{minimize}} \quad F(u,y(u)), \tag{4.1}$$

where $y(u)$ is expressed implicitly in the PDE, $C(y,u) = 0$. There are many numerical algorithms dealing with unconstrained optimization problems, but all can be put into one of two categories: gradient and non-gradient based algorithms. Non-gradient based algorithms are zero-order methods, whereas gradient based algorithms are higher-order methods. Thus, gradient based algorithms are more popular than non-gradient based one. Among gradient based algorithms, some use only first derivatives and others also use second derivatives. If a Hessian is used properly, a second order asymptotic convergence rate is obtained if it converges. However, a Hessian may not be available or may be tedious to compute. In either case, one can approximate a Hessian with only gradient information via a quasi-Newton approach, where super-linear convergence rate may be achieved. If neither a Hessian nor an approximate Hessian, but only the gradient is used, then normally a linear convergence rate will be observed. Assuming that a gradient-based method without high order derivative information (e.g., a Hessian) is used, the gradient g of $F(u,y(u))$ in (4.1) is derived

in this section. The gradient is obtained by the chain rule

$$g := \frac{dF}{du} = \frac{\partial F}{\partial u} + \frac{\partial F}{\partial y} \cdot \frac{\partial y}{\partial u}. \tag{4.2}$$

Consider $\frac{\partial F}{\partial y}$ or $\frac{\partial F}{\partial u}$ to be a row vector and $\frac{\partial F}{\partial y} \cdot \frac{\partial y}{\partial u}$ as usual the row vector-matrix product. The only unknown is $\frac{\partial y}{\partial u}$. This can be obtained from the fact that the derivative of PDE should vanish, that is,

$$0 = \frac{dC}{du} = \frac{\partial C}{\partial y} \cdot \frac{\partial y}{\partial u} + \frac{\partial C}{\partial u}. \tag{4.3}$$

This results in the gradient being

$$g = \frac{\partial F}{\partial u} + \frac{\partial F}{\partial y} \left(\frac{\partial C}{\partial y} \right)^{-1} \frac{\partial C}{\partial u}. \tag{4.4}$$

There are two ways of computing the gradient: the direct and the adjoint approach. The direct approach solves the last two terms in the second part of (4.4), while the adjoint approach solves the first two terms. In other words, the direct approach takes the following two steps in order to compute the gradient:

1. solve for $\frac{\partial y}{\partial u}$ in (4.3)
2. compute g via (4.2),

and the adjoint approach takes the following two steps:

1. solve for λ in $\left(\frac{\partial C}{\partial y} \right)^T \lambda - \left(\frac{\partial F}{\partial y} \right)^T = 0$
2. compute $g = \frac{\partial F}{\partial u} + \lambda^T \frac{\partial C}{\partial u}$.

Let n_c be the number of PDE constraints and n_u the number of control variables. Note that the number of state variables is the same as the number of PDE constraints. Thus, the size of $\frac{\partial C}{\partial y}$ is $n_c \times n_c$. In general, the direct approach is more expensive than the adjoint approach since the direct approach needs to solve an $n_c \times n_c$ system of equations n_u times (i.e., solving (4.3)). On the other hand, the adjoint method needs to solve an $n_c \times n_c$ system of equations only once (i.e. solving for λ). However, the adjoint solver (i.e., $\left(\frac{\partial C}{\partial y} \right)^T$) may not be available since the PDE solver has not implemented it. The only options then are modify the PDE solvers or use the direct method. Thus, the direct method might be preferred if n_c is small.

It is clear now why the name, "NAND (Nested Analysis and Design)" was coined. The PDE solver is nested inside of the optimization solver. Thus, one can use any two available, separate PDE and optimization solvers. The nested PDE solver must provide updated state variables that satisfy PDE constraints and corresponding sensitivity information (e.g., gradient g). Then, the optimization solver uses the information to generate the next point of control variables and feed

them to the PDE solver. It repeats this process until the minimizer u^* that minimizes F in (4.1) is found. One obvious convergence criterion is to check the norm of the gradient (i.e. $\|g\|$). Note that the optimization solver needs only control variables to be stored, which has the benefit of less storage than is needed for SAND. Also, note that it keeps PDE constraints (e.g., C=0) implicitly satisfied all the time.

4.2 Simultaneous analysis and design

In simultaneous analysis and design, PDE constraints are explicitly specified. That is, SAND solves the following optimization:

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & F(y,u), \\ \text{subject to} \quad & C(y,u) = 0, \end{aligned} \tag{4.5}$$

where $C(y,u)$ is an PDE constraints. The Lagrangian is

$$L(y,u,\lambda) = F(y,u) + \lambda^T C(y,u), \tag{4.6}$$

where the λ are Lagrange multipliers. The first order necessary optimality condition states that there should be λ^* such that

$$\begin{aligned} 0 &= \frac{\partial F}{\partial y}(y^*,u^*) + \left(\frac{\partial C}{\partial y}\right)^T (y^*,u^*)\lambda^* \\ 0 &= \frac{\partial F}{\partial u}(y^*,u^*) + \left(\frac{\partial C}{\partial u}\right)^T (y^*,u^*)\lambda^* \\ 0 &= C(y^*,u^*), \end{aligned} \tag{4.7}$$

is satisfied at the optimal solution, (y^*,u^*,λ^*). If $F(y,u)$ is quadratic and $C(y,u)$ linear in y and u, then the optimization formulation (4.5) becomes a quadratic program (QP) and the conditions (4.7) becomes linear system of equations in (y^*,u^*,λ^*). Thus, one linear solve will give a solution if it exists. If the formulation (4.5) is nonlinear, but not QP, then the equations in (4.7) are still linear in multiplier estimates λ but are nonlinear in optimization variables, (y, u), so a Newton type numerical method needs to be applied in order to solve them. The standard Newton method applied to (4.7) gives the search directions $\Delta x = (\Delta y, \Delta u)$ and the updated multiplier estimates λ, which satisfy

$$\begin{aligned} H_{yy}\Delta y + H_{yu}\Delta u + &= -\frac{\partial F}{\partial y} - \left(\frac{\partial C}{\partial y}\right)^T \lambda \\ H_{uy}\Delta y + H_{uu}\Delta u + &= -\frac{\partial F}{\partial u} - \left(\frac{\partial C}{\partial u}\right)^T \lambda \\ \frac{\partial C}{\partial y}\Delta y + \frac{\partial C}{\partial u}\Delta u &= -C(y,u), \end{aligned} \tag{4.8}$$

where the Hessian is defined as

$$\begin{aligned} H_{yy} &= \frac{\partial^2 F}{\partial y^2} + \sum_{i=1}^{n_y} \lambda_i \left(\frac{\partial^2 C_i}{\partial y^2} \right), \\ H_{uu} &= \frac{\partial^2 F}{\partial u^2} + \sum_{i=1}^{n_y} \lambda_i \left(\frac{\partial^2 C_i}{\partial u^2} \right), \\ H_{yu} &= \frac{\partial^2 F}{\partial y \partial u} + \sum_{i=1}^{n_y} \lambda_i \left(\frac{\partial^2 C_i}{\partial y \partial u} \right), \\ H_{uy} &= \frac{\partial^2 F}{\partial u \partial y} + \sum_{i=1}^{n_y} \lambda_i \left(\frac{\partial^2 C_i}{\partial u \partial y} \right). \end{aligned} \tag{4.9}$$

On the other hand, the traditional sequential quadratic programming (SQP) solves the following QP subproblem:

$$\begin{aligned} \underset{\Delta x}{\text{minimize}} \quad & g^T \Delta x + \frac{1}{2} \Delta x^T H \Delta x, \\ \text{subject to} \quad & C(x) + A\Delta x = 0, \end{aligned} \tag{4.10}$$

where

$$H := \begin{bmatrix} H_{yy} & H_{yu} \\ H_{uy} & H_{uu} \end{bmatrix}, \qquad x := \begin{pmatrix} y \\ u \end{pmatrix}, \qquad g := \frac{\partial F}{\partial x}, \qquad A := \frac{\partial C}{\partial x} \tag{4.11}$$

whose first order necessary optimality condition is the same as (4.8). This shows that the traditional SQP method is equivalent to the standard Newton method applied to (4.7) with some multiplier estimates λ for λ^*. This also implies that if the Newton method applied to the optimality conditions (4.7) converges, then the traditional SQP method converges and vice versa. Since the Newton method has quadratic local convergence if it converges, the traditional SQP method will converge quadratically if the current point is near a solution. Of course, global convergence is not guaranteed for the standard Newton method if no special treatment is used (e.g., in the linesearch or trust region method). The linesearch method is explained in Chapter 6. There are two different ways of solving SAND: reduced SAND, full SAND. Reduced SAND is described in Section 4.2.1 and full SAND in Section 4.2.2

4.2.1 Reduced SAND

Reduced SAND reduces the number of variables and solves a smaller system of equations for each QP subproblem. There are two ways: the reduced space method and the range space method. The reduced space method (or the null space method) decompose the search direction Δx in (4.10) into the null and range space of the Jacobian of C (i.e. A in (4.10)). That is,

$$\Delta x = Z\Delta z + Y\Delta y, \tag{4.12}$$

where the columns of Z form the basis for the null space of A (i.e. $AZ = 0$), while the columns of Y form the basis for the range space of A. Substituting this decomposition into (4.10),

$$\begin{aligned} \underset{\Delta z, \Delta y}{\text{minimize}} \quad & g_z^T \Delta z + \Delta z^T (Z^T H Y) \Delta y + \frac{1}{2} \Delta z^T (Z^T H Z) \Delta z + \frac{1}{2} \Delta y^T (Y^T H Y) \Delta y + g_y^T \Delta y, \\ \text{subject to} \quad & C(x) + AY \Delta y = 0, \end{aligned} \tag{4.13}$$

where $g_z = Z^T g$ and $g_y = Y^T g$. $AZ\Delta z$ in the constraint vanishes because $AZ = 0$. Note that there is no Δz in the constraint. Thus, one can solve for Δy directly from the constraints. Then, the QP above becomes equivalent to the following unconstrained minimization problem:

$$\underset{\Delta z}{\text{minimize}} \quad g_z^T \Delta z + \Delta z^T (Z^T H Y) \Delta y + \frac{1}{2} \Delta z^T (Z^T H Z) \Delta z, \tag{4.14}$$

where Δy is known. The minimizer of (4.14) can be found by solving the following equation:

$$(Z^T H Z) \Delta z = -(g_z + Z^T H Y \Delta y). \tag{4.15}$$

Once Δy and Δz are obtained, Δx is computed via (4.12) and x is updated to $x + \Delta x$. The multiplier estimate, λ, is computed via a reduced form of the adjoint equation, that is,

$$(AY)^T \lambda = -Y^T H \Delta x - g_y. \tag{4.16}$$

There are many choices for Z. One is to use the LQ factorization of A [37]. The LQ factorization transforms the matrix A into a lower triangular matrix by applying a series of orthonormal matrices from the left (e.g., the Givens rotation or the Householder reflection). That is,

$$AQ = \begin{bmatrix} L & 0 \end{bmatrix}. \tag{4.17}$$

Let n_y be the rank of the range space of A, and n_z the rank of the null space of A. From the LQ factorization, the first n_y columns of Q can be set to be Y and the rest of columns to be Z. One advantage of this approach is that the condition number of $Z^T H Z$ is not bigger than the condition number of H because the condition number of Z is 1. This approach may be very efficient for a certain type of problem, such as a problem with a small number of constraints, linear programming, or an active set method, where one row of A is changing at a time. However, if the number of constraints or state variables are large, then it is impractical to use LQ factorization. This is the usual case for PDE-constrained optimization. As the discretization of PDE becomes finer, the number of constraints becomes larger and it is impractical to form an LQ factorization. Furthermore, for the case of nonlinear PDE constraints, in which A is changing at every major iteration, it is not clear how to update the previous LQ factorization efficiently. Thus, one usually

uses the following null space matrix:

$$Z = \begin{bmatrix} -J^{-1}Q \\ I \end{bmatrix}, \tag{4.18}$$

where $J = \frac{\partial C}{\partial y}$ and $Q = \frac{\partial C}{\partial u}$. Note that indeed $AZ = 0$. Although this approach may increase the condition number of $Z^T HZ$, it is readily available because J^{-1} is an operator used in both linear and nonlinear PDE solvers. The usual choice for the Y matrix is

$$Y = \begin{bmatrix} I \\ 0 \end{bmatrix}. \tag{4.19}$$

With this particular Z and Y, problem (4.13) becomes

$$\begin{aligned} \underset{\Delta z, \Delta y}{\text{minimize}} \quad & g_z^T \Delta z + \Delta z^T (Z^T HY)\Delta y + \frac{1}{2}\Delta z^T (Z^T HZ)\Delta z + \frac{1}{2}\Delta y^T H_{yy}\Delta y + g_y^T \Delta y, \\ \text{subject to} \quad & C(x) + J\Delta y = 0. \end{aligned} \tag{4.20}$$

Again, the constraint is independent of Δz. Thus Δy can be directly solved solely with the constraint. Given Δy, Δz is obtained via (4.15) with

$$\begin{aligned} g_z &= -Q^T J^{-T} g_y + g_u, \\ Z^T HY\Delta y &= -Q^T J^{-T} H_{yy}\Delta y + H_{uy}\Delta y, \end{aligned} \tag{4.21}$$

where g_y and g_u are state and control parts of the gradient. Then the Δx is updated via (4.12). The multiplier estimate λ is obtained via

$$J^T \lambda = -H_{yy}\Delta x_y - H_{yu}\Delta u - g_y. \tag{4.22}$$

Note that no approximation is introduced in the process of solving the QP subproblem (4.20). The major computation occurs when forming $Z^T HZ$ and solving for it in (4.15). This computation can be efficiently approximated by a quasi-Newton update [37]. The BFGS update approximates the effective Hessian (e.g., the second order information $Z^T HZ$) with the rate of change in gradients. The update is done so that each update gives rank-two update in the resultant matrix. Although the BFGS method is very attractive since it only requires the matrix-vector multiplications, it may have a storage issue. In order to resolve this, the limited memory BFGS is developed, in which only a fixed number of first derivatives are used [13].

Many reduced space (e.g., rSQP) methods introduce some kind of approximation other than BFGS approximation. For example, when Δz is computed via (4.15), the second term of the right hand side (i.e. $Z^T HY\Delta y$) may be ignored. Also, when the multiplier estimates are updated via

(4.22), the first two terms of the right hand side (i.e. $-H_{yy}\Delta x_y - H_{yu}\Delta u$) may be ignored [9]. If those terms are not neglected and the QP (4.20) is solved using the BFGS update, then the operators J^{-1} or J^{-T} need to be applied four times for each major iteration. On the other hand, if those terms are omitted and the QP (4.20) is solved with the BFGS update, then the operators J^{-1} or J^{-T} need to be applied three times for each major iteration.

The second type of reduced SAND is the range space method. The range space method solves for the multiplier estimates first then compute optimization variables. The necessary optimality condition for (4.10) is as follows:

$$\begin{aligned} H\Delta x + A^T\lambda &= -g \\ A\Delta x &= -c. \end{aligned} \tag{4.23}$$

Eliminating Δx, one can solve the following system of equations in order to obtain λ:

$$S\lambda = -c + AH^{-1}g, \tag{4.24}$$

where $S = -AH^{-1}A^T$ is the Schur complement. The search direction Δx is updated by solving the following system of equations:

$$H\Delta x = -g - A^T\lambda. \tag{4.25}$$

In order to solve (4.24), one can form the Schur complement S and apply some factorization, which is very expensive. Instead of forming S, one can solve for A and A^T sequentially. In order to solve for (4.25), one needs to solve for H, which can be done either by a direct or iterative method.

The null space and range space methods are related to two different factorizations of the KKT system. The null space method is related to the following factorization:

$$\begin{bmatrix} H_{yy} & H_{yu} & J^T \\ H_{uy} & H_{uu} & Q^T \\ J & Q & \end{bmatrix} = \begin{bmatrix} H_{yy}J^{-1} & 0 & I \\ H_{uy}J^{-1} & I & Q^TJ^{-T} \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} J & Q & 0 \\ 0 & H_z & 0 \\ 0 & H_{yu} - H_{yy}J^{-1}Q & J^T \end{bmatrix}, \tag{4.26}$$

where $H_z = Q^TJ^{-T}H_{yy}J^{-1}Q - Q^TJ^{-T}H_{yu} - H_{uy}J^{-1}Q + H_{uu}$ is the Schur complement of control variable. On the other hand, the range space method is related to the following factorization:

$$\begin{bmatrix} H & C^T \\ C & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ CH^{-1} & I \end{bmatrix} \begin{bmatrix} H & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & H^{-1}C^T \\ 0 & I \end{bmatrix}. \tag{4.27}$$

These factorizations are used in developing preconditioners introduced in Section 5.2

4.2.2 Full SAND

A KKT system (e.g., equations (4.8)), which appears in the PDE-constrained optimal control is sparse, but tends to be very large when the discretization is made finer. The strategy of full SAND is simple. It attempts to solve the KKT system simultaneously. Due to the advances in numerical linear algebra, it is possible to solve a large sparse linear system of equations, $Ax = b$. There are two ways of solving the large sparse linear system of equations: the direct and the iterative method. The iterative method is studied because the iterative method has many benefits although there are many efficient direct sparse solvers. The equations (4.8) can be expressed in matrix form, that is,

$$\begin{bmatrix} H & A^T \\ A & \end{bmatrix} \begin{pmatrix} \Delta x \\ \lambda \end{pmatrix} = \begin{pmatrix} -g \\ -C(y,u) \end{pmatrix}. \tag{4.28}$$

Note that the KKT matrix above is indefinite. It can be shown that if $\ker(H) \cap \ker(A) = \{0\}$, then the KKT matrix is invertible [7]. Thus, one cannot apply any iterative method, such as the conjugate gradient (CG) method, which is only applicable for symmetric positive definite matrices. However, many iterative methods can handle a symmetric indefinite matrix. Several iterative methods are listed in Table 5.1 with explanation of what types of matrices can be handled for each method. For example, (4.8) is solved using an iterative method, such as MINRES or GMRES. In order to apply any iterative method, a good preconditioner is needed. Many preconditioners are available for the saddle point system of equations. Currently existing preconditioners and some newly developed preconditioners are presented in Chapter 5.

Chapter 5

Iterative methods and preconditioners

5.1 Iterative methods

There are two main approaches to solve a system of linear equations, $Ax = b$: direct methods and iterative methods, and a variation of the former can be used to assist the latter. Direct methods attempt to solve the system directly. For example, they factorize the matrix A into several matrices for which linear systems are easy to solve. Those matrices include orthogonal, diagonal, lower, and upper triangular matrices. The direct methods give an exact solution with exact arithmetic. Even with rounding errors, their behavior is well studied and understood, so they are robust and predictable. However, for a large system of equations, a direct method may not be practical due to storage or computational efficiency issues. This has become more obvious recently when the size of the problem has become bigger (e.g., 3D PDE discretization). In order to overcome these shortcomings of direct methods, two main developments have been made. First, various direct methods that conserve sparsity as much as possible were developed (e.g., MA27, MUMPS, PARDISO, SPOOLES, superLU, sparse Cholesky factorization, etc.). These direct methods are now also used in developing a preconditioner in iterative methods. Second, the combination of Krylov subspace and preconditioning led to a group of iterative methods that outperform direct methods on very large sparse systems of equations. Such iterative methods include conjugate gradient (CG) [46], MINRES, SYMMLQ [63], and generalized minimum residual (GMRES) methods [74]. Choi has organized such iterative methods inclusively in her thesis [15]. Table 5.1 reproduces Choi's table and adds one more iterative method that was developed recently by Fong and Saunders.

As shown in the table, CG is applicable to symmetric positive definite matrices, while MINRES and SYMMLQ are applicable to symmetric indefinite matrices. GMRES is applicable to square

Linear Equations	Authors	Properties of A	A^T?
CG	Hestenes and Stiefel (1952) [46]	Symmetric positive definite	
CRAIG	Faddeev and Faddeeva (1963) [23]	Square or rectangular	yes
MINRES	Paige and Saunders (1975) [63]	Symmetric indefinite	
SYMMLQ	Paige and Saunders (1975) [63]	Symmetric indefinite	
Bi-CG	Fletcher (1976) [28]	Square unsymmetric	yes
LSQR	Paige and Saunders (1982) [65, 64]	Square or rectangular	yes
GMRES	Saad and Schultz (1986) [74]	Square unsymmetric	
CGS	Sonneveld (1989) [78]	Square unsymmetric	
QMR	Freund and Nachtigal (1991) [34]	Square unsymmetric	yes
Bi-CGSTAB	Van der Vorst (1992) [18]	Square unsymmetric	yes
TFQMR	Freund (1993) [33]	Square unsymmetric	yes
SQMR	Freund and Nachtigal (1994) [35]	Symmetric	yes
CGLS	Hestenes and Stiefel (1952) [46]	Square or rectangular	yes
RRLS	Chen (1975) [14]	Square or rectangular	yes
RRLSQR	Paige and Saunders (1982) [65]	Square or rectangular	yes
LSQR	Paige and Saunders (1982) [65]	Square or rectangular	yes
LSMR	Fong and Saunders (2011) [29]	Square or rectangular	yes

Table 5.1: Existing iterative algorithms since CG was created in 1952.

unsymmetric matrices. CG, MINRES, and SYMMLQ have the attractive property of short term recurrence while GMRES does not have that property. However, because some preconditioners (e.g., constraint preconditioners) introduced in this thesis do not preserve the symmetry of the KKT system, neither MINRES nor SYMMLQ will work, but GMRES will. This is why GMRES is used here, and as explained in this chapter[1]. However, first, some classical iterative methods and their main ideas are briefly illustrated. In particular, Jacobi scaling is explained in more detail because it will be used to develop a new preconditioner for the KKT system that appears in PDE-constrained optimization problems.

5.1.1 Classical iterative methods and Jacobi scaling

Iterative methods for solving linear equations generate a sequence of approximate solutions that converge to a real solution. The classical iterative methods include Jacobi, Gauss-Seidel, SOR, and Chebyshev semi-iterative [40]. The main idea of these standard iterations is best explained as splitting. Assume that the matrix A is split into M and N, such that $A = M - N$, where M is nonsingular. Then, $Ax = b$ can be rewritten as $Mx = Nx + b$. By introducing a sequence of x^k, $Mx^{k+1} = Nx^k + b$ can be used as an updating rule in the hope of achieving convergence, that is, $x^k \to x^*$, where $x^* = A^{-1}b$. For example, Jacobi iteration uses M as the diagonal matrix whose diagonal elements are the same as the ones in A, while Gauss-Seidel iteration employs the lower

[1]Several other constraint preconditioners have recently been developed for use with CG, especially for saddle point systems, but are not in investigated in this thesis [30, 80, 19].

triangular part of A as M. Convergence is guaranteed if the spectral radius of $M^{-1}N$ is less than 1. Furthermore, Gauss-Seidel iteration converges for any starting point x^0 if the matrix A is symmetric positive definite [40].

The diagonal elements in Jacobi iteration can also be used in scaling the system of equations. Jacobi scaling is often used as preconditioning. That is, for a symmetric matrix A, instead of solving $Ax = b$ directly, one can solve $DADy = Db$, and the solution x^* is obtained from $x^* = Dy$, where D is the diagonal matrix whose diagonals are over the square root of the diagonal elements of A. The underlying motivation for this alternative is that DAD is more well-conditioned than the original matrix A (e.g., $\kappa(DAD) \ll \kappa(A)$). Additionally, van der Sluis in his 1969 paper [83] has proved that Jacobi scaling gives near-optimal conditioning within a factor of n among all diagonal scaling matrices for a symmetric positive definite matrix A. In the same paper, he also proved that if A has maximum q non-zeros in a row, then Jacobi scaling gives near-optimal condition number to within a factor of q. Even earlier, in 1955, Forsythe showed that if a symmetric positive definite matrix A has Young's property A (i.e. there is a permutation matrix P such that PAP^T is a 2×2 block matrix whose diagonal block matrices are diagonal), then Jacobi scaling provides the optimal conditioning among all diagonal scaling matrices. The idea of diagonal scaling as a preconditioning is studied and further conceptualized to bi-normalization and equilibration [11, 51, 53]. The idea of diagonal scaling will be used in a later section in developing a novel preconditioner for PDE-constrained optimization problems.

5.1.2 GMRES

Given a matrix A and a vector q_1, the Krylov subspace $K(A,q_1,k)$ is defined as

$$K(A,q_1,k) = \text{span}\{q_1, Aq_1, \ldots, A^{k-1}q_1\}. \tag{5.1}$$

A Krylov subspace iterative method seeks to generate a sequence of approximate solutions $\{x_k\}$ to $Ax = b$, where $x_k \in x_0 + K(A,r_0,k)$ and $r_0 = b - Ax_0$. GMRES is a Krylov subspace method that works for unsymmetric $Ax = b$ problems. It can be thought of as a generalization of MINRES, in which x_k minimizes the 2-norm of the residual (i.e., $\|b - Ax\|_2$) for a symmetric indefinite matrix. The Lanczos method is used in order to get the orthonormal basis for the Krylov subspace in MINRES, while GMRES uses the Arnoldi method for orthogonalization in order to handle unsymmetric systems. Instead of the tridiagonal matrix produced in MINRES, the Arnoldi process produces an upper Hessenberg matrix. Thus, the three-term recurrence property MINRES possesses is lost in GMRES. The Arnoldi process up to the k-th iteration is shown in Algorithm 1. Note that the Arnoldi process is nothing more than applying the Gram-Schmidt algorithm to the Krylov subspace

$K(A,q_1,k)$ with basis vectors of $q_1, Aq_1, \ldots, A^{k-1}q_1$.

Algorithm 1: Arnoldi Iteration

Choose an initial vector q_1 with $\|q_1\|_2 = 1$.

for $j = 1 \ldots k$ **do**

$r_j = Aq_j$,

$h_{i,j} = (r_j,q_i)$ for $i = 1, \ldots, j$,

$r_j = r_j - \sum_{i=1}^{j} h_{i,j} q_i$,

$h_{j+1,j} = \|r_j\|_2$, and

$q_{j+1} = r_j / h_{j+1,j}$.

end

Let an upper Hessenberg matrix H_k be defined as,

$$H_k = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & \cdots & h_{1,k} \\ h_{2,1} & h_{2,2} & \cdots & \cdots & h_{2,k} \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & h_{k,k-1} & h_{k,k} \end{bmatrix}. \tag{5.2}$$

Then the original matrix A is related to H_k through the Arnoldi process as

$$AQ_k = Q_k H_k + r_k e_k^T, \tag{5.3}$$

where $Q_k = [q_1 \cdots q_k]$. The last operation in the **for** loop in the k-th iteration of Arnoldi iteration 1 gives $r_k = h_{k+1,k} q_{k+1}$. Substituting this into relation (5.3),

$$\begin{aligned} AQ_k &= Q_k H_k + q_{k+1}(h_{k+1,k} e_k^T), \\ &= Q_{k+1} \hat{H}_k, \end{aligned} \tag{5.4}$$

where $Q_{k+1} = [q_1 \ldots q_{k+1}]$ and $\hat{H}_k$ is defined as

$$\hat{H}_k = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & \cdots & h_{1,k} \\ h_{2,1} & h_{2,2} & \cdots & \cdots & h_{2,k} \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & h_{k,k-1} & h_{k,k} \\ 0 & \cdots & \cdots & 0 & h_{k+1,k} \end{bmatrix}. \tag{5.5}$$

GMRES takes q_1 to be a normalized initial residual for a given initial guess x_0, that is, $q_1 = \frac{r_0}{\|r_0\|_2}$.

For some vector $y_k \in \mathbb{R}^k$, the kth approximate solution is $x_k = x_0 + Q_k y_k$. Then the residual at the k-th iteration becomes

$$\begin{aligned} r_k = b - Ax_k &= b - A(x_0 + Q_k y_k) \\ &= r_0 - AQ_k y_k \\ &= r_0 - Q_{k+1}\hat{H}_k y_k \\ &= Q_{k+1}(\|r_0\|_2 e_1 - \hat{H}_k y_k). \end{aligned} \tag{5.6}$$

Since the orthonormal matrix is invariant in 2-norm, the residual norm becomes $\|r_k\|_2 = \|\|r_0\|_2 e_1 - \hat{H}_k y_k\|_2$. GMRES finds y_k^* such that $\|\|r_0\|_2 e_1 - \hat{H}_k y_k\|_2$ is minimized. This is done by applying a sequence of Givens rotations in order to transform $\hat{H}_k$ into an R_k upper triangular matrix augmented with an extra zero row. That is, after a series of Givens rotations G, the structure of the residual norm is

$$\|G\|r_0\|_2 e_1 - R_k y_k\|_2 = \left\| \begin{pmatrix} f_k \\ \varphi_{k+1} \end{pmatrix} - \begin{bmatrix} \times & \times & \cdots & \cdots & \times \\ 0 & \times & \cdots & \cdots & \times \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \times & \times \\ 0 & \cdots & \cdots & 0 & \times \\ 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} y_k \right\|_2, \tag{5.7}$$

where s_i for $i = 1, \ldots, k$ is a *sine* operation from the i-th Givens rotation. The least-squares solution is obtained by solving $R_k y = f_k$, and the corresponding residual is $\varphi_{k+1} = |s_1 s_2 \ldots s_k \|r_0\|_2|$. Since the *sine* function is strictly less than 1 unless $h_{k+1,k}$ is exactly zero (i.e., lucky break-down), the residual is monotonically decreasing. The monotonic decrease in residual can be also seen from the fact that the previous Krylov subspace is included in the current subspace, that is, $K(A,r_0,k-1) \subset K(A,r_0,k)$. Thus, GMRES can be terminated if $\varphi_{k+1} = |s_1 s_2 \ldots s_k \|r_0\|_2|$ is small enough. Then, a solution is obtained by $x^* = x_0 + Q_k y_k$.

The convergence of GMRES for a symmetric matrix can be derived easily. The residual vector r_k is $b - Ax_k$. For simplicity, let x_0 be a zero vector. Since $x_k \in K(A,b,k)$, GMRES finds x_k such that

$$\|r_k\|_2 = \min_{\alpha \in \mathbb{R}^k} \left\| \sum_{j=1}^{k} \left(\alpha_j A^j - I\right) b \right\|_2. \tag{5.8}$$

Let $\wp_k$ be a set of polynomials with degree k or less. Then (5.8) can be rewritten as

$$\|r_k\|_2 = \min_{p \in \wp_k, p(0)=1} \|p(A)b\|_2. \tag{5.9}$$

Since A is symmetric, it is diagonalizable. That is, $A = VDV^T$, where D is a diagonal matrix with diagonal elements of eigenvalues of A, and V is an orthogonal matrix whose columns are eigenvectors

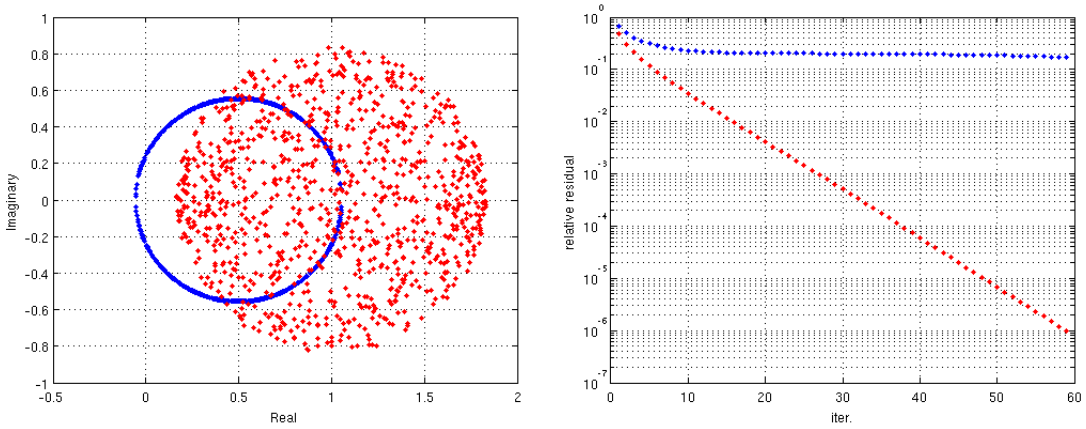


Figure 5.1: **Left**: eigenvalue distributions of two matrices in the complex plane. The red dots have c/r ratio of 1.2 while the blue dots have $c/r = 0.9$. **Right**: GMRES performance (e.g., relative residual) for two matrices from the left figure are shown. The red dots correspond to the matrix whose eigenvalue distribution is represented by the red dots in the left figure.

of A. Then (5.9) can be bounded above by

$$\begin{aligned} \|r_k\|_2 &= \min_{p\in\wp_k, p(0)=1} \|Vp(D)V^T b\|_2, \\ &\leqslant \kappa(V)\|b\|_2 \min_{p\in\wp_k, p(0)=1} \max_{\lambda\in\sigma(A)} p(\lambda), \end{aligned} \tag{5.10}$$

where $\sigma(A)$ is a set of eigenvalues of A. One can bound this min max by Chebyshev polynomials. Precisely speaking, if all the eigenvalues of A are included in the ellipse $E(c,d,a)$, centered at c, focal distance d, semi-major axis a, and excluding the origin, then the residual norm $\|r_k\|_2$ is bounded by

$$\|r_k\|_2 \leqslant \kappa(V)\|b\|_2 \frac{C_k(a/d)}{C_k(c/d)}. \tag{5.11}$$

Furthermore, the term $\frac{C_k(a/d)}{C_k(c/d)}$ can be approximated by (e.g., see the Chapter 6 of [76])

$$\frac{C_k(a/d)}{C_k(c/d)} \approx \left(\frac{a+\sqrt{a^2-d^2}}{c+\sqrt{c^2-d^2}}\right)^k. \tag{5.12}$$

Note that if the centroid of a cluster of eigenvalues of A is far away from the origin (e.g., c big), the cluster size is small (e.g., a and d small), and the condition number of V is small (e.g., $\kappa(V)$ is small), then GMRES converges quickly. Indeed, the performance of GMRES can be characterized by the c/r ratio when the eigenvalues are included in a circle centered at c with radius r. On the left of Figure 5.1, eigenvalue distributions are shown for two different matrices whose size is 1000×1000

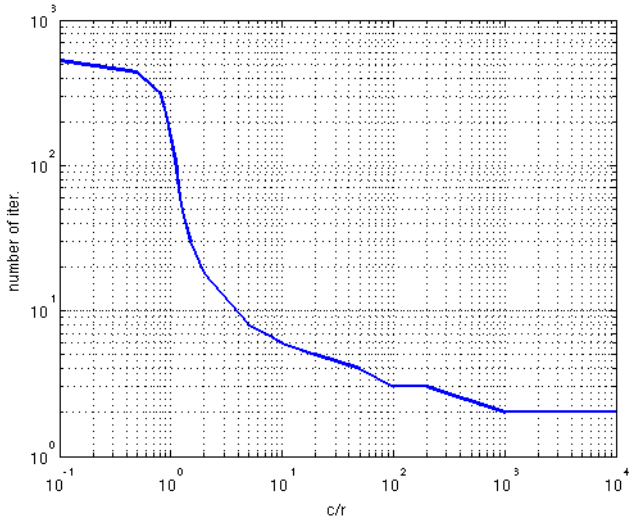


Figure 5.2: A study of c/r ratio versus the number of iterations in GMRES.

in the complex plane. The red dots are included in a circle centered at (1,0) with a radius of about 0.83, so the c/r ratio is 1.2. On the other hand, the eigenvalue distribution of the blue dots lies on the circumference centered at (0.5,0) with a radius of about 0.56, so it has a c/r ratio of 0.9. In order to avoid a (nearly) singular matrix, the matrix whose c/r ratio is less than 1 has eigenvalues that lie only on the circumference. On the right of Figure 5.1, the relative residuals (e.g., $\|r\|_2/\|b\|_2$) computed by applying GMRES to these two matrices are shown for the first 60 iterations. As expected, the matrix with higher c/r ratio converges faster than the one with smaller c/r. This pattern appears true. Figure 5.2 shows a plot of c/r ratio versus the number of iterations taken to converge. As expected, GMRES converges faster for bigger c/r ratio. In general, GMRES converges fast for any symmetric positive definite matrix whose eigenvalues are clustered. It is important to note that the convergence analysis for symmetric matrices presented in this section is not only applicable to GMRES, but also to any Krylov subspace-based iterative method. Also note that if there are m distinct nonzero eigenvalues in a symmetric matrix A, then theoretically GMRES must converge within at most m iterations. This is because if A has only one nonzero eigenvalue λ_0, then there is a linear function that vanishes at λ_0, resulting in zero residual in the first iteration due to the inequality in (5.10). If there are two distinct nonzero eigenvalues λ_0 and λ_1, then one can find a quadratic function that vanishes at both λ_0 and λ_1, resulting in zero residual in the second iteration due to the inequality in (5.10).

5.2 Preconditioners

As seen in the previous section, GMRES (or other Krylov iterative methods) perform best when the eigenvalues of a system are clustered. If the matrix does not have clustered eigenvalues, then one can apply preconditioners and try to make the preconditioned system have clustered eigenvalues. Because it is natural to think of saddle point systems in block structures, only block preconditioners are considered. However, it must be noted that incomplete factorization preconditioners are also possible for saddle point systems. There are three types of block preconditioners: block diagonal, block constraint, and block triangular. The subsequent sections will describe each type of preconditioner and introduce some new preconditioners for a saddle point system of the following block structure:

$$\begin{bmatrix} V & & J^T \\ & \phi G & Q^T \\ J & Q & \end{bmatrix} \begin{pmatrix} y \\ u \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}, \qquad (Ax = b), \tag{5.13}$$

which is identical to the block structures for all four types of optimal control formulation presented in Chapter 3.

5.2.1 Block diagonal

Murphy, et al. [59] present a short note on block diagonal preconditioner based on the Schur complement for a saddle point system. That is,

$$P_{bd} := \begin{bmatrix} V & & \\ & \phi G & \\ & & -S \end{bmatrix}, \tag{5.14}$$

where the Schur complement S is given by

$$S = -(JV^{-1}J^T + \frac{1}{\phi} QG^{-1}Q^T). \tag{5.15}$$

The preconditioned system has three distinct nonzero eigenvalues, so a Krylov iterative method is theoretically expected to converge in at most three iterations. The only disadvantage of applying this preconditioner is that it requires solving for the Schur complement, which is impractical since it is as expensive as solving for original saddle point system.

Rees, et al. [71] developed a block diagonal preconditioner that is based on an approximate Schur complement. There are two parts in the Schur complement: $S_1 := -JV^{-1}J^T$ and $S_2 := -\frac{1}{\phi}QG^{-1}Q^T$. V and G are symmetric positive definite, and it may be possible to make them easier to invert (i.e., turn them into a diagonal matrix) since V and G only appear in the objective function and their two main roles are defining a norm and weighting each component of state vector

proportional to volume occupation. J is the Jacobian of constraints with respect to the state variables. It is a stiffness matrix for static PDEs and a time integrator scheme for dynamic PDEs (see Chapter 3). Although there are cases where J becomes rank-deficient, only the full-rank case is considered. Thus, the first part of the Schur complement has full rank. However, except for some special cases (i.e., optimal body force control for static PDEs), Q is not even a square matrix. Thus, the second part of the Schur complement is usually rank-deficient. Note that ϕ determines the relative significance of the two parts. For ϕ small, the second part is more important than the first part, and vice versa for ϕ large. Rees et al. take advantage of these facts in order to develop preconditioners. Note that for ϕ not too small, S can be approximated by S_1, which defines a block diagonal preconditioner,

$$P_{rees} := \begin{bmatrix} V & & \\ & \phi G & \\ & & -S_1 \end{bmatrix}. \tag{5.16}$$

Rees et al. further approximate S_1 by replacing J by some approximation $\bar{J}$ that is obtained by a multi-grid method. Defining $\bar{S} := -\bar{J}V^{-1}\bar{J}^T$, Rees's block diagonal preconditioner $\bar{P}_{rees}$ is

$$\bar{P}_{rees} := \begin{bmatrix} V & & \\ & \phi G & \\ & & -\bar{S} \end{bmatrix}. \tag{5.17}$$

They have shown in [71] that an eigenvalue λ of the preconditioned system $P_{rees}^{-1}A$ satisfies one of the following:

$$\begin{aligned} \lambda &= 1, \\ \frac{1}{2}(1+\sqrt{1+4\sigma_1}) \leqslant \lambda &\leqslant \frac{1}{2}(1+\sqrt{1+4\sigma_m}), \\ \frac{1}{2}(1-\sqrt{1+4\sigma_m}) \leqslant \lambda &\leqslant \frac{1}{2}(1-\sqrt{1+4\sigma_1}), \end{aligned} \tag{5.18}$$

where $0 \leqslant \sigma_1 \leqslant \ldots \leqslant \sigma_m$ are the eigenvalues of $\frac{1}{\phi}(JV^{-1}J^T)^{-1}(QG^{-1}Q^T) + I$. Note that for ϕ extremely big, $\sigma_i \simeq 1$ for $i = 1, \ldots, m$, which results in eigenvalues identical to those of Murphy's preconditioned system [59]. This analysis explains why P_{rees} performs better for ϕ large (see P_{rees} in Table 7.2).

One can go to the other extreme by replacing S by S_2 in P_{bd}, defining $P_{bd2} = blkdiag(V,\phi G,S_2)$. This must work well for ϕ small because S_2 is more important than S_1 when ϕ small. Indeed, P_{bd2} works better for smaller ϕ as shown in Table 7.2 and Figure A.1 (i.e., case 3 in Appendix A.4). However, as pointed out earlier, the problem with this preconditioner is that S_2 has full-rank only if Q is a square full rank matrix. Q is not a square matrix if the number of control variables is not equal to that of state variables and for dynamic problems in general.

There is a simpler way of approximating S if ϕ is small because the order of magnitude of S will be determined by $\frac{1}{\phi}$. One can replace S with ξI, where I is the identity matrix and ξ is a coefficient. This defines the preconditioner $P_{sbd2} = blkdiag(V,\phi G,\xi I)$. ξ must be set so that it is similar to S_2 in magnitude. To be precise, ξ must be bigger than the maximum absolute eigenvalue of the Schur complement S. This choice is appropriate even when G is not a square matrix. Unfortunately, this preconditioner does not work well because the preconditioned system becomes nearly singular as ϕ decreases. For example, if one sets $\xi = \frac{1}{\phi}$, then the preconditioned system has an eigenvalue of $\frac{1-\sqrt{1+4\mu}}{2}$, where μ is an eigenvalue of S. Note that it becomes nearly zero when μ is nearly zero. Figure A.3 shows the performance of GMRES applied to contrived example 1 with preconditioner P_{sbd2} and $\phi = 10^{-6}$. The spectral analysis for all the block diagonal preconditioners introduced in this section is done in Appendix A.3.

Murphy's block diagonal preconditioner may be thought of as being derived from the factorization (4.27) of the KKT system. Ignoring H^{-1} in the first and third matrices and putting minus in front of S, the resultant matrix becomes a block diagonal preconditioner identical to Murphy's. Depending on how S is approximated, one obtains several other practical preconditioners, including Rees's preconditioner and all other block diagonal preconditioners introduced in this section. The concept of approximating or modifying the factorization of the KKT system in order to derive a new preconditioner will be used again later in this chapter.

5.2.2 Block lower triangular

The factorization of the KKT system (4.27) can be used to find another type of preconditioner: the block lower triangular preconditioner. Note that the multiplication of the first two factors gives

$$P_{lt} = \begin{bmatrix} V & 0 & 0 \\ 0 & \phi G & 0 \\ J & Q & S \end{bmatrix}, \tag{5.19}$$

where S is the Schur complement. Note that from factorization (4.27), the preconditioned system $P_{lt}^{-1}A$ is a block upper triangular matrix in which the diagonal elements are 1s. Thus, it has a unique eigenvalue, 1. However, as in the case of P_{bd}, this is not practical because P_{lt} needs to solve for S, which is as expensive as solving the original KKT system directly. Thus, one needs to replace S with approximate S. As in the block diagonal preconditioners, for ϕ relatively large, one can replace S with $S_1 = -JV^{-1}J^T$, which defines the preconditioner P_{lt1},

$$P_{lt1} = \begin{bmatrix} V & 0 & 0 \\ 0 & \phi G & 0 \\ J & Q & S_1 \end{bmatrix}. \tag{5.20}$$

One can prove the following theorem

Theorem 1. *Let μ be an eigenvalue of $(JV^{-1}J^T)^{-1}QG^{-1}Q^T$. Then, the eigenvalues of $P_{lt1}^{-1}A$ include 1 with multiplicity of $n+m$ and $1+\frac{\mu}{\phi}$, where ϕ is a regularization parameter.*

The theorem states that if ϕ is larger than $|\mu|$, then all the eigenvalues are between 0 and 2. The upper right of Figure A.4 shows the eigenvalue distribution of $P_{lt1}^{-1}A$ for $\phi = 10^{-2}$. All the real parts of the eigenvalues are equal to or larger than 1. The lower right of Figure A.4 shows the convergence of GMRES with preconditioner P_{lt1} and $\phi = 10^{-2}$. Although P_{lt1} is no better than P_{lt}, the number of iterations required for convergence is much lower than the number of iterations required for no preconditioners. However, for a small regularization parameter ϕ, the performance of P_{lt1} is poor. The left two Figures of A.4 show eigenvalue distribution and convergence of GMRES for $\phi = 10^{-6}$. The eigenvalues are not clustered and GMRES converges poorly. This is because S_1 does not represent the Schur complement S well if ϕ is small. An alternative way is to replace S with $S_2 = -\frac{1}{\phi}QG^{-1}G^T$ as in the case of block diagonal preconditioners. This defines P_{lt2}:

$$P_{lt2} = \begin{bmatrix} V & 0 & 0 \\ 0 & \phi G & 0 \\ J & Q & S_2 \end{bmatrix}. \tag{5.21}$$

Note that there is no minus sign in front of S_2, unlike block diagonal preconditioners. This is because of the preference of signs of eigenvalues of preconditioned system.

Theorem 2. *Let μ be an eigenvalue of $(JV^{-1}J^T)^{-1}QG^{-1}Q^T$. Then, the eigenvalues of $P_{lt2}^{-1}A$ include 1 with multiplicity of $n+m$ and $1+\frac{\phi}{\mu}$, where ϕ is a regularization parameter.*

The theorem states that if ϕ is much smaller than μ, then all the eigenvalues of $P_{lt2}^{-1}A$ are clustered around 1. The upper left of Figure A.4 shows the eigenvalue distribution of $P_{lt2}^{-1}A$ for $\phi = 10^{-6}$. Note that they are indeed clustered around 1. The lower left of Figure A.4 shows the convergence of GMRES for P_{lt2}. Although P_{lt2} performs worse than P_{lt}, the number of iterations required for convergence is around 20, which is much less than for GMRES with P_{lt1} or no preconditioners. The spectral analysis of the preconditioned systems is shown in Appendix A.4, where both Theorems 1 and 2 are proved.

5.2.3 Range space

If the second derivatives of the PDE constraints are not included in the Hessian, then it is straightforward to invert H in the factorization (4.27). Taking advantage of this fact, one can modify the Schur complement in the middle factor and use it as a preconditioner. For large ϕ, one can replace

S with S_1, which defines P_{rs1}:

$$P_{rs1} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ JV^{-1} & \frac{1}{\phi}QG^{-1} & I \end{bmatrix} \begin{bmatrix} V & 0 & J^T \\ 0 & \phi G & Q^T \\ 0 & 0 & S_1 \end{bmatrix}. \tag{5.22}$$

Applying P_{rs1} is equivalent to solving for S_1 instead of S in (4.24) of the range space method. For small ϕ, one can replace S with S_2, which defines P_{rs2}:

$$P_{rs2} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ JV^{-1} & \frac{1}{\phi}QG^{-1} & I \end{bmatrix} \begin{bmatrix} V & 0 & J^T \\ 0 & \phi G & Q^T \\ 0 & 0 & S_2 \end{bmatrix}. \tag{5.23}$$

Applying P_{rs2} is equivalent to solving for S_2 instead of S in (4.24) of the range space method.

5.2.4 Constraint preconditioner

Biros and Ghattas

Biros and Ghattas [9] use factorization (4.26) of the KKT system, which is based on the reduced space method. Based on this factorization, they introduce four preconditioners by approximating H_z with BFGS, omitting second-order terms H, or replacing J by its preconditioner $\tilde{J}$. Two of four preconditioners introduced by Biros and Ghattas are

$$P_{bg1} = \begin{bmatrix} H_{yy}\tilde{J}^{-1} & 0 & I \\ H_{uy}\tilde{J}^{-1} & I & Q^T\tilde{J}^{-T} \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{J} & Q & 0 \\ 0 & B_z & 0 \\ 0 & H_{yu} - H_{yy}\tilde{J}^{-1}Q & \tilde{J}^T \end{bmatrix}, \tag{5.24}$$

$$P_{bg2} = \begin{bmatrix} 0 & 0 & I \\ 0 & I & Q^T\tilde{J}^{-T} \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{J} & Q & 0 \\ 0 & B_z & 0 \\ 0 & 0 & \tilde{J}^T \end{bmatrix} = \begin{bmatrix} 0 & 0 & \tilde{J}^T \\ 0 & B_z & Q \\ \tilde{J} & Q & \end{bmatrix}, \tag{5.25}$$

where B_z is the BFGS approximation of $H_z = Q^TJ^{-T}H_{yy}J^{-1}Q - Q^TJ^{-T}H_{yu} - H_{uy}J^{-1}Q + H_{uu}$. For linear problems or the initial Hessian approximation for nonlinear problem, they implemented two choices: the identity matrix and the two-step stationary method. In this thesis, a variant of P_{bg2} is considered where B_z is replaced with the H_{uu} block (i.e., ϕG):

$$P_{vbg} = \begin{bmatrix} 0 & 0 & \tilde{J}^T \\ 0 & \phi G & Q \\ \tilde{J} & Q & \end{bmatrix}. \tag{5.26}$$

One can prove the following theorem

Theorem 3. *Let μ be an eigenvalue of $(JV^{-1}J^T)^{-1}QG^{-1}Q^T$. Then, the eigenvalues of $P_{vbg}^{-1}A$ include 1 with multiplicity of $n+m$ and $1+\frac{\mu}{\phi}$, where ϕ is a regularization parameter.*

The spectral analysis of the preconditioned system is shown in Appendix A.2. Note that the eigenvalues are identical to those of $P_{lt1}^{-1}A$. Thus, P_{lt1} and P_{vbg} are supposed to perform similarly. This approach of developing preconditioners is based on approximating or modifying a block factorization, as for the block diagonal preconditioners in Section 5.2.1.

Rees' constraint preconditioner

Rees et al. [71], introduced a constraint preconditioner

$$P_{crees} = \begin{bmatrix} & & J^T \\ & -\phi S_1 & Q^T \\ J & Q & \end{bmatrix}. \tag{5.27}$$

Zero-regularization constraint preconditioner

If the block matrix Q is a square full-rank matrix, then there is an ideal preconditioner for ϕ small. With $\phi = 0$, the KKT matrix (5.13) defines the zero-regularization constraint preconditioner

$$P_{zrc} = \begin{bmatrix} V & & J^T \\ & & Q^T \\ J & Q & \end{bmatrix}. \tag{5.28}$$

P_{zrc} is a block triangular matrix, so one can solve for Q^T first, then V, and last Q in order to apply it. This preconditioner works best when ϕ small. However, this desirable property is marred by the fact that this preconditioner is only applicable when Q is square and nonsingular.

Theorem 4. *Let μ be an eigenvalue of $(JV^{-1}J^T)^{-1}QG^{-1}Q^T$. Then, the eigenvalues of $P_{zrc}^{-1}A$ include 1 with multiplicity of $2n$ and $1+\frac{\phi}{\mu}$, where ϕ is a regularization parameter.*

The spectral analysis of the preconditioned system is shown in Appendix A.1.

5.2.5 GMRES performance comparison

The example below compares the performance of GMRES with several preconditioners introduced in this section.

Example 1. *A KKT matrix (5.13) is constructed with the number of state variables n and the number of control variables m. Each block matrix is generated randomly in MATLAB in a way that*

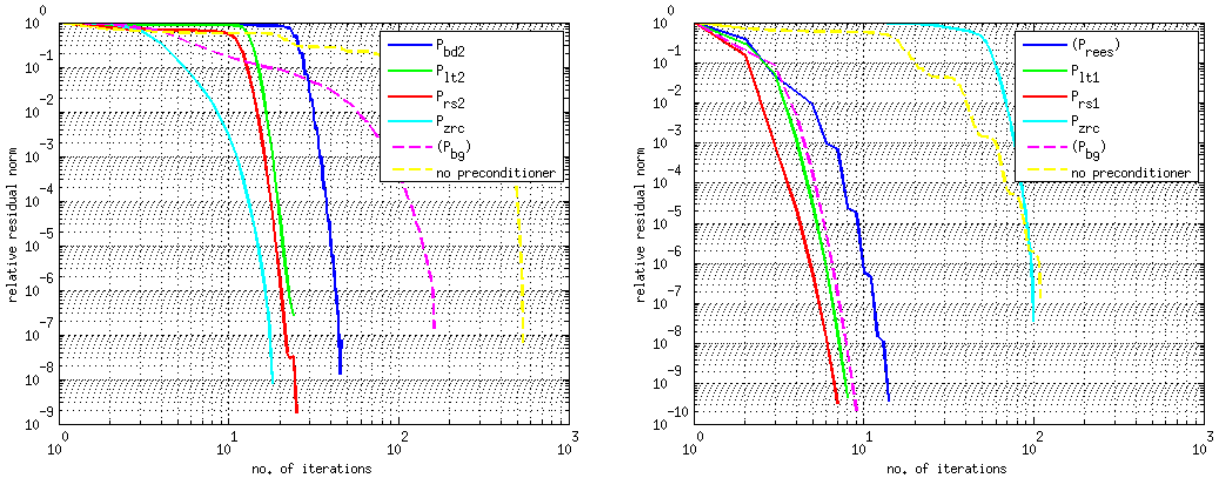


Figure 5.3: Comparison of the preconditioners introduced in this section. A KKT system of equations is generated as explained in Example 1 with $m = n = 200$. (a) **Left**: $\phi = 10^{-6}$. (b) **Right**: $\phi = 10^{-2}$.

makes it similar to a KKT matrix generated in PDE-constrained optimization based on the FEM discretization; $V \in \mathbb{R}^{n\times n}$ *and* $G \in \mathbb{R}^{m\times m}$ *are randomly generated diagonal symmetric positive definite matrices,* $J \in \mathbb{R}^{n\times n}$ *is a randomly generated symmetric positive definite matrix with bandwidth of* 5*, and* $Q \in \mathbb{R}^{n\times m}$ *is a random sparse matrix generated by the* ***sprand*** *function in MATLAB with density of* 0.2*. The right-hand side* b *is generated by the* ***rand*** *function in MATLAB. Finally, all the convergence results are averages of 100 samples.*

Figure 5.3 compares all the preconditioners introduced in this section by solving the KKT system of equations in example 1 with $m = n = 200$. The figure on the left shows the results for $\phi = 10^{-6}$ and the one on the right shows the results for $\phi = 10^{-2}$. For $\phi = 10^{-6}$, the constraint preconditioner P_{zrc} works the best. Among the preconditioners derived from factorization (4.27) (i.e., P_{bd2}, P_{lt2}, and P_{rs2}), P_{rs2} is slightly better than P_{lt2} and P_{bd2} is the worst. However, all three preconditioners perform better than the constraint preconditioner P_{bg} and the case where no preconditioner is applied. For $\phi = 10^{-2}$, the preconditioner P_{rs1} based on the range-space method works the best. Lower triangular preconditioner P_{lt1} and constraint preconditioner P_{bg} perform similarly as second best and Rees's block diagonal preconditioner P_{rees} is next. Figure 5.4 compares all the preconditioners introduced in this section by solving the KKT system of equations in example 1 with $n = 500$ and $m = 100$. The figure on the left shows the results for $\phi = 10^{-6}$ and the one on the right shows the results for $\phi = 10^{-2}$. For $\phi = 10^{-6}$, the range-space preconditioner P_{rs2} works the best. For $\phi = 10^{-2}$, the preconditioner P_{rs1} based on the range-space method works the best. Lower triangular preconditioner P_{lt1} and constraint preconditioner P_{bg} perform similarly as second best and Rees's block diagonal preconditioner P_{rees} is next.

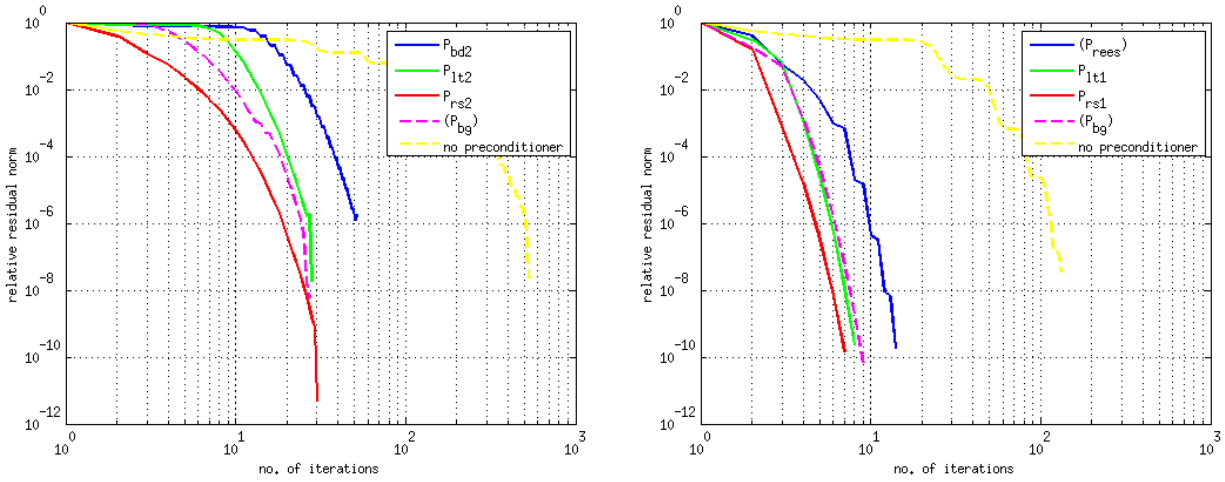


Figure 5.4: Comparison of the preconditioners introduced in this section. A KKT system of equations is generated as explained in example 1 with $m = 100$, $n = 500$. (a) **Left**: $\phi = 10^{-6}$. (b) **Right**: $\phi = 10^{-2}$.

5.2.6 An exact representation of a Schur complement

Pearson and Wathen [66] introduced a new approximation for the Schur complement of the Laplacian optimal control problem with volume controls. The KKT system corresponding to the problem is

$$\begin{bmatrix} V & 0 & J \\ 0 & \phi V & -V \\ J & -V & 0 \end{bmatrix} \begin{pmatrix} y \\ u \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}, \tag{5.29}$$

where J and V are symmetric matrices. The Schur complement corresponding to this KKT system is $S = JV^{-1}J + \frac{1}{\phi}V$. They approximated S with

$$S_{pearson} = (J + \frac{1}{\sqrt{\phi}}V)V^{-1}(J + \frac{1}{\sqrt{\phi}}V). \tag{5.30}$$

This approximation works well because the error is $O(\frac{1}{\sqrt{\phi}})$ (note that $S = S_{pearson} + \frac{2}{\sqrt{\phi}}J$, which shows that the error term is $\frac{2}{\sqrt{\phi}}J$). Also it is easy to solve with $J + \frac{1}{\sqrt{\phi}}V$ via a multi-grid method. Pearson and Wathen showed that if the block diagonal preconditioner with this approximation of the Schur complement is used, a Krylov iterative method will converge in a fixed number of iterations regardless of mesh size or ϕ value.

It is possible to derive the following exact representation of the Schur complement S:

$$S_{exact} = (J - \frac{1}{\sqrt{-\phi}}V)V^{-1}(J + \frac{1}{\sqrt{-\phi}}V). \tag{5.31}$$

Note that the structure of S_{exact} is the same as that of $S_{pearson}$ (i.e., the linear combination of J and V in the first factor, V^{-1} in the second factor, and again the linear combination of J and V in the third factor). A difference between S_{exact} and $S_{pearson}$ is that S_{exact} includes imaginary numbers. For example, in order to solve with $(J + \frac{1}{\sqrt{-\phi}}V)$, one needs to use a linear solver that can solve with a symmetric complex matrix[2]. Such solvers include CG [32], FETI [26], and multi-grid methods [52]. In Section 7.1.1, the expression S_{exact} is used in the range space method via FETI-DP in order to solve a linear static thermal optimal control problem. There, numerical results show that FETI-DP on $(J - \frac{1}{\sqrt{-\phi}}V)$ and $(J + \frac{1}{\sqrt{-\phi}}V)$ does not depend on the mesh size h as long as the size of subdomain H changes to keep the ratio H/h fixed. The numerical results also show that FETI-DP on $(J - \frac{1}{\sqrt{-\phi}}V)$ and $(J + \frac{1}{\sqrt{-\phi}}V)$ does depend on ϕ, but the dependence is minor.

5.3 Multi-precondition

The preconditioners introduced in this section have different ranges of regularization parameters in which they work best. If there is a way of combining two preconditioners (e.g., one works best for small regularization parameter, the other for bigger regularization parameter), then this may result in a combination of preconditioners that is robust in the sense that it works for any value of regularization. The robustness in linear solvers is more crucial in nonlinear problems than in linear problems. In nonlinear problems, a sequence of linearized problems is solved (e.g., by Newton's method) and each linearized problem varies in its characteristics. Thus it is important in nonlinear problems to have a linear solver that can handle a broad range of linear systems and we try to achieve robustness by introducing a multi-preconditioned Krylov iterative method.

The idea of mixing preconditioners in order to develop robust iterative methods goes back to 1993, the year in which Saad developed flexible GMRES (FGMRES) [75, 76]. Since then, many variants of FGMRES have been developed. A few of them are flexible conjugate gradient (FCG) [61], multi-preconditioned CG (MPCG) [12], and multi-preconditioned GMRES (MPGMRES) [41], all of which fall into the category of Krylov iterative methods. Both FCG and MPCG lose an attractive property of CG (or PCG), namely, short-term recurrence. Similarly, both GMRES and MPGMRES do not have the short-term recurrence property. However, GMRES and MPGMRES work for unsymmetric preconditioned systems. Since the preconditioners P_{zrc}, P_{lt1} and P_{lt2} yield unsymmetric preconditioned systems, we focus on MPGMRES.

5.3.1 Multi-preconditioned GMRES

MPGMRES [41] is similar to flexible GMRES, the only difference being that MPGMRES has a richer subspace than flexible GMRES. There are two main categories of MPGMRES: full and truncated MPGMRES. Both are based on the block Arnoldi process. Full MPGMRES has a richer subspace

[2]Neither $(J + \frac{1}{\sqrt{-\phi}}V)$ nor $(J - \frac{1}{\sqrt{-\phi}}V)$ is Hermitian.

than truncated MPGMRES at each iteration, but full MPGMRES is computationally expensive if it does not converge early.

Algorithm 2: MPGMRES: vectorized version with elimination of redundant search directions

Choose $\mathbf{x}^{(0)}$, $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
$\beta = \|\mathbf{r}^{(0)}\|$, $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/\beta$
$Z^{(1)} = \textbf{fullmultiprecondition}(\mathbf{r}^{(0)})$
$\tilde{V} = \mathbf{v}^{(1)}$
$n_{Z^{(1)}}$ = no. of columns in $Z^{(1)}$, $n_Z = Z_{Z^{(1)}}$
for $i = 1, \ldots$ **until** *convergence* **do**
  $n_{\tilde{V}}$ = no. of columns in $\tilde{V}$
  for $\ell = 1 \ldots n_{Z^{(i)}}$ **do**
    $\mathbf{w} = AZ_\ell^{(i)}$
    for $j = 1 \ldots n_{\tilde{V}}$ **do**
      $(\tilde{H}_i)_{j,n_{\tilde{V}}} = \mathbf{w}^T \tilde{V}_j$
      $\mathbf{w} = \mathbf{w} - \tilde{V}_j (\tilde{H}_i)_{j,n_{\tilde{V}}}$
    end
    $(\tilde{H}_i)_{n_{\tilde{V}}+1,n_{\tilde{V}}} = \|\mathbf{w}\|_2$
    $\tilde{V} = \left[\tilde{V}\ \mathbf{w}/(\tilde{H}_i)_{n_{\tilde{V}}+1,n_{\tilde{V}}}\right]$
    if $(\tilde{H}_i)_{n_{\tilde{V}}+1,n_{\tilde{V}}} = 0$ **then**
      if *residual small enough* **then**
        lucky breakdown
      else
        remove this column of $Z^{(i)}, n_{Z^{(i)}} = n_{Z^{(i)}} - 1$
        return to top of loop
      end
    else
      $n_{\tilde{V}} = n_{\tilde{V}} + 1$
    end
  end
  $\mathbf{y}^{(i)} = \text{argmin}\|\beta\mathbf{e}_1 - \tilde{H}_i\mathbf{y}\|_2$
  $\mathbf{x}^{(i)} = \mathbf{x}^{(0)} + \left[Z^{(1)} \ldots Z^{(i)}\right]\mathbf{y}^{(i)}$
  $Z^{i+1} = \begin{cases} \textbf{fullmultiprecondition}(\tilde{V}_{n_Z+1:n_Z+n_{Z^{(i)}}}), \text{or} \\ \textbf{truncmultiprecondition}(\tilde{V}_{n_Z+1:n_Z+n_{Z^{(i)}}}) \end{cases}$
  $n_{Z^{(i+1)}}$ = no. of columns in $Z^{(i+1)}$, $n_Z = n_Z + n_{Z^{(i)}}$
end

MPGMRES is described in Algorithm 2. The columns in $Z = \left[Z^{(1)} \ldots Z^{(i)}\right]$ form basis vectors for $x^{(i)} - x^{(0)}$. Full MPGMRES calls the function **fullmultiprecondition** to update $Z^{(i)}$ in the main loop, while truncated MPGMRES calls the function **truncmultiprecondition**. Note that the size of the number of columns in $Z^{(i)}$ increases exponentially in full MPGMRES as the number of outer iterations increases. On the other hand, truncated MPGMRES has a fixed number of columns in $Z^{(i)}$ added to Z for each outer iteration. Due to the way Z is formed, it is possible to get linearly

Algorithm 3: Subroutine: full multi-preconditoined step

function $Z =$ **fullmultiprecondition**(V)
$Z = [P_1^{-1}V \dots P_t^{-1}V]$
end function

Algorithm 4: Subroutine: truncated multi-preconditoined step

function $Z =$ **truncmultiprecondition**(V)
$Z = [P_1^{-1}V_1 \dots P_t^{-1}V_t]$
end function

dependent columns in Z. If this happens, $(\tilde{H}_i)_{n_{\tilde{V}}+1,n_{\tilde{V}}} = 0$ although it has not yet converged. To prevent this undesirable event, a check is made for linear dependency of every column in Z. At the i-th iteration of the Arnoldi process, the following relation is true: For $Z_i = [Z^{(1)} \dots Z^{(i)}]$ and $V_{i+1} = [V^{(1)} \dots V^{(i+1)}]$,

$$AZ_i = V_{i+1}\tilde{H}_i,$$

where $\tilde{H}_i$ is a matrix with one extra row at the bottom of an upper Hessenberg matrix. This relation is directly used in MPGMRES to minimize $\|\mathbf{b} - A\mathbf{x}\|_2$ in order to get updated $\mathbf{y}^{(i)}$ in the following sense:

$$\|\mathbf{b} - A\mathbf{x}\|_2 = \|\beta\mathbf{v}_1 - AZ_i\mathbf{y}\|_2 = \|V_{i+1}[\beta\mathbf{e}_1 - \tilde{H}_i\mathbf{y}]\|_2 = \|\beta\mathbf{e}_1 - \tilde{H}_i\mathbf{y}\|_2.$$

In order to see the difference between MPGMRES and FGMRES, let us look at the preconditioned Krylov subspace in the first couple of iterations of MPGMRES with two right preconditioners and FGMRES with preconditioners P_1 and P_2. At the first iteration, FGMRES seeks the solution $\mathbf{x}^{(1)}$ that minimizes $\|\mathbf{b} - A\mathbf{x}^{(1)}\|_2$ subject to

$$\mathbf{x}^{(1)} \in \text{Span}\{P_1^{-1}\mathbf{r}_0\}.$$

At the second iteration, it seeks the solution $\mathbf{x}^{(2)}$ that minimizes $\|\mathbf{b} - A\mathbf{x}^{(2)}\|_2$ subject to

$$\mathbf{x}^{(2)} \in \text{Span}\{P_1^{-1}\mathbf{r}_0, P_2^{-1}AP_1^{-1}\mathbf{r}_0\}.$$

At the third iteration,

$$\mathbf{x}^{(3)} \in \text{Span}\{P_1^{-1}\mathbf{r}_0, P_2^{-1}AP_1^{-1}\mathbf{r}_0, P_3^{-1}AP_2^{-1}AP_1^{-1}\mathbf{r}_0\}.$$

On the other hand, MPGMRES seeks the solution $\mathbf{x}^{(1)}$ that minimizes $\|\mathbf{b} - A\mathbf{x}^{(1)}\|_2$ subject to

$$\mathbf{x}^{(1)} \in \text{Span}\{P_1^{-1}\mathbf{r}_0, P_2^{-1}\mathbf{r}_0\}.$$

At the second iteration, it seeks the solution $\mathbf{x}^{(2)}$ that minimizes $\|\mathbf{b} - A\mathbf{x}^{(2)}\|_2$ subject to

$$\mathbf{x}^{(2)} \in \text{Span}\{P_1^{-1}\mathbf{r}_0, P_2^{-1}\mathbf{r}_0, P_1^{-1}AP_1^{-1}\mathbf{r}_0, P_1^{-1}AP_2^{-1}\mathbf{r}_0, P_2^{-1}AP_1^{-1}\mathbf{r}_0, P_2^{-1}AP_2^{-1}\mathbf{r}_0, \}.$$

As mentioned earlier, MPGMRES has a richer subspace than FGMRES at each major iteration. If $\mathbf{x}^*$ is a solution of $A\mathbf{x}^* = \mathbf{b}$, then convergence (i.e., $\|\mathbf{x}^{(i)} - \mathbf{x}^*\| \to 0$ as $i \to \infty$) is expected. Let us define $S_{(i)}$ to be a subspace in which $x_{(i)}$ resides. If convergence occurs for small i, this means that $\mathbf{x}^* \in S_{(i)}$ with i small is "almost" true. "Almost" means that convergence will still occur even with $\mathbf{x}^* \notin S_{(i)}$ if a large convergence threshold is used. For a more detailed description of MPGMRES, see the paper by Grief, et al. [41].

The robustness of MPGMRES becomes apparent in nonlinear dynamic optimal control problems. A combination of one preconditioner that works well for a small regularization parameter (e.g., P_{lt2}) and another preconditioner that works well for a large regularization parameter (e.g., P_{vbg}) in MPGMRES turns out to be useful in nonlinear dynamic problems (see Sections 7.4.1, 7.4.2 and 7.4.3).

Chapter 6

Global convergence of sequential quadratic programming

Newton-type methods do not guarantee global convergence if a starting point is not close to a solution and if the method is not safeguarded. One safeguard that is necessary for global convergence is explained. It is the linesearch method. An important concept in the linesearch method is a merit function, which is also explained in this chapter.

6.1 Linesearch

In a linesearch based SQP method, a search direction p is determined by solving a QP subproblem taking a step α along p. That is, the next point x_{k+1} is updated from the current point x_k through the formula

$$x_{k+1} = x_k + \alpha_k p.$$

There are many criteria that determine a "good" step length α_k. One obvious criterion is the decrease in some functional M that quantifies both an objective function and a constraint violation. That is,

$$M(x_{k+1}) < M(x_k). \tag{6.1}$$

M is called a merit function. Various linesearch terminating criteria and merit functions are explained.

6.1.1 Terminating criteria

A simple decrease in M is not sufficient to guarantee global convergence. For example, consider $M(x) = x^2 - 1$ and assume that we aim to find a minimum of M, which is $M(x^*) = -1$. If one

takes a series of step lengths α_k that make the merit function $M(x_k)$ $\frac{5}{k}$ at the k-th iteration, then $M(x_{k+1}) < M(x_k)$ is satisfied at every iteration, but $M(x_k)$ converges to 0, not to the minimum -1. In order to guarantee global convergence, stricter terminating conditions (e.g., the Wolfe condition) need to be imposed in a linesearch procedure. In each iteration, a sufficient decrease is required. Clearly the decrease in M will be converging to zero. However, what we require is that the decrease is not converging to zero with respect to decrease that is possible.

To understand these stricter conditions, a univariate function $\rho(\alpha)$ needs to be defined for simplicity as follows:

$$\rho(\alpha) = M(x_k + \alpha p).$$

The Wolfe condition imposes two conditions, *the Armijo condition* [4] and *the curvature condition*. The Armijo condition is measured by the following inequality:

$$M(x_k + \alpha p_k) \leqslant M(x_k) + c_1 \alpha \nabla M_k^T p_k \tag{6.2}$$

for some constant $c_1 \in (0,1)$. In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$. The Armijo condition is also called the "sufficient decrease" condition because it imposes the decrease (i.e., $M(x_k) - M(x_k + \alpha p_k)$) in a merit function to be greater than a positive lower bound (i.e., $-c_1 \alpha \nabla M_k^T p_k$). The Armijo condition is not sufficient by itself for a reasonable progress because it is satisfied only for a sufficiently small value of α. To rule out an unacceptably short step, a second requirement, called the curvature condition, is necessary, in which α_k must satisfy the following inequality:

$$\nabla M(x_k + \alpha_k p_k)^T p_k \geqslant c_2 \nabla M_k^T p_k \tag{6.3}$$

for some constant $c_2 \in (c_1,1)$, where c_1 is a constant from (6.2). Note that the left-hand side is simply the derivative $\rho'(\alpha_k)$, so the curvature condition ensures that the slope of ρ at α_k is greater than c_2 times the initial slope $\rho'(0)$. This makes sense because if the slope $\rho'(\alpha)$ is strongly negative, we have an indication that we can reduce M significantly by moving further along a chosen direction. On the other hand, if $\rho'(\alpha_k)$ is only slightly negative or even positive, this is a sign that we cannot expect much more decrease in M in this direction, so it makes sense to terminate the linesearch. The curvature condition rules out an unacceptably short step. A typical value of c_2 is 0.9 when a search direction p_k is chosen by a Newton-type method.

Ensuring curvature condition (6.3) requires a gradient of a merit function at $x_k + \alpha_k p_k$, which may not be readily available. It is possible to construct a simple linesearch algorithm without the curvature condition and rule out a small step length. The backtracking approach checks if the Armijo condition (6.2) is satisfied or not starting with a sufficiently large α. The backtracking algorithm makes progress as follows:

Algorithm 5: Backtracking Linesearch

Choose $\hat{\alpha} > 0, \varphi \in (0,1), c \in (0,1)$; set $\alpha \leftarrow \hat{\alpha}$;
repeat until $f(x_k + \alpha p_k) \leqslant f(x_k) + c\alpha \nabla f_k^T p_k$
$\alpha \leftarrow \varphi\alpha$
end(repeat)
Terminate with $\alpha_k = \alpha$.

In this procedure, the initial step length $\hat{\alpha}$ is typically chosen to be 1 in a Newton-type method, but it can have a different value in other algorithms such as the steepest descent method. An acceptable step length will be found after a finite number of trials, because α_k will eventually become small enough for the Armijo condition to hold. In practice, the contraction factor φ is often allowed to vary at each iteration of the linesearch. For example, it can be chosen by some safeguarded interpolation. The only requirement on φ is $\varphi \in [\varphi_{lo}, \varphi_{hi}]$ for some fixed constants $0 < \varphi_{lo} < \varphi_{hi} < 1$.

The backtracking approach ensures either that the selected step length α_k is some fixed value (i.e., an initial choice $\hat{\alpha}$) or that it is short enough to satisfy the Armijo condition but not too short. The latter claim holds because an accepted value α is within a factor φ of the previous trial value (i.e., α_k/φ), which is rejected for violating the Armijo condition, that is, for being too long.

Besides the Wolfe condition and the backtracking linesearch, there are more linesearch terminating criteria. The strong Wolfe condition replaces the curvature condition in the Wolfe condition with the following stronger curvature condition:

$$|\nabla M(x_k + \alpha_k p_k)^T p_k| \leqslant c_2 |\nabla M_k^T p_k|. \tag{6.4}$$

As c_2 decreases to zero, condition (6.4) becomes closer to the minimization condition. It is harder to find a step that satisfies the strong Wolfe condition than the Wolfe condition, but there is an algorithm that finds a step satisfying the strong Wolfe condition [57]. Some modifications to the Wolfe condition can be found in [39] and [31].

In an SQP algorithm implemented in AERO-S, both the Wolfe condition and the backtracking algorithm are available. A user can choose one of these two algorithms from an input file. For more sophisticated linesearch algorithms, the textbooks [37, 60] are good references.

6.1.2 Merit functions

In an SQP method, constraints need not be satisfied exactly at an intermediate iterate. One should make sure that the SQP method creates a sequence of points that minimize both an objective function and infeasibilities. Thus, a merit function in a linesearch algorithm has to quantify a combination of an objective function and a constraint violation. Kaustuv [49] has made a table in which a list of a few

representative merit functions and their characteristics are shown. His table is repeated in Table 6.1.

Merit function	Exactness	Differentiability	Maratos Effects
$f(x) + \rho\|c(x)\|_1$	Exact	Non-differentiable at some point	Yes
$f(x) + \rho\|c(x)\|_2$	Exact	Non-differentiable at some point	Yes
$f(x) + \frac{\rho}{2}\|c(x)\|_2^2$	Inexact	Differentiable at all points	Yes
$f(x) - \lambda^T c(x) + \rho\|c(x)\|_2^2$	Inexact	Differentiable at all points	No

Table 6.1: A list of a few representative merit functions

In Table 6.1, an objective function is denoted as $f(x)$ and constraints as $c(x)$. The penalty parameter ρ is used to penalize infeasibility. The first merit function in Table 6.1 is the ℓ_1 penalty function. It has the desirable property of exactness [67], meaning that a minimizer of an original optimization problem is also a minimizer of the corresponding merit function with a sufficiently large penalty parameter. However, it is differentiable only if all the constraints are infeasible at a current point. Thus, one may not be able to apply a sophisticated surrogate model that can be useful to apply a sophisticated linesearch condition such as the Wolfe condition. Additionally, the ℓ_1 penalty merit function is known to exhibit Maratos effects [55], according to which a merit function hinders a unit step to be taken near a minimizer. If a Maratos effect occurs, a superlinear or quadratic rate of convergence for the quasi-Newton or Newton method is prohibited, which is undesirable. The second and third merit functions (the ℓ_2 penalty function and the quadratic penalty function, respectively) also suffer from the Maratos effect. On the other hand, the augmented Lagrangian merit function (i.e., the fourth merit function in Table 6.1) does not suffer from the Maratos effect. Actually, it has the exact opposite properties to the ℓ_1 or ℓ_2 penalty functions. For example, the augmented Lagrangian merit function does not have the exactness property, but it is differential everywhere. The augmented Lagrangian merit function has been used in SNOPT [38] successfully. In the PDE-constrained optimal control routine in AERO-S, all four merit functions in Table 6.1 are implemented.

Chapter 7

Numerical experiments

Optimal control has many practical engineering applications. Several numerical experiments of optimal control will be considered. Throughout the experiments, the performance of preconditioners introduced in Chapter 5 is examined. All the numerical results presented are based on simulations run on a single process except when the FETI-DP method is used in the range space method. Also, note that none of the optimal control examples considered consider a feedback system although the SAND formulation for nonlinear dynamic PDE-constrained optimal control in Chapter 3 can be readily adapted to the closed-loop control problem.

7.1 Linear static PDE-constrained optimal control

7.1.1 Linear static heat conduction with heat control

Thermal control is required in various applications. In computer hardware, one may want to maintain the temperature of the computer below some maximum temperature. In a greenhouse, a farmer may want a particular temperature change. For both of these cases, an appropriate cooling or heating system and its control are required. By solving a PDE-constrained optimal control problem, precise heat control can be obtained. In this section, heat conduction on a square plate is considered. Heat conduction is considered because its PDE is a Poisson's equation, which is linear.

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & F(y,u) = \frac{1}{2}(y-\bar{y})^T V(y-\hat{y}) + \frac{\phi}{2}u^T V u \\ \text{subject to} \quad & Ky + K_c y_c - Vu = 0, \end{aligned} \tag{7.1}$$

where $y \in \mathbb{R}^n$ is temperature, $u \in \mathbb{R}^m$ is distributed heat source, and V is the volume matrix. This example is identical to the example 5.1 in [71] by Rees et al. The domain Ω is $[0,1]^2 \subset \mathbb{R}^2$, which is a unit square plate, whose material properties are simple. For example, Young's Modulus of 1 Pa,

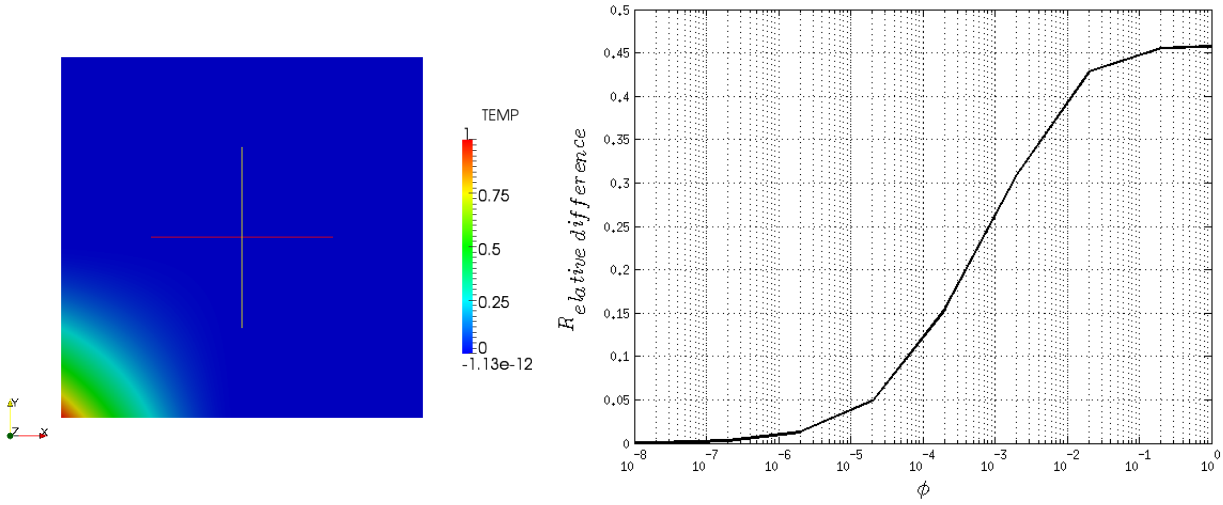


Figure 7.1: (a) **Left**: target temperature. **Right**: ϕ vs $\|y - \bar{y}\|/\|\bar{y}\|$

density of 1 kg/m^3, thickness of 1 m, specific heat coefficient of 1 $J/kg \cdot K$, and heat conduction coefficient of 1 $W/m \cdot K$ are used. The target temperature $\bar{y}$ is defined as

$$\bar{y} = \begin{cases} (2x_1 - 1)^2(2x_2 - 1)^2 & \text{if } (x_1,x_2) \in [0,\frac{1}{2}]^2, \\ 0 & \text{otherwise,} \end{cases} \tag{7.2}$$

which is illustrated in Figure 7.1(a). The boundary condition y_c is defined as

$$y_c = \bar{y} \quad \text{on } \partial\Omega = \{(x_1,x_2)|x_1 \in \{0,1\}, x_2 \in \{0,1\}\} \tag{7.3}$$

The optimal control (7.1) tries to find a temperature y that is close to the target temperature $\bar{y}$ by controlling heat u. How close y can be to $\bar{y}$ is determined by the regularization parameter ϕ. As ϕ decreases, y is expected to approach $\bar{y}$, but $\|u\|$ increases as ϕ decreases. This makes sense because the objective function value is not sensitive to the second term if the regularization parameter is small and the first term dominates the objective function value. These results are shown both in Figure 7.1(b) and Table 7.1.

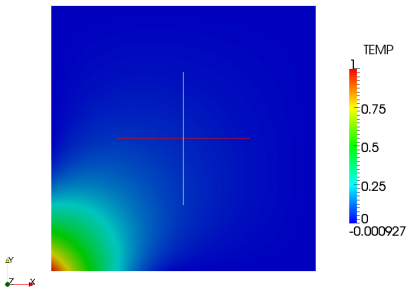
Temperature and heat distributions for various regularization parameters are shown in Figures 7.2(a)-(h). Note that for $\phi = 2$ and 2×10^{-2}, heat is almost zero everywhere and the corresponding temperature distributions (i.e., 7.2(a) and (c)) are slightly different from the target temperature (i.e., Figure 7.1). They are induced only by boundary condition y_c. The target temperature can be matched more precisely if smaller regularization is used. Figures 7.2(e) and (g) show temperature distributions that are closer to the target temperature. They are produced by setting

ϕ	$\|u\|$	$\|y-\bar{y}\|/\|\bar{y}\|$	$\|b-Ax\|/\|b\|$
2e-2	4.7e00	4.3e-01	2.75e-12
2e-3	2.6e01	3.1e-01	1.86e-11
2e-4	7.1e01	1.5e-01	4.80e-11
2e-5	1.2e02	4.9e-02	7.70e-11
2e-6	1.6e02	1.3e-02	4.99e-11
2e-7	1.8e02	3.2e-03	9.61e-11
2e-8	1.9e02	9.1e-04	9.23e-11
2e-10	1.9e02	1.6e-04	9.46e-11

Table 7.1: Study of objective function values of example 1 with respect to various ϕ. $\|y-\bar{y}\|/\|\bar{y}\|$ measures the first part of objective function, $\|u\|$ the second part. GMRES is used with a convergence tolerance of 10^{-10}. The mesh size h of 2^{-6} is used. The last column, $\|b-Ax\|/\|b\|$, presents the residual norm of the KKT system of equations, showing that the system has converged. $\|\cdot\|$ is ℓ_2 norm, A is the KKT matrix, and b the corresponding right-hand side.

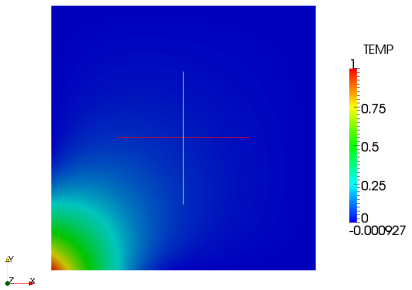
smaller ϕ values (i.e., $\phi = 2\times10^{-5}$ and 2×10^{-6}). Note that the corresponding heat distributions (i.e., Figures 7.2(f) and (h)) show nonzero heat at the left bottom of the domain. As ϕ decreases, a sharper heat gradient is present near the boundary.

According to Figure 7.1(b) (i.e., the graph of ϕ vs $\|y-\bar{y}\|/\|\bar{y}\|$), ideally, one would like to use a ϕ of less than 2×10^{-6} in order to match the target temperature to within 1%. However, as discussed in Chapter 5, each preconditioner has a certain range of ϕ in which it works the best. Tables 7.2 and 7.3 show the performance of seven preconditioners proposed in this thesis along with two preconditioners previously developed by others (i.e., P_{rees} and P_{bg2}). Table 7.2 shows the number of iterations whereas Table 7.3 shows the computational time in seconds according to various ϕs. nc means that GMRES does not converge within 100 restarts of 100 iterations. The mesh size (h) is 2^{-6} and the convergence threshold is 10^{-10}. For each fixed ϕ, the best performing preconditioner is colored as dark blue and the second best performing one as light blue.

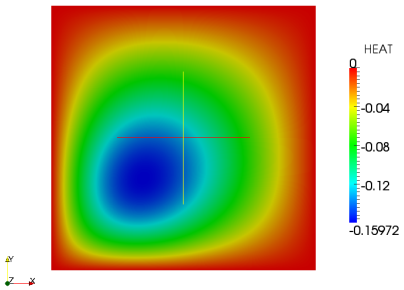


(a) temperature, $\phi = 2$

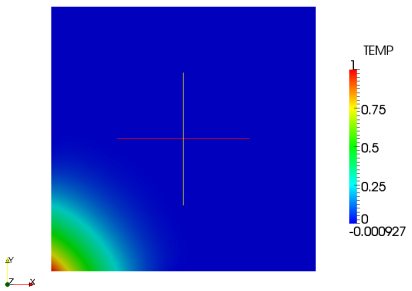
(b) heat, $\phi = 2$



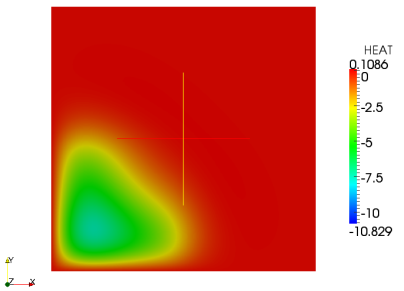
(c) temperature, $\phi = 2 \times 10^{-2}$



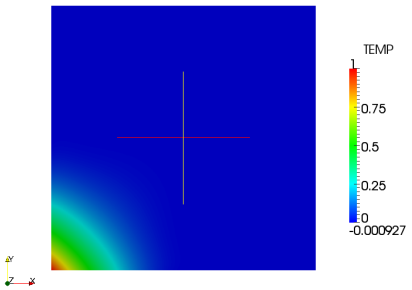
(d) heat, $\phi = 2 \times 10^{-2}$



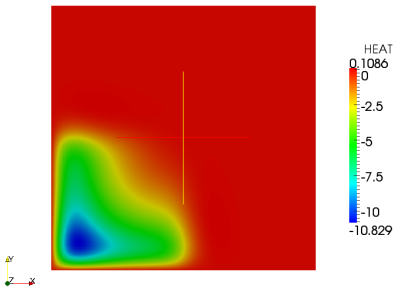
(e) temperature, $\phi = 2 \times 10^{-5}$



(f) heat, $\phi = 2 \times 10^{-5}$



(g) temperature, $\phi = 2 \times 10^{-6}$



(h) heat, $\phi = 2 \times 10^{-6}$

Figure 7.2: temperature and heat distribution for various regularization parameter

ϕ	(P_{rees})	(P_{bg2})	P_{vbg}	P_{zrc}	P_{lt2}	P_{lt1}	P_{bd2}	P_{rs1}	P_{rs2}
2E-2	9	7	6	nc	nc	6	nc	5	nc
2E-3	13	8	8	nc	nc	8	nc	7	nc
2E-4	21	12	10	9091	4607	11	8200	10	4561
2E-5	35	17	14	1365	848	18	2334	18	843
2E-6	66	30	23	176	227	33	476	31	226
2E-7	199	42	31	34	69	63	138	61	68
2E-8	2044	50	43	6	15	139	43	138	22
2E-9	4499	53	63	3	9	300	17	352	8
2E-10	nc	53	95	3	5	587	9	597	4
2E-11	nc	53	280	3	4	nc	9	3399	3
2E-12	nc	53	400	3	3	nc	9	-	2

Table 7.2: The numbers of iterations needed to converge are shown for various ϕ and preconditioners. GMRES is used as a solver. nc means that GMRES has not converged within 100 restarts of 100 iterations. The mesh size is 2^{-6} and the convergence threshold is 10^{-10}.

ϕ	(P_{rees})	(P_{bg2})	P_{vbg}	P_{zrc}	P_{lt2}	P_{lt1}	P_{bd2}	P_{rs1}	P_{rs2}	RSE	RSEfeti
2E-2	0.06	0.06	0.04	nc	nc	0.05	nc	0.06	nc	0.14	0.14
2E-3	0.07	0.06	0.06	nc	nc	0.06	nc	0.06	nc	0.14	0.12
2E-4	0.1	0.07	0.06	28.12	13.49	0.07	22.62	0.07	13.50	0.14	0.14
2E-5	0.14	0.09	0.07	3.93	2.45	0.09	6.40	0.11	2.70	0.14	0.14
2E-6	0.29	0.13	0.11	0.5	0.65	0.16	1.30	0.15	0.72	0.15	0.13
2E-7	0.93	0.18	0.13	0.08	0.19	0.30	0.37	0.30	0.20	0.13	0.13
2E-8	9.26	0.22	0.19	0.03	0.05	0.64	0.10	0.67	0.07	0.15	0.12
2E-9	20.57	0.23	0.27	0.02	0.04	1.37	0.06	1.74	0.03	0.14	0.14
2E-10	nc	0.22	0.45	0.03	0.03	2.81	0.03	3.00	0.04	0.13	0.13
2E-11	nc	0.23	1.26	0.03	0.02	nc	0.04	16.99	0.03	0.15	0.13
2E-12	nc	0.23	1.84	0.02	0.02	nc	0.04	nc	0.03	0.14	0.13

Table 7.3: Computational times in seconds are shown for various ϕ and preconditioners. GMRES is used as a solver. nc means that GMRES has not converged within 100 restarts of 100 iterations. The mesh size is 2^{-6} and the convergence threshold is 10^{-10}.

Block diagonal preconditioner P_{rees}, developed by Rees et al. [71] only works well for ϕ greater than or equal to 2×10^{-7} as expected from the discussion in Chapter 5. Constraint preconditioner P_{bg2} developed by Biros and Ghattas [9], where the reduced Hessian is replaced by an identity matrix, also tends to perform better for larger ϕ, but it performs better than P_{rees} for every ϕ. The variant of P_{bg2} (i.e., P_{vbg}) performs slightly better than P_{bg2} for a large ϕ, but the performance of P_{vbg} becomes worse than that of P_{bg2} as ϕ decreases. This shows that p_{vbg} is more sensitive to ϕ than P_{bg2}. Preconditioners P_{zrc}, P_{lt2}, P_{bd2}, and P_{rs2} are expected to work well for a small ϕ as discussed in Chapter 5. Indeed, Table 7.2 reflects this expectation. Also preconditioners P_{lt1} and P_{rs1} work better, the larger the value of ϕ.

Table 7.3 has two extra columns, RSE and RSEfeti. These columns show computational time

for the range space method with exact Schur complement factorization (5.31). RSE uses sparseLU factorization from the EIGEN[1] library in a single process in order to solve with the first and third factors in (5.31). Thus, computational time shown in the RSE column is used as a reference indicating whether or not a preconditioner is worth using. For example, if the computational time of a preconditioner is slower than one in the RSE column, then the corresponding computational time is colored red to indicate that the preconditioner performs worse than that in the range space method in a single process. This shows that preconditioners P_{zrc}, P_{lt2}, P_{bd2}, and P_{rs2} are worth using for a small ϕ whereas they are not worth using for a large ϕ. On the other hand, preconditioners P_{rees}, P_{bg2}, P_{vbg}, P_{lt1}, and P_{rs1} are worth using for a large ϕ, but not worth using for a small ϕ. RSEfeti uses FETI-DP [25] in multi-processes in order to solve with the first and third factors in (5.31). RSEfeti is expected to run a simulation faster than RSE, but since the problem size is not large, the effect of multi-processes does not appear in Table 7.3 (e.g., the size of the KKT system is 11,907).

In order to see the impact of multi-processes and to compare preconditioners on a larger problem, Tables 7.4 and 7.5 show the number of iterations and computational time on a problem whose mesh size is 2^{-8} and the size of the corresponding KKT system is 190,075. The same patterns are found. Preconditioners P_{zrc}, P_{lt2}, P_{bd2}, and P_{rs2} perform better the smaller the value of ϕ whereas preconditioners P_{rees}, P_{bg2}, P_{vbg}, P_{lt1}, and P_{rs1} performs better the larger the value of ϕ. RSEfeti works well throughout all the values of ϕ although the computational time increases slightly as ϕ decreases. FETI-DP is an iterative method, so one can report the number of FETI-DP iterations. Table 7.6 shows that the number of FETI-DP iterations on both $J - \frac{1}{\sqrt{-\phi}}V$ and $J + \frac{1}{\sqrt{-\phi}}V$ increases as ϕ decreases. This shows the obvious dependency of FETI-DP on ϕ.

ϕ	(P_{rees})	(P_{bg2})	P_{vbg}	P_{zrc}	P_{lt2}	P_{lt1}	P_{bd2}	P_{rs1}	P_{rs2}
2E-2	9	7	6	nc	nc	5	nc	4	nc
2E-3	13	9	8	nc	nc	7	nc	6	nc
2E-4	19	11	9	nc	nc	10	nc	9	nc
2E-5	33	16	14	nc	nc	17	nc	16	nc
2E-6	62	25	21	nc	nc	31	nc	30	nc
2E-7	189	31	22	476	1403	59	4302	57	1400
2E-8	1940	33	23	4	327	146	777	127	326
2E-9	5724	33	30	3	95	382	273	383	94
2E-10	nc	33	45	3	31	1291	59	1284	30
2E-11	nc	33	74	3	11	7049	21	7154	10
2E-12	nc	33	185	3	6	nc	11	nc	5

Table 7.4: The numbers of iterations to converge are shown for various ϕ and preconditioners. GMRES is used as a solver. nc means that GMRES has not converged within 100 restarts of 100 iterations. The mesh size is 2^{-8} and the convergence threshold is 10^{-10}.

Tables 7.7 and 7.8 compare preconditioners for various mesh size (h). Table 7.7 shows that the number of iterations increases as h decreases for P_{lt2}, P_{bd2}, and P_{rs2} as expected from the spectral

[1] http://eigen.tuxfamily.org/index.php?title=Main_Page

ϕ	(P_{rees})	(P_{bg2})	P_{vbg}	P_{zrc}	P_{lt2}	P_{lt1}	P_{bd2}	P_{rs1}	P_{rs2}	RSE	RSEfeti
2E-2	1.5	1.36	1.30	nc	nc	1.26	nc	1.21	nc	5.24	0.53
2E-3	1.76	1.49	1.44	nc	nc	1.39	nc	1.36	nc	5.22	0.75
2E-4	2.17	1.63	1.50	nc	nc	1.60	nc	1.57	nc	5.25	0.57
2E-5	3.27	1.98	1.83	nc	nc	2.24	nc	2.10	nc	5.24	0.59
2E-6	5.89	2.63	2.34	nc	nc	3.21	nc	3.27	nc	5.52	0.93
2E-7	18.16	3.08	2.41	25.07	75.09	5.45	216.76	5.87	82.16	5.26	0.78
2E-8	180.70	3.25	2.50	0.51	17.24	14.12	38.90	13.10	18.85	5.28	1.00
2E-9	531.41	3.25	3.01	0.50	5.33	37.00	13.59	39.14	5.73	5.31	1.19
2E-10	nc	3.23	4.18	0.52	1.40	124.33	2.63	129.85	1.52	5.33	1.37
2E-11	nc	3.23	7.11	0.51	0.72	706.28	0.95	722.12	0.74	5.38	1.31
2E-12	nc	3.25	17.52	0.50	0.58	nc	0.70	nc	0.57	13.99	1.30

Table 7.5: Computational time in seconds are shown for various ϕ and preconditioners. GMRES is used as a solver. nc means that GMRES has not converged within 100 restarts of 100 iterations. The mesh size is 2^{-8} and the convergence threshold is 10^{-10}. For RSEfeti, the size of the subdomain is 2^{-2} and the number of processes is 16.

analysis in Appendix A. On the other hand, the number of iterations for P_{rees}, P_{bg2}, P_{vbg}, P_{lt1}, and P_{rs1} does not depend on h. It is surprising to see that P_{zrc} also does not depend on h although the spectral analysis in Appendix A shows that it is supposed to depend on h. As in the previous numerical results, RSE is used as a reference. Results worse than the one for RSE for a fixed h are colored red. The best performing one is colored dark blue and the second best performing one is colored light blue. For large mesh sizes (e.g., $h = 2^{-4}$ or $h = 2^{-5}$), the computational times are so small for all the preconditioners that they are not comparable. As h decreases further, obvious performance differences start to appear. It is worthwhile noting that the full SQP method with preconditioners P_{bg2}, P_{vbg}, and P_{zrc} shows better scalability than the reduced SQP method (i.e., the range space method (RSE)).

Tables 7.9 and 7.10 present the scalability of FETI-DP on $J - \frac{1}{\sqrt{-\phi}}V$ and $J + \frac{1}{\sqrt{-\phi}}V$. Table 7.9 shows the number of FETI-DP iterations for various mesh sizes (h) with a fixed ratio of $H/h = 2^6$. The number of iterations does not depend on h. Table 7.10 shows the computational time for various numbers of processes. As the number of processes increases, the computational time decreases.

ϕ	$J - \frac{1}{\sqrt{-\phi}}V$	$J + \frac{1}{\sqrt{-\phi}}V$
2E-2	21	21
2E-3	21	22
2E-4	23	23
2E-5	29	29
2E-6	37	37
2E-7	47	47
2E-8	60	58
2E-9	75	74
2E-10	89	89
2E-11	93	93
2E-12	87	87

Table 7.6: Number of FETI-DP iterations to convergence is shown for various ϕ. The mesh size is 2^{-8} and the convergence threshold is 10^{-10}. The size of the subdomain is 2^{-2} and the number of processes is 16.

h (KKT size)	(P_{rees})	(P_{bg2})	P_{vbg}	P_{zrc}	P_{lt2}	P_{lt1}	P_{bd2}	P_{rs1}	P_{rs2}
2^{-4} (675)	88	41	38	5	5	45	9	49	4
2^{-5} (2,883)	699	55	52	5	9	82	15	87	8
2^{-6} (11,907)	2044	50	43	6	23	139	43	138	22
2^{-7} (48,387)	1942	41	31	4	82	151	199	139	81
2^{-8} (195,075)	1940	33	23	4	327	146	777	127	326
2^{-9} (783,363)	1354	26	19	5	1908	141	6430	113	1848
2^{-10} (3,139,587)	950	22	14	6	nc	136	nc	98	nc

Table 7.7: The number of iterations to convergence are shown for various mesh sizes h and preconditioners. The regularization parameter ϕ is 2×10^{-8}. The convergence threshold is set to be 10^{-10}.

h (KKT size)	(P_{rees})	(P_{bg2})	P_{vbg}	P_{zrc}	P_{lt2}	P_{lt1}	P_{bd2}	P_{rs1}	P_{rs2}	RSE
2^{-4}(675)	0.03	0.01	0.01	0	0	0.01	0	0.01	0	0.01
2^{-5}(2,883)	0.73	0.06	0.05	0.01	0.01	0.09	0.01	0.1	0.01	0.03
2^{-6}(11,907)	9.30	0.24	0.18	0.03	0.02	0.65	0.10	0.68	0.07	0.15
2^{-7}(48,387)	39.9	0.82	0.63	0.14	1.02	3.21	2.49	3.11	1.11	0.82
2^{-8}(195,075)	180.6	3.24	2.47	0.53	17.4	14.2	38.9	13.1	18.9	5.29
2^{-9}(783,363)	634.0	13.8	11.4	2.26	497.8	65.6	1594.6	57.7	528.8	37.71
2^{-10}(3,139,587)	2495.3	57.84	41.74	13.15	nc	356.3	nc	285.41	nc	276.8

Table 7.8: Computational time in seconds are shown for various mesh sizes h and preconditioners. The regularization parameter ϕ is 2×10^{-8}. The convergence threshold is set to be 10^{-10}.

num. of elem.	h	H	nsub	$J - \frac{1}{\sqrt{-\phi}}V$	$J + \frac{1}{\sqrt{-\phi}}V$
16,384	1/128	1/2	4	31	30
65,536	1/256	1/4	16	45	42
262,144	1/512	1/8	64	40	40
1,048,576	1/1024	1/16	256	35	35
1,638,400	1/1280	1/20	400	35	35
1,806,336	1/1344	1/21	441	32	32
2,166,784	1/1472	1/23	529	34	34
2,359,296	1/1536	1/24	576	34	34
2,560,000	1/1600	1/25	625	34	35
3,211,264	1/1792	1/28	784	34	34
3,368,400	1/1920	1/30	900	34	34

Table 7.9: The table shows how RSEfeti depends on the mesh size. The number of iterations is shown for various mesh sizes h and for the various subdomain sizes H. The fixed ratio $H/h = 2^6$ is used. The regularization parameter ϕ is 2×10^{-8}. The convergence threshold is set to be 10^{-10}.

num. of processes	comp. time (sec)
1	101.99
2	52.60
4	28.13
8	15.51
16	9.02
32	6.02
64	5.52
128	4.30
144	3.81

Table 7.10: The table shows how RSEfeti depends on the number of processes. The computational time is shown for the various number of processes. The mesh size $h = 1/1024$ and the subdomain size $H = 1/16$ are used. The regularization parameter ϕ is 2×10^{-8}. The convergence threshold is set to be 10^{-10}.

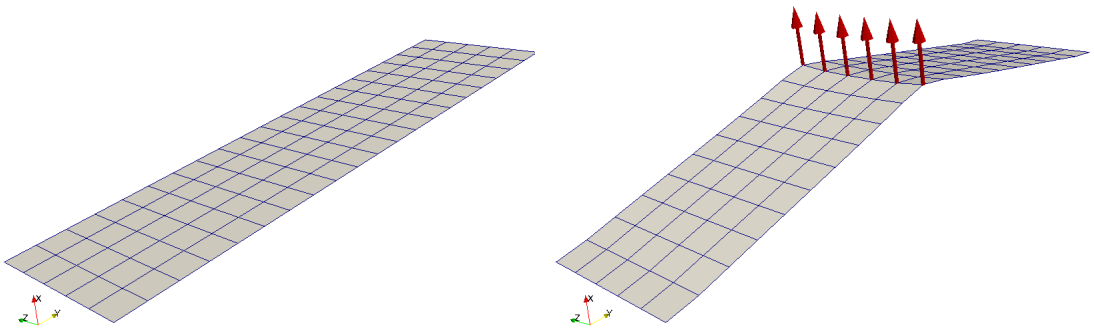


Figure 7.3: **Left**: initial configuration of a plate. **Right**: target configuration

7.2 Nonlinear static PDE-constrained optimal control

7.2.1 Large deflection of a plate

When a structure undergoes large deformation, nonlinear analysis is necessary since the stiffness changes as the structure's configuration changes. A plate composed of shell elements is considered. The shell element has a membrane [1] as well as bending [56] stiffness. The geometric nonlinearity is handled with corotational formulation developed by Felippa [27]. The stiffness of the plate is 2.1×10^8 Pa and Poisson's ratio is 0.3. Initial and target configurations are shown in Figures 7.3. The width of the plate is 0.25 m and the length is 1 m. All the degrees of freedom at both ends are completely fixed and the side-displacement in the z-direction is fixed at zero. The target configuration is generated by applying a uniform nodal force of 1000 N along the mid-line of the plate as shown by red arrows on the right of Figure 7.3.

Four optimal control problems are considered in this section. First, the same boundary conditions are applied as the ones in the case used to generate the target configuration. Then, control forces are allowed on every degree of freedom; these include angular as well as translational forces. The optimal translational force and torque controls corresponding to the first problem are shown on the left and right in Figures 7.4, respectively, with the initial configuration in a wire-frame format to see how much the plate deforms. The contour colors of the deformed plate also indicate the degree of deformation. The legend on the left (DISP Magnitude) shows that the maximum displacement is around 0.1 m. The arrows and the legends (i.e., "TCON Magnitude" and "RCON Magnitude") show the optimal translational force (on the left) and torque (on the right) controls. These are almost the same as the forces that are used in order to generate the target configuration (e.g., 1002.9 N in a slightly perturbed x-direction for translational control and relatively small magnitude of torque).

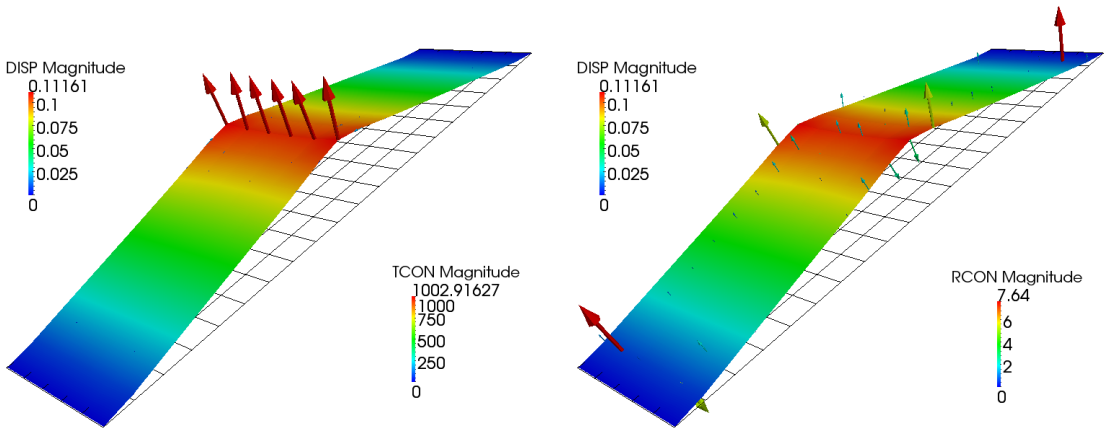


Figure 7.4: Optimal solutions of the first problem. **Left**: the optimal translational force control. **Right**: the optimal torque control

	P_{zrc}	P_{lt2}	P_{rs2}
computational time (sec)	6.90	1.23	0.99
number of iterations	2(10895)	9(1815)	9(1300)

Table 7.11: The table compares the total computational time in seconds and the number of iterations to converge of three different preconditioners for the first optimal control problem of a nonlinear static plate. The regularization parameter of 10^{-20} and the convergence threshold of 10^{-5} is used. For the number of iterations, the number outside of parentheses is the number of major iterations and the inside is the total number of minor iterations.

The size of the KKT system for the first problem is 1938×1938. The number of control variables is the same as the number of state variables. Thus, preconditioner P_{zrc} as well as P_{lt2} and P_{rs2} are applicable to this problem. Table 7.11 compares the total computational time in seconds and the number of iterations required for convergence for these three preconditioners in order to solve the first problem. The constraint preconditioner P_{zrc} requires only two major iterations, while the other two preconditioners require nine major iterations. If the number of major iterations is the only criterion for the choice of preconditioners, then P_{zrc} must be used for the first problem. However, P_{zrc} needs more computational time for convergence than the other two preconditioners because each major iteration needs many more minor iterations.

The second optimal control problem is solved to see if the optimal control routine can regenerate the forces used to generate the target state by allowing control forces to be applied only to the mid-line of the plate in the x-direction. The optimal control routine could, indeed, regenerate the forces used to generate the target state as shown on the left of Figure 7.5. The number of control variables is much lower than the number of state variables. The size of the KKT system that arises

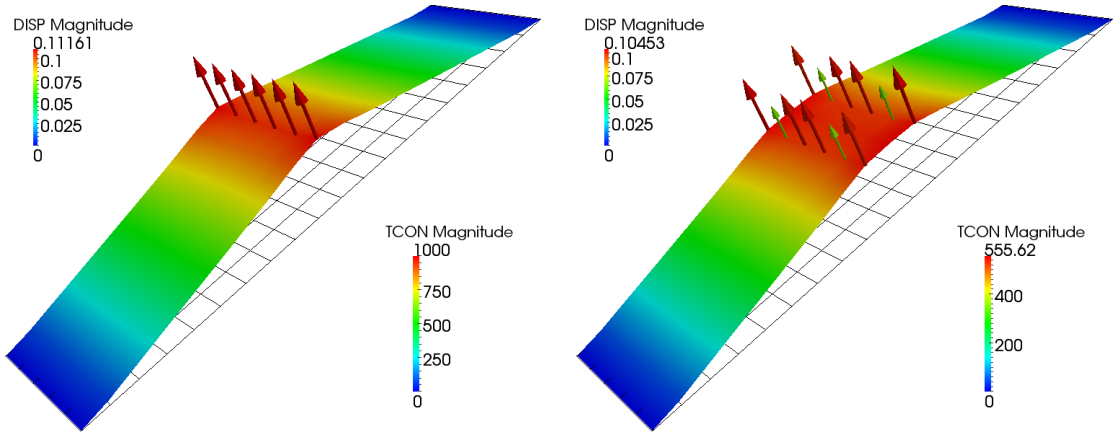


Figure 7.5: Optimal solutions of the second and third problem. **Left**: the optimal translational force control of the second problem. **Right**: the optimal traslational force control of the third problem

in the QP subproblem for the second problem is 1298×1298. Since the KKT system is smaller than the one in the first problem, the second problem is expected to converge faster than the first problem. However, it turns out that the SAND method on the second problem converges slower than on the first problem. Plus, the number of major iterations for the second problem is greater than that for the first problem (see Figure 7.7). The slow convergence rate for the smaller number of control variables appears again for the third and fourth optimal control problems in this section.

The third problem is solved just to see what kind of solution is returned by the optimal control routine if the control is limited to applying to places different from the mid-line. The control is allowed to be applied to the nodes neighboring the ones on the mid-line. The result is shown on the right of Figure 7.5. As expected, it does not match the target state precisely. For example, the sharp edge present in the mid-line of the target is absent. Plus, the maximum displacement is not quite the same as the target maximum displacement (around 0.105 m for the third problem and around 0.112 for the target). However, it matches the target fairly well.

One may wonder if the target is realizable with only torque, and, if not, how close the configuration can be to the target only with torque only. This question can be answered easily by solving an PDE-constrained optimization problem. The fourth problem limits the control to be torque only. Figure 7.6 shows the optimal configuration capable of being generated by torque control only. It turns out that a configuration generated by torque only can be almost identical to the target configuration. In order to see how close it is to the target, the first part of the objective function (i.e. $\frac{1}{2}\|y - \bar{y}\|_V^2$) can be examined. Table 7.12 compares the first part of the objective function values for all four problems considered in this section. Note that optimal configuration from the fourth problem is closer to the target than that from the third problem.

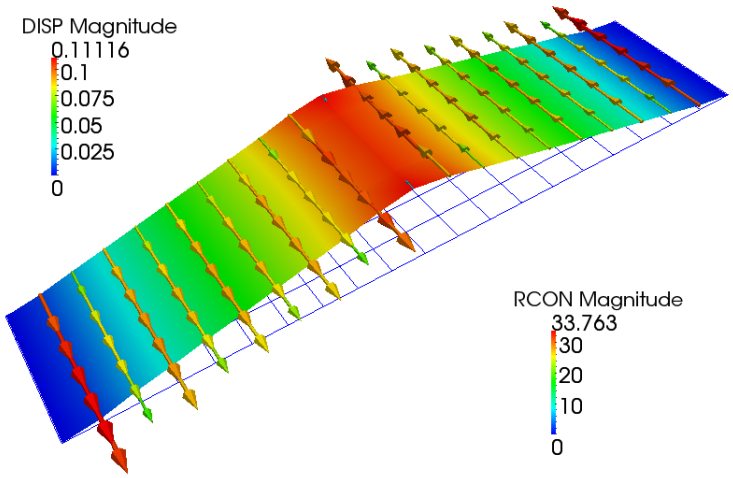


Figure 7.6: The optimal solutions of the fourth problem, in which only torque controls are allowed.

	problem 1	problem 2	problem 3	problem 4
$\frac{1}{2}\|y-\bar{y}\|_V^2$	2.3×10^{-17}	9.8×10^{-14}	1.8×10^{-4}	7.2×10^{-6}

Table 7.12: Comparison of the first part of the objective function values for all four cases considered in Section 7.2.1 in order to see how close optimal configurations are to the target.

As noted in Chapter 3.3, when QP subproblems are formed in SQP, second order derivatives of nonlinear PDE-constraints are omitted. Thus, the convergence rate is expected to be linear at best. As shown in Figure 7.7, the SQP method applied to all of the four problems exhibits a linear convergence rate. The figure shows decrease in the residual norm of the necessary optimality conditions. For the first problem, the rate of convergence is less than 0.1; for the fourth problem, it is around 0.7. However, for the second and third problems, it is close to 1, which implies that it is nearly sub-linear. Note also that the number of control variables is biggest in the first problem (i.e., 646) and next biggest in the fourth problem (i.e., 114), while the second and third problems have the smallest number of control variables (i.e., 6 and 12, respectively). As the number of control variables increases, the rate of convergence seems to be better in the full SAND approach. This is speculated to be the case because greater freedom to reach a target is allowed with more control variables, but a more thorough study needs to be done to verify this. In order to improve the rate of convergence, it may be worth adding either exact or approximate second order derivative information of nonlinear constraints.

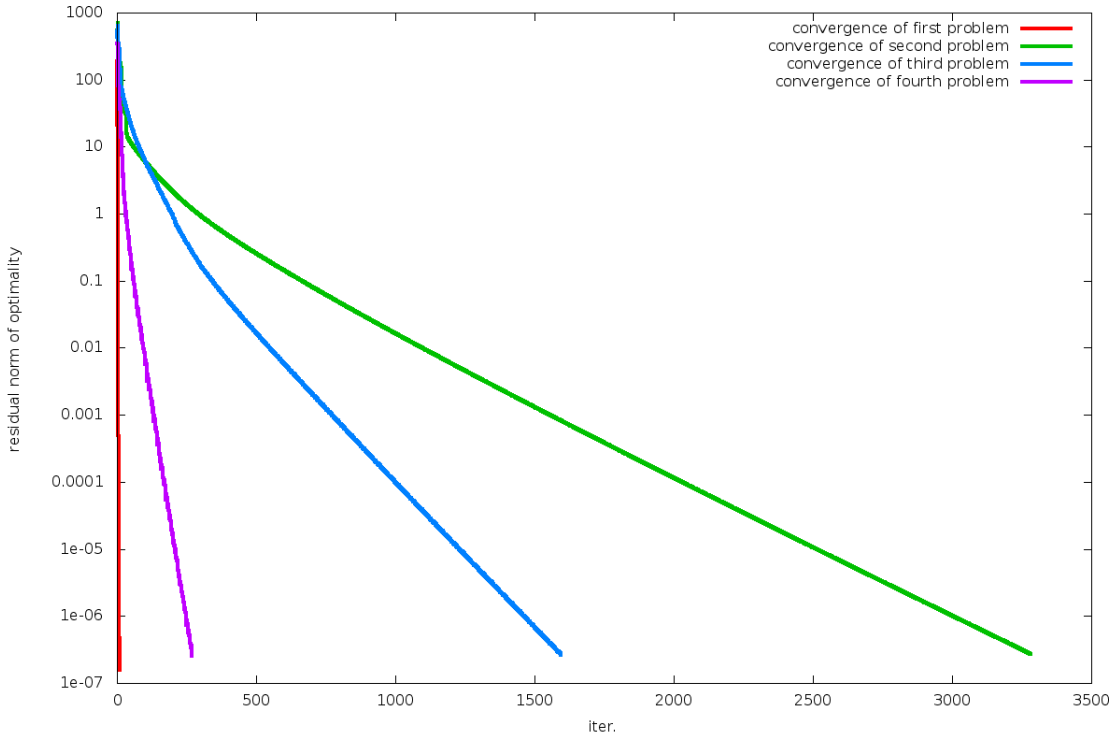


Figure 7.7: Convergence plots for all four problems. The absence of second-order derivatives of nonlinear constraints results in a linear convergence rate. For the first problem, the rate of convergence is less than 0.1. For the fourth problem the rate of convergence is around 0.7. However, for the second and third problems, the rate of convergence is close to one, which implies that it is almost sublinear.

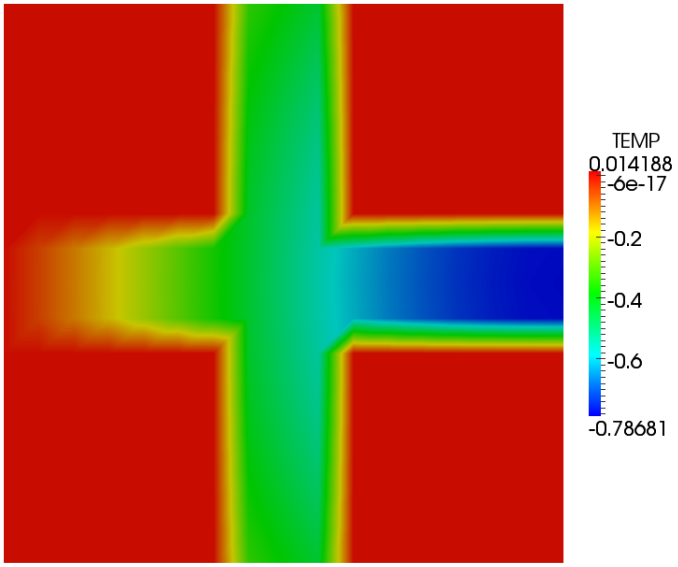


Figure 7.8: The target temperature of the cross heat control problem.

7.3 Linear dynamic PDE-constrained optimal control

7.3.1 A target cross on a heat plate

As in Section 7.1.1, a heat conduction problem on a unit square plate is considered. Instead of a solving static problem, this time, a dynamic problem is considered, and a discontinuous cross-shaped target is used, as shown in Figure 7.8. The target is the same as the one used in the paper [79] by Stoll and Wathen, that is,

$$\bar{y} = \begin{cases} -x_1 e^{-(x_1-0.5)^2-(x_2-0.5)^2} & \text{if } 0.4 \leqslant x_1 \leqslant 0.6 \text{ or } 0.4 \leqslant x_2 \leqslant 0.6, \\ 0 & \text{otherwise.} \end{cases} \tag{7.4}$$

The initial temperature distribution is all zero, and the homogeneous Dirichlet boundary condition is applied. The heat conduction coefficient of 10 $W \cdot m^{-1} \cdot C^{-1}$, specific heat coefficient of 10 J/C, thickness of 10 m are used. From Figure 7.9 to Figure 7.14, optimal temperatures and heat control are shown with various regularization parameters in decreasing order (i.e., $\phi = 1$ for Figure 7.9 and 7.10, $\phi = 0.1$ for Figure 7.9 and 7.10, and $\phi = 0.01$ for Figure 7.9 and 7.10). The legend for heat control shows the heat from -220 J/m^3 to 80 J/m^3, whereas the legend for optimal temperature is the same as the one used for the target temperature (i.e., Figure 7.8). As expected, the optimal temperature gets closer to the target temperature as the regularization decreases.

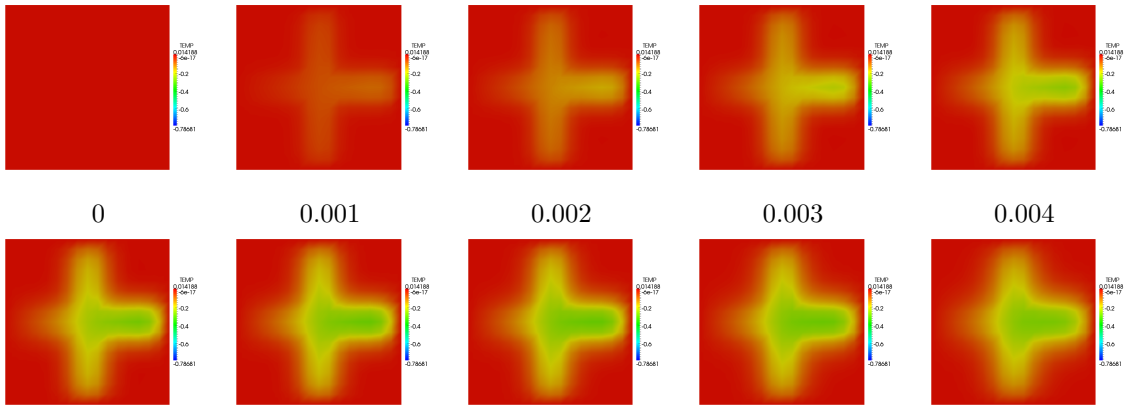


Figure 7.9: A series of stick figures of optimal temperature solutions of the cross heating problem. The regularization $\phi = 1$ is used. The number below each figure indicates the passing time in seconds.

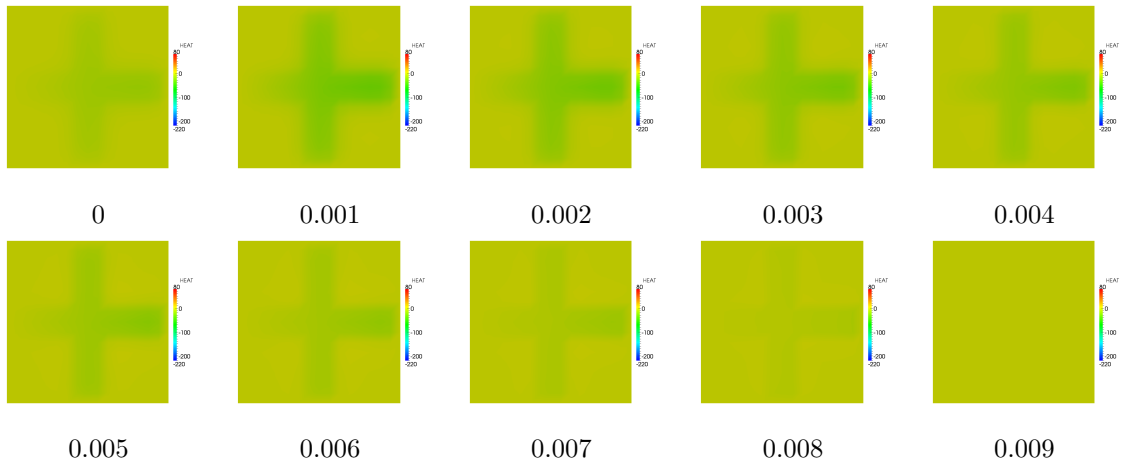


Figure 7.10: A series of stick figures of optimal heat control of the cross heating problem. The regularization $\phi = 1$ is used. The number below each figure indicates the passing time in seconds.

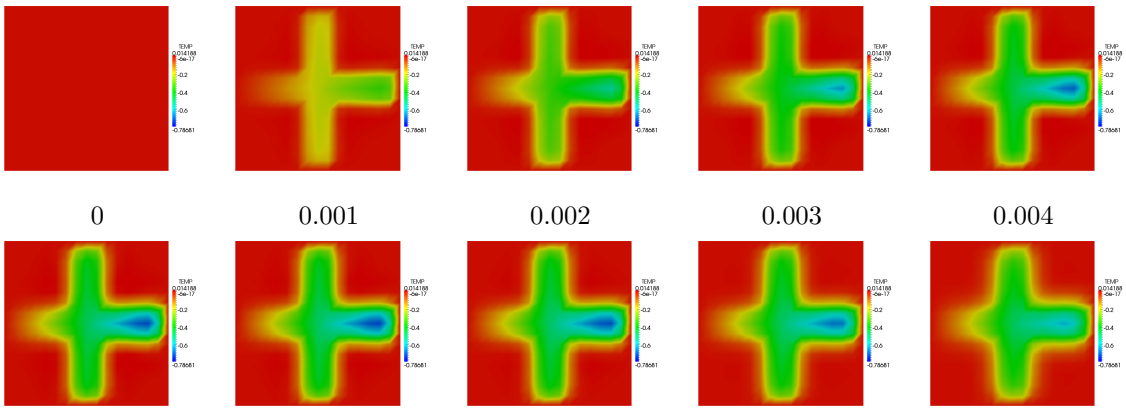


Figure 7.11: A series of stick figures of optimal temperature solutions of the cross heating problem. The regularization $\phi = 0.1$ is used. The number below each figure indicates the passing time in seconds.

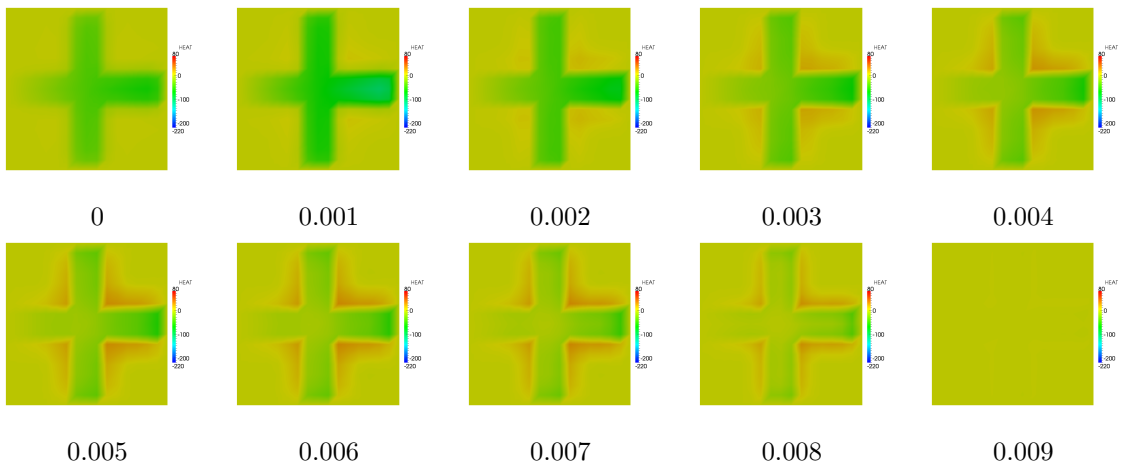


Figure 7.12: A series of stick figures of optimal heat control of the cross heating problem. The regularization $\phi = 0.1$ is used. The number below each figure indicates the passing time in seconds.

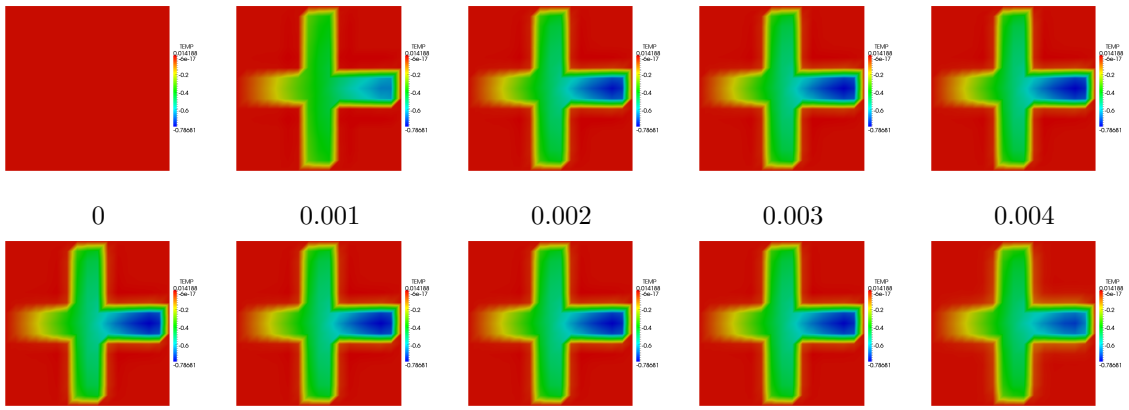


Figure 7.13: A series of stick figures of optimal temperature solutions of the cross heating problem. The regularization $\phi = 0.01$ is used. The number below each figure indicates the passing time in seconds.

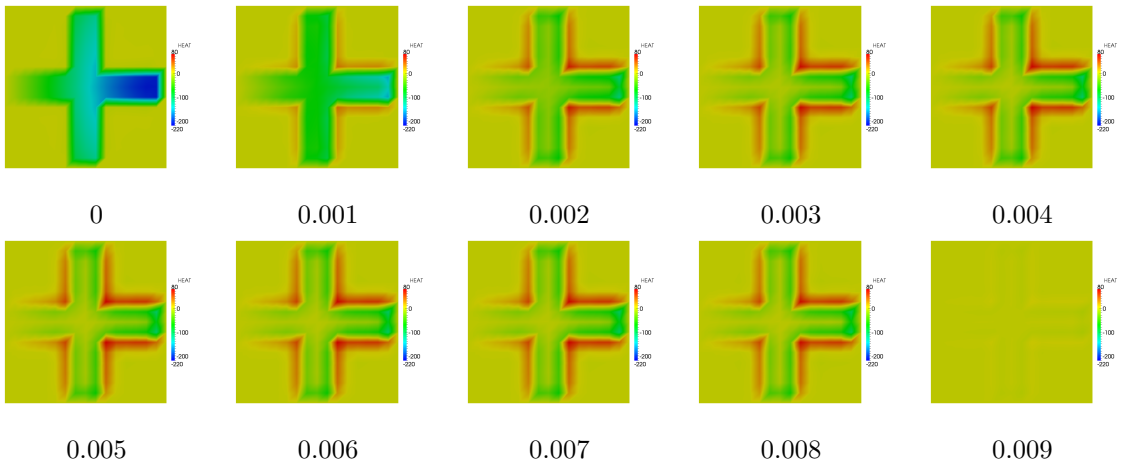


Figure 7.14: A series of stick figures of optimal heat control of the cross heating problem. The regularization $\phi = 0.01$ is used. The number below each figure indicates the passing time in seconds.

final time, t_f	0.1	0.2	0.3	0.4	0.5
number of states	45,450	90,450	135,450	180,450	225,450
number of controls	22,725	45,225	67,725	90,225	112,725
computational time (sec)	11.16	60.80	164.75	339.82	580.01
number of iterations	64	103	131	159	181

Table 7.13: The table shows the scalability of the SAND method in a linear dynamic PDE-constrained optimal control problem. GMRES is used as a Krylov iterative method. The preconditioner is P_{vbg}. The convergence threshold is 10^{-10} and the regularization parameter ϕ is 1.

final time, t_f	0.6	0.7	0.8
number of states	270,450	315,000	360,450
number of controls	135,225	157,500	180,225
computational time (sec)	1046.91	1498.890	2001.740
number of iterations	204	226	245

Table 7.14: The table shows the scalability of the SAND method in a linear dynamic PDE-constrained optimal control problem. GMRES is used as a Krylov iterative method. The preconditioner is P_{vbg}. The convergence threshold is 10^{-10} and the regularization parameter ϕ is 1.

As explained in Chapter 3, the formulation of the SAND method for the linear dynamic PDE-constrained optimization problem in this thesis is simultaneous in the time as well as the space domain. Thus the size of the KKT system that GMRES needs to solve simultaneously increases as the terminal time increases. Tables 7.13 and 7.14 show the scalability of the SAND method. The number of iterations increases linearly as the terminal time t_f increases.

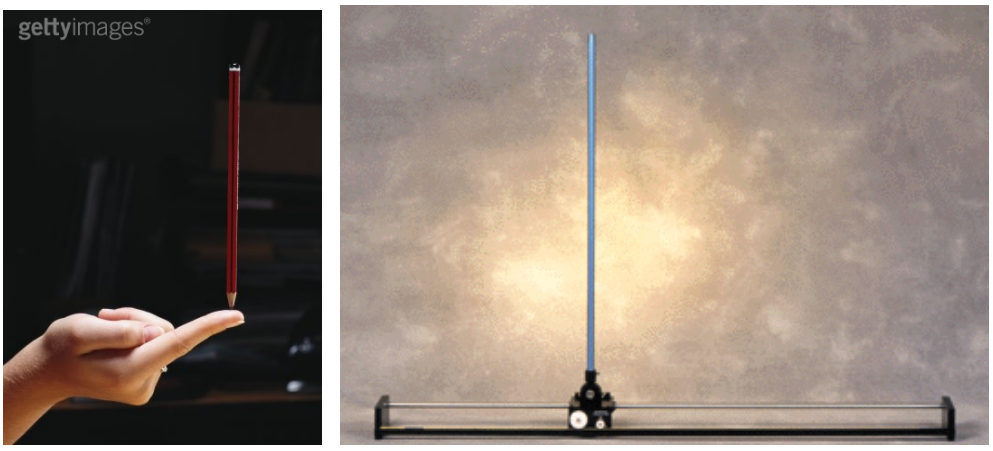


Figure 7.15: **Left**: a photo describing a finger that tries to balance a pencil. **Right**: an inverted pendulum

7.4 Nonlinear dynamic PDE-constrained optimal control

7.4.1 Stabilizing inverted pendulum

An inverted pendulum problem is a classical control problem. The bottom of a pendulum is free to move horizontally, but the pendulum needs to stay as upright as possible. Imagine that you have a pencil on a fingertip, trying to balance it as depicted in Figure 7.15 on the left. You may find yourself moving your finger back and forth or left and right horizontally. The motion of the finger reflects the effort to stabilize the upright pencil. The inverted pendulum is modeled with one beam element with Young's modulus of 2×10^9 Pa, density of 7850 kg/m^3, cross-sectional area of 0.1 m^2, and Poisson's ratio of 0.3. The bottom of the pendulum is fixed in the vertical direction (i.e. the y-direction) and the z-direction (i.e. perpendicular to the paper) but is free to move in the x-direction (i.e. the horizontal direction). The target is set to be a static upright pendulum which does not move. Then, the optimal control problem is solved by only allowing the translational force in the x-direction on the bottom of the pencil.

Figure 7.16 shows a series of stick figures of the target (light gray) and an optimal motion (black) of an inverted pendulum. The initial configuration of the pendulum is slanted slightly in order to show an interesting optimal solution. The red arrows represent the translational control. If the initial configuration were identical to the target configuration, then no movement would be observed in the optimal solution because the pendulum is already in the equilibrium state. The link http://www.stanford.edu/~yc344/anInvertedPendulum.avi is to a movie clip which shows the slow motion of eight complete cycles of stabilizing inverted pendulum motion. As expected, back and forth stabilizing translational force control is shown and the inverted pendulum is trying to stay

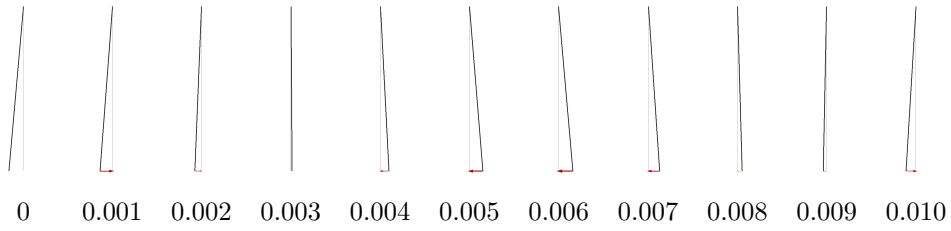


Figure 7.16: A series of stick figures of target (light gray) and optimal solutions of an inverted pendulum are shown. The numbers below each stick figure indicate the corresponding elapsed time in seconds.

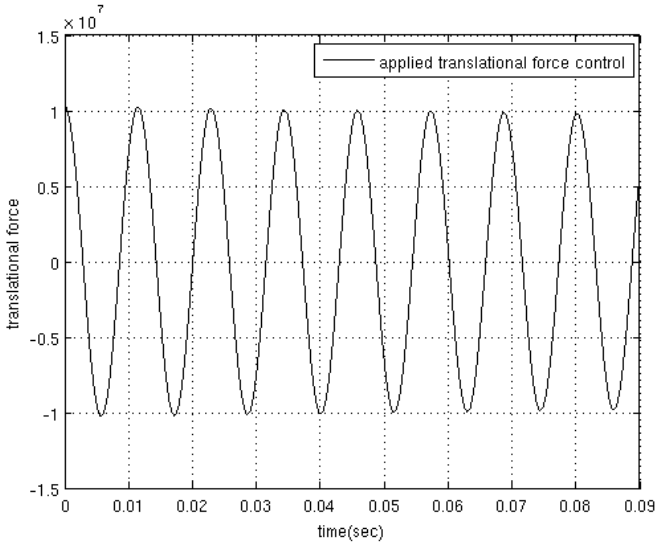


Figure 7.17: Time history of control force at the bottom of the inverted pendulum

upright accordingly. Figure 7.17 shows the corresponding control force plot with respect to time. A periodic pattern with period of about 0.012 seconds and amplitude of about 10^7 is observed.

For the simulation of the inverted pendulum, multi-preconditioned GMRES (MPGMRES) is used as a Krylov iterative method with two preconditioners P_{vbg} and P_{lt2}. When P_{vbg} is used as a single preconditioner, the simulation converges fast, but the optimal solution is static, which means that the inverted pendulum stays at the initial configuration. This solution is not desirable. If P_{lt2} is used, the simulation fails to converge because the target is not realizable. When both P_{vbg} and P_{lt2} are used in MPGMRES, however, the simulation converges fast and the periodic motion of an inverted pendulum is produced. The ℓ_1-penalty function is used as a merit function in the linesearch. The convergence threshold for the major loop is 10^{-7} and the one for the minor loop is 10^{-15}. The regularization parameter is set to be 10^{-18}. The time step Δt is 10^{-5} seconds and the terminal time

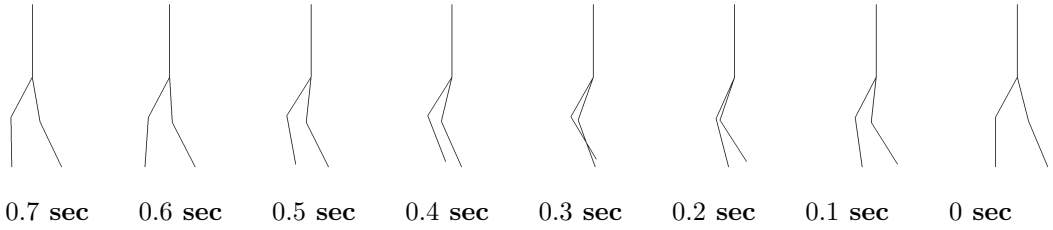


Figure 7.18: **Target motion** a series of stick figures of the target walk of a five-link biped

t_f is 0.09 seconds.

7.4.2 Five-link rigid biped control

Finding a natural gait for a biped robot and its control is an active research area. This is because it lays the foundation for the development of a humanoid robot. The first biped robot of practical size was developed by Kato [22]. Since 1986, Honda has been developing a domestic robot, that coexists with people and helps them with household chores [47]. Their current version of the humanoid robot is ASIMO[5]. Although ASIMO is able to walk and run in a very stable manner, it exhibits a particular gait, which looks different from a normal human gait. For further development of a humanoid robot, it is necessary to be able to generate different gaits and to control them in a stable manner. In the literature, the sagittal gait of a five-link biped is often considered due to its simplicity and similarity to human structures. There are two ways of computationally achieving a natural gait of a biped: using image processing [50] and polynomial interpolation [81, 58]. Mu and Wu used a polynomial function as a way of getting a natural gait of a five-link biped[58]. Their approach is used to generate a target motion of a biped robot in this section. The polynomials and their coefficients are obtained by satisfying a number of constraints (e.g., stability, minimizing impact). The polynomial functions describe a motion of two tips of legs and a hip. Then the trajectory of two joints is obtained using rigid body constraints. One of the constraints is used to minimize impacts with the ground when the swing leg hits it by setting the arrival velocity to zero. Kim, et al. [50] used sagittal and frontal plane images of an actual human walking in order to get 3D gait information. They used a genetic algorithm (GA) to generate a natural and stable 3D gait.

Several numerical methods of solving optimal robotic control have been proposed in the literature: Hardt, et al. [43] first formed the recursive symbolic dynamic model by reducing the variables, then used an optimal control software, DIRCOL, which again used optimization software NPSOL and SNOPT. They minimized the power provided by controllers.

A full SQP method implemented in AERO-S is used to obtain the control necessary on each joint of a five-link biped model to match a given target gait. Figure 7.18 shows a series of stick figures of target motion generated by polynomial functions described in the paper by Mu and Wu [58]. Two

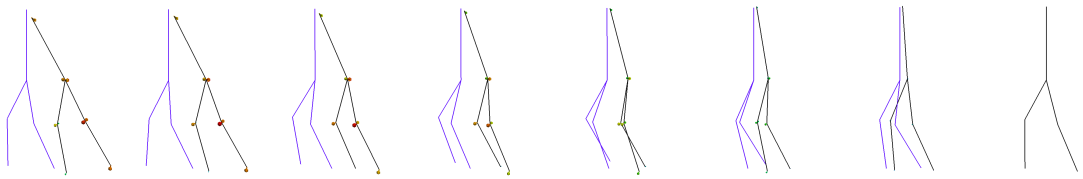


Figure 7.19: **The first problem**: only torque controls are allowed for each joint. A series of stick figures of target in light blue and control solutions in black are shown. Arrows indicate the torque vectors applied to joints.

optimal control problems are then solved with two different sets of controls. The first problem only allows rotational controls at each joint whereas the second problem allows additional translational controls at the tip of a leg. All the optimal control problems are solved assuming there is no friction on the ground because AERO-S does not have a friction model. Thus, without a translational control on the tip of a leg (e.g., in the first problem), the biped is expected to slip on the ground. The ground condition (i.e., the biped is not allowed to go below the ground) is applied with linear multi-point constraints (LMPC). All the constraints necessary to impose a rigid beam, revolute joint, and spring condition including LMPC are solved with a penalty method.

Figure 7.19 shows a series of stick figures of two robots for the first problem. The light blue one is the target, the black one is generated by solving the first problem. Arrows attached to the black biped's joints display a set of torque controls necessary for the indicated black biped's motion. Since there is no friction on the ground, it does not move forward as the target motion does. However, it tries to follow the target by rotating and getting closer to the target (e.g., the top node of the torso is trying to follow the top of the torso in the target biped). Note that the ground condition is not satisfied since the toe of the black biped is slightly under the ground (e.g., see 0.7 second in Figure 7.19). It is because the penalty method is used to impose the ground condition.

In order to mimic ground friction, translational force control is allowed to be applied to the tip of a leg in the second problem. The resultant motion compared with the target motion is shown in Figures 7.20 and 7.21. The arrows in Figure 7.20 show the torque controls and the ones in Figure 7.21 show the translational control force at the tip of a leg. They do match the target motion fairly well for this set of controls but not precisely. At the beginning of walking, a big translational stabilizing force at the tip of a leg is required. After that, a moderate translational force follows.

For both problems in this section, MPGMRES is used as a Krylov iterative method and two preconditioners P_{lt2} and P_{vbg} are used. The time step Δt is 0.01 seconds. The terminal time is 1 second. The convergence threshold for major iterations is 10^{-5} and the convergence threshold for minor iterations is 10^{-15}. The value of the regularization parameter is 10^{-18}.

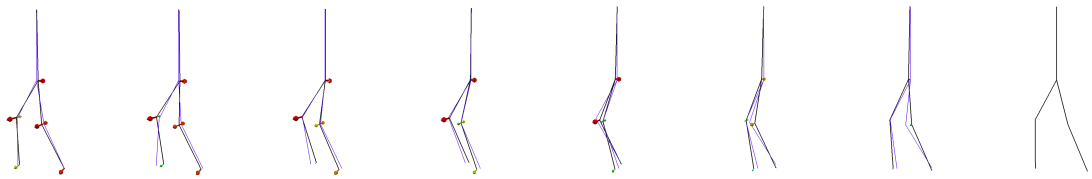


Figure 7.20: **The second problem**: Torque controls at each joint and a translational force control at the tip of a leg. A series of stick figures of the target in light blue and the control solutions in black are shown. Arrows indicate the torque controls applied to joints.

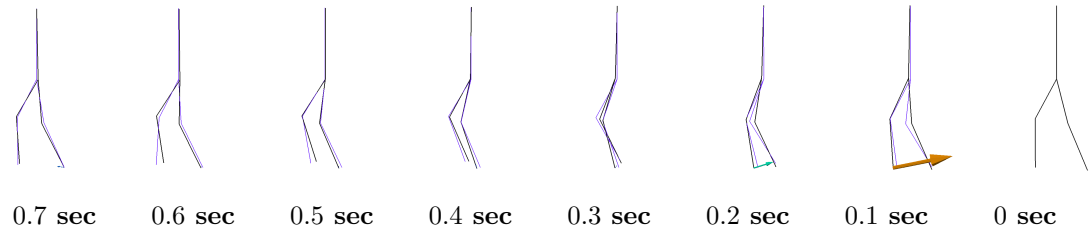


Figure 7.21: **The second problem**: Torque controls at each joint and a translational force control at the tip of a leg. A series of stick figures of the target in light blue and the control solutions in black are shown. Arrows indicate the translational force control applied to a toe.

The animation for the target motion can be found at the following link, http://www.stanford.edu/~yc344/targetbiped.avi. The animation for the first problem can be found at http://www.stanford.edu/~yc344/case1.avi. The animation that shows torque controls at each joint for the second problem can be found at http://www.stanford.edu/~yc344/case2rot.avi. The animation that shows translational controls at the tip of a leg can be found at http://www.stanford.edu/~yc344/case2trans.avi.

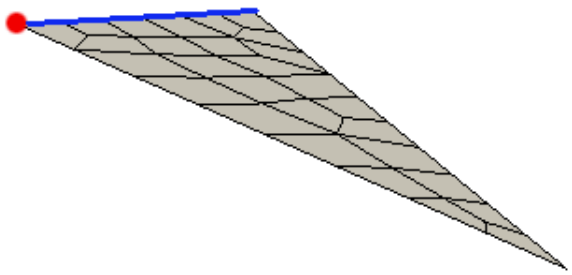


Figure 7.22: A picture of flapping wing

7.4.3 Nonlinear flapping wing

The US Army is interested in developing a drone with flapping wing. The reason for this interest is that flapping wings are more efficient and wind-tolerant than inert wings. Past military drones were limited in their ability to stop-and-start in the air - a serious barrier to drones offering surveillance of a given area [2]. In 2011, Aerovironment designed the winged drone - the Hummingbird. It consists of its own motor, battery, communication system, and a video camera and weighs 19 grams. The hummingbird hovers for eight minutes, flies at 11 miles per hour, and soar from outdoors to indoors and back outside again through a normal doorway. In addition to being able to climb and descend vertically, the Nano Hummingbird can fly forward and backward. To give the video camera a full 360-degree view of a room, it can rotate during flight [3]. For any motion (e.g., hovering or soaring), it is important to know how to control actuator which induces the flapping.

In this section, a numerical example of a nonlinear flapping wing control is considered. Flapping results in large deformation in the wing, so the PDE that describes a flapping wing must be nonlinear. Fluid-structure interaction is necessary in order for flapping motion to induce lift. However, the simulation is done without fluid (i.e., a dry simulation) because fluid optimal control routine has not been implemented yet. Only one wing is modeled under the assumption of the symmetry of the wings (see Figure 7.22). The span of the a flapping wing is modeled with thin shell elements. Two flexible beams are connected through the red point in Figure 7.22: one in the direction of the blue line and the other in the direction of the frontal line. A flapping motion of the wing is generated as a target by prescribing rotational displacements at the frontal node of the wing and running the PDE solver AERO-S. The frontal node is depicted as a red dot in Figure 7.22. The prescribed rotational displacements are applied in the direction of the blue line. Eight snapshots of the target states are shown in Figure 7.24. Then optimal control is solved with a control of torque allowed to be applied only to the same node and direction to which the prescribed rotational displacement is applied. In the top left of Figure 7.23, two plots of rotational displacements of both target and optimal solutions at the red point are shown. The rotational displacement for the target (blue dashed line) follows

a sinusoidal curve, while the one for optimal control (thick black line) is more or less the same as that of the target. In the top right of Figure 7.23, the reaction torque from the target (blue dashed line) and optimal control torque (thick black line) are shown. The two plots are again more or less similar to each other, but at the beginning of the simulation, more oscillation is present for the reaction torque from the target than for the optimal control torque. Magnified portions of the graphs are shown at the bottom of Figure 7.23. On the left, the rotational displacement plots are shown, while the torque plots are shown on the right. In the magnified figures, the optimal rotational displacement starts with almost zero velocity, while the target rotational displacement has nonzero initial slope, meaning nonzero initial velocity. As a result, the initial target reaction torque needs to be large and the oscillation follows right after in order to stabilize the flapping motion. The absence of the oscillation with optimal control torque is due to almost zero initial velocity and no need for stabilization. In the design or control of flapping wing applications, the data from the optimal control routine is more realistic than that from the target because the real flapping wing model will not have nonzero initial velocity, unlike the target. Additionally, it is easier for the actuator to apply smooth rather than oscillatory torque. Figure 7.25 shows both snapshots of target (light gray) and optimal state (black). The optimal states match the target states almost exactly up to 0.01 seconds. From 0.015 seconds, some visually obvious discrepancy starts to show up in tip displacements. The optimal states always fall behind in movement because they do not start with nonzero initial velocity. The arrow in Figure 7.25 depicts the direction and magnitude of optimal control torque. The simulation can be watched through the following link http://www.stanford.edu/~yc344/thickcase1.avi. The movie shows about one and a half cycles of flapping.

Remark 1. For the generation of target states, the time step of 10^{-4} is small enough to obtain a stable solution. However, the optimal control with external torque requires a finer time step. The results shown in this section are generated with a time step of 5.0×10^{-5}.

Remark 2. For the simulation of the flapping wing, MPGMRES with two preconditioners P_{lt2} and P_{vbg} is used as a Krylov iterative method.

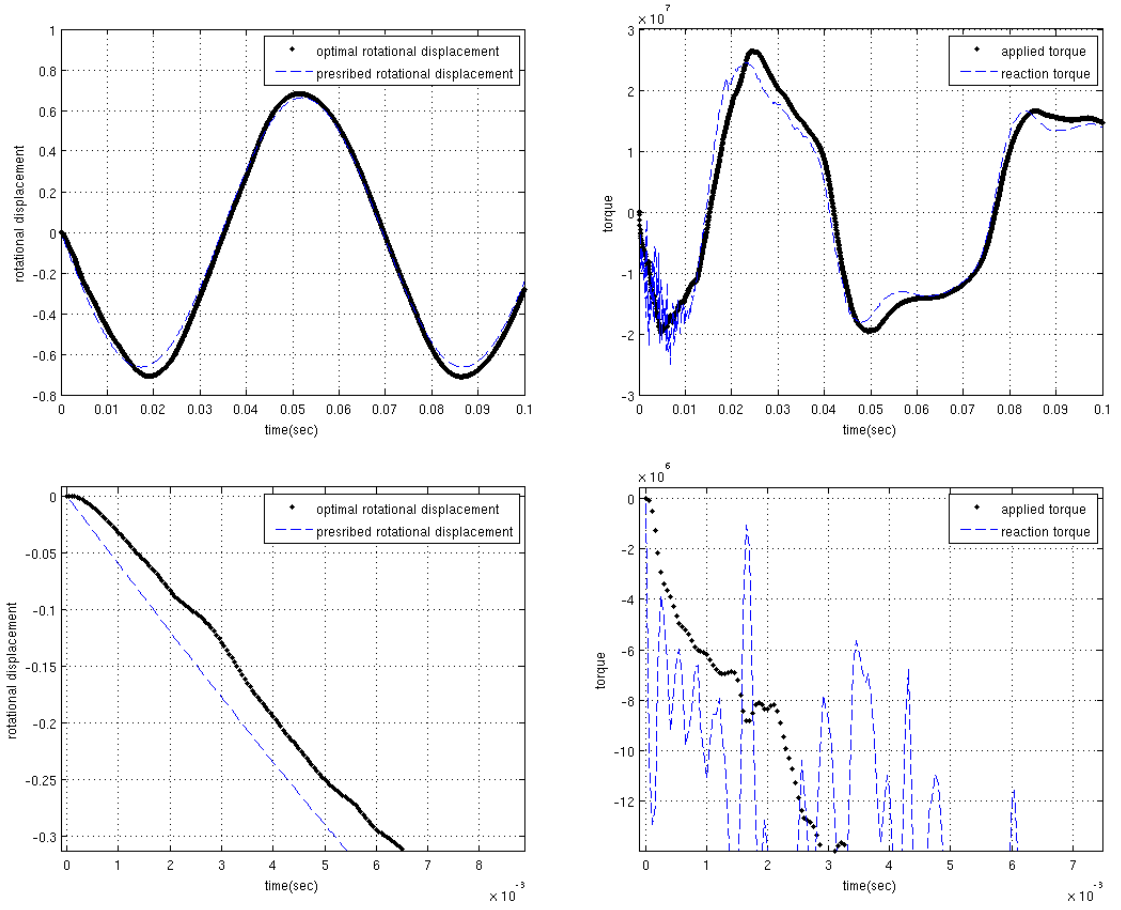


Figure 7.23: Rotational displacements and applied force at the red dot in Figure 7.22 are depicted. **Top left**: prescribed rotational displacement with blue dashed line and corresponding optimal rotational displacement with thick black line. **Top right**: reaction torque in the target with blue dashed line and optimal torque in the optimal solutions with think black line. **Bottom left**: magnified prescribed rotational displacement at the beginning of time. **Bottom right**: magnified applied torque at the beginning of time

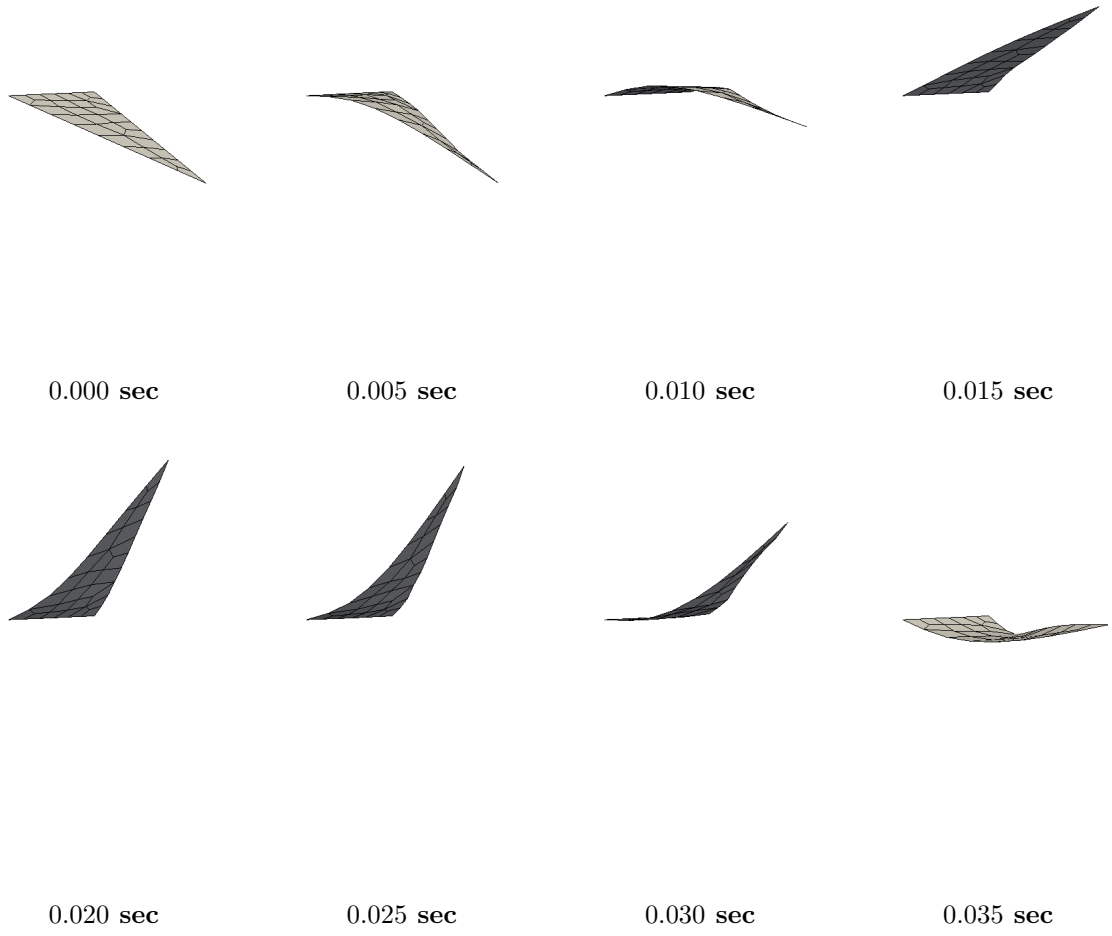


Figure 7.24: **Snapshots of target flapping motion.** A series of snapshot figures of target states, generated by applying prescribed rotational displacements as shown in the middle of Figures 7.22

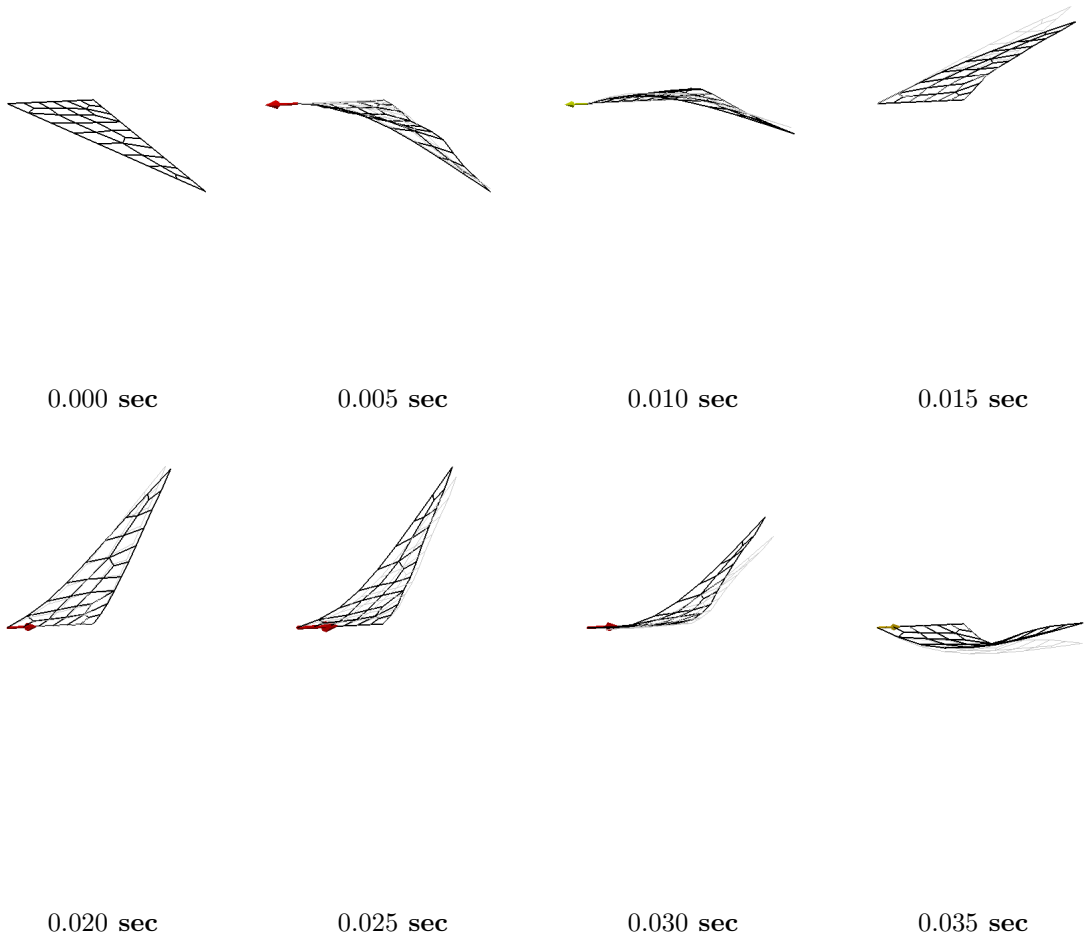


Figure 7.25: **Snapshots of target and optimal solution motion.** A series of snapshot figures of target and optimal solutions. Targets are depicted with light gray and optimal solution with black wire-frames. The arrow depicts the direction and magnitude of applied torque as the optimal control solution.

Chapter 8

Conclusion

As shown in this thesis, PDE-constrained optimization is a powerful tool for many practical applications. This chapter provides a summary of the work done in this research and proposes potential future work.

8.1 Summary

A SAND method for four types of PDE-constrained optimization is formulated and implemented as an SQP augmentation to AERO-S. The PDEs focused on in this thesis are ones that reflect thermal and structural systems. The four types include linear static, nonlinear static, linear dynamic, and nonlinear dynamic PDEs. Many existing preconditioners in the literature improve as the value of a regularization parameter becomes larger, but until now, none existed that improve as the regularization parameter becomes smaller. In order to fill this gap in the literature, several novel preconditioners for a small regularization parameter are introduced. Various numerical results show that the novel preconditioners work well when the target is realizable. Some preconditioners do not even need to solve with the Jacobian related to the PDE, which is a good feature because solving with the Jacobian is computationally expensive. A "useful" exact representation of the Schur complement for the KKT system that arises in a volume control problem is introduced. With this exact representation, a fast way of solving with the Schur complement is now possible. The exact representation can be used as a solver in the range-space method directly or as a preconditioner in a full SQP method. The numerical results show that the exact representation enables the range-space method to work well for any value of the regularization parameter. The SAND framework presented here is well suited to augmenting sophisticated numerical PDE methods. For example, a parallel solver FETI-DP in AERO-S is augmented to the PDE-constrained optimization routine. Also, the generalized α method for a nonlinear dynamic PDE-constrained optimization problem is adapted to the SAND formulation, which has not been done in the literature. The nonlinear

dynamic formulation for optimization is sequential, and can be readily adapted to closed-loop control problems.

8.2 Future work

The SAND method and novel preconditioners introduced in this thesis are applied to a rather simple problem, namely the PDE-constrained optimal control problem whose goal is to match a target. More complicated PDE-constrained optimization need to be tackled, such as shape optimization and inverse problems. A common practical problem is any fluid-structure interaction (FSI) problem. Formulating and implementing a SAND method for an FSI problem has great potential for applications. The PDE-constrained optimization formulation considered here is limited to an equality constraint. Including inequality constraints is necessary to broaden the range of problems that PDE-constrained optimization can solve. Also, the current formulation does not include second-order information of the PDE constraints, resulting in a linear rate of convergence. To achieve a higher rate of convergence, one needs to include approximate second-order information of the PDE constraints. Although the SAND method is a highly useful interface to augment any existing PDE solvers, the current implementation in AERO-S only makes use of the generalized-α method and FETI-DP. More numerical methods for PDEs, such as PITA and reduced-order methods, need to be augmented in the SAND method. In particular, constructing a surrogate PDE model with reduced-order methods is attractive because it reduces the number of optimization variables significantly. In the process of adapting reduced-order methods, discovering methods to update the basis vectors as the optimization proceeds is also a promising future research topic.

Appendix A

Spectral analysis

In this appendix, a spectral analyses of the preconditioners P_{zrc}, P_{rees}, P_{bd2}, P_{lt1}, and P_{lt2}, – which have been introduced in Chapter 5 – are presented. Since P_{rees} and P_{bd2} differ by how they modify the Schur complement S, it is possible to introduce a generalized form of such a block diagonal preconditioner and conduct the spectral analysis of it (e.g., see Section A.3). Similarly, in Section A.4, a generalized form of block triangular preconditioners are introduced and the spectral analysis is done in order to see the properties of eigenvalue distribution of P_{lt1} and P_{lt2}. All the preconditioners are structured as a permuted block triangular matrix. Thus applying them is equivalent to the block forward or back substitution.

The following factorization of the saddle point system [7] is useful in the spectral analysis:

$$\begin{bmatrix} A & B_1^T \\ B_2 & -C \end{bmatrix} = \begin{bmatrix} I & \\ B_2A^{-1} & I \end{bmatrix} \begin{bmatrix} A & \\ & S \end{bmatrix} \begin{bmatrix} I & A^{-1}B_1^T \\ & I \end{bmatrix}, \tag{A.1}$$

which leads to the following identity,

$$\mathbf{det}\left(\begin{bmatrix} A & B_1^T \\ B_2 & -C \end{bmatrix}\right) = \mathbf{det}(A)\mathbf{det}(S), \tag{A.2}$$

where $S = -C - B_2A^{-1}B_1^T$. Since all the preconditioners introduced in this thesis are block matrices, the preconditioned systems are also block matrices. Thus, it is clear that identity (A.2) will be useful in analyzing the eigenvalues of the preconditioned system.

A.1 Zero-regularization constraint preconditioner

As introduced in Section 5.2.4, the constraint preconditioner P_{zrc} is defined as

$$P_{zrc} = \begin{bmatrix} V & & J^T \\ & & Q^T \\ J & Q & \end{bmatrix}. \tag{A.3}$$

An eigenvalue λ of $P_{zrc}^{-1}A$ can be found by solving

$$0 = \mathbf{det}(A - \lambda P_{zrc}) = \begin{vmatrix} (1-\lambda)V & & (1-\lambda)J^T \\ & \phi G & (1-\lambda)Q^T \\ (1-\lambda)J & (1-\lambda)Q & \end{vmatrix}. \tag{A.4}$$

Due to identity (A.2), it is equivalent to

$$\begin{aligned} 0 &= (1-\lambda)^n \mathbf{det}\left(-(1-\lambda)^2 \begin{bmatrix} J & Q \end{bmatrix} \begin{bmatrix} \frac{1}{1-\lambda}V^{-1} & \\ & \frac{1}{\phi}G^{-1} \end{bmatrix} \begin{bmatrix} J^T \\ Q^T \end{bmatrix} \right), \\ &= (-1)^n (1-\lambda)^n \mathbf{det}\left((1-\lambda) J V^{-1} J^T + \frac{(1-\lambda)^2}{\phi} Q G^{-1} Q^T \right), \\ &= \left(\frac{-1}{\phi}\right)^n (1-\lambda)^{2n} \mathbf{det}\left(\phi J V^{-1} J^T + (1-\lambda) Q G^{-1} Q^T \right). \end{aligned} \tag{A.5}$$

Assuming J is not singular, the determinant in (A.5) is further manipulated to

$$0 = \mathbf{det}\left(\phi I + (1-\lambda) \right) H), \tag{A.6}$$

where $H = (JV^{-1}J^T)^{-1} Q G^{-1} Q^T$. Let μ be an eigenvalue of H. The determinant vanishes if

$$\lambda = \frac{\phi}{\mu} + 1 \quad \text{for } \mu \neq 0,$$

which proves Theorem 4. Note that the eigenvalues become clustered around 1 for ϕ small. Thus, P_{zrc} must work well for small ϕ. However, μ is located in the denominator, which depends on the FEM discretization. For example, if the linear basis functions of either triangular or quadrilateral elements are used in the linear optimal distributional heat control in Section 7.1.1, then μ is known to be bounded as

$$c_1 h^2 \leqslant \mu \leqslant c_2, \tag{A.7}$$

due to Proposition 1.29 and Theorem 1.32 in [21]. The coefficients c_1 and c_2 are some finite constants that are independent of mesh size h. In this particular case, as the discretization becomes finer, the magnitude of the eigenvalue λ becomes larger, which causes the eigenvalue distribution to be spread

out. This causes a Krylov iterative method such as GMRES to perform poorly; however Table 7.7 shows that the mesh dependence of P_{zrc} is relatively minor compared to other preconditioners such as P_{tl2}, P_{bd2}, and P_{rs2}.

A.2 A variant of Biros-Ghattas constraint preconditioner

As introduced in Section 5.2.4, the constraint preconditioner P_{vbg} is defined as

$$P_{vbg} = \begin{bmatrix} & & J^T \\ & \phi G & Q^T \\ J & Q & \end{bmatrix}. \tag{A.8}$$

An eigenvalue λ of $P_{vbg}^{-1}A$ can be found by solving

$$0 = \mathbf{det}(A - \lambda P_{vbg}) = \begin{vmatrix} V & & (1-\lambda)J^T \\ & \phi(1-\lambda)G & (1-\lambda)Q^T \\ (1-\lambda)J & (1-\lambda)Q & \end{vmatrix}. \tag{A.9}$$

Due to identity (A.2), it is equivalent to

$$\begin{aligned} 0 &= \phi^m(1-\lambda)^m \mathbf{det}\left(-(1-\lambda)^2 \begin{bmatrix} J & Q \end{bmatrix} \begin{bmatrix} V^{-1} & \\ & \frac{1}{\phi(1-\lambda)}G^{-1} \end{bmatrix} \begin{bmatrix} J^T \\ Q^T \end{bmatrix}\right), \\ &= (-1)^n \phi^m (1-\lambda)^{m+n} \mathbf{det}\left((1-\lambda)JV^{-1}J^T + \frac{1}{\phi}QG^{-1}Q^T\right), \\ &= (-1)^n \phi^{m-n}(1-\lambda)^{m+n} \mathbf{det}\left(\phi(1-\lambda)JV^{-1}J^T + QG^{-1}Q^T\right). \end{aligned} \tag{A.10}$$

Assuming J is not singular, the determinant in (A.5) is further manipulated to

$$0 = \mathbf{det}\left(\phi(1-\lambda)I + H\right), \tag{A.11}$$

where $H = (JV^{-1}J^T)^{-1}QG^{-1}Q^T$. Let μ be an eigenvalue of H. Due to (A.10) and (A.11), the determinant vanishes if

$$\lambda = 1, 1, 1 + \frac{\mu}{\phi},$$

which proves Theorem 3. Note that the eigenvalues become clustered around 1 for ϕ large. Thus, P_{vbg} must work well for large ϕ. Additionally, μ is located in the numerator. If the linear basis functions of either triangular or quadrilateral elements are used, refining the mesh size does not cause a Krylov iterative method such as GMRES to increase the number of iterations for convergence due to the bound (A.7) on μ. Table 7.7 shows that the number of iterations for P_{vbg} actually decreases

as mesh size decreases.

A.3 Block diagonal preconditioner

In order to analyze the eigenvalue distribution of P_{bd1} and P_{bd2}, a generalized form of the block diagonal preconditioner P_{gbd} is defined as follows:

$$P_{gbd} = \begin{bmatrix} \gamma V & & \\ & \rho\phi G & \\ & & \zeta U \end{bmatrix}, \tag{A.12}$$

where α and β are real constants. An eigenvalue λ of $P_{gbd}^{-1}A$ can be found by solving

$$0 = \mathbf{det}(A - \lambda P_{gbdp}) = \begin{vmatrix} (1-\gamma\lambda)V & & J^T \\ & (1-\rho\lambda)\phi G & Q^T \\ J & Q & -\zeta\lambda U \end{vmatrix}. \tag{A.13}$$

Due to identity (A.2), it is equivalent to

$$\begin{aligned} 0 &= (1-\gamma\lambda)^n(1-\rho\lambda)^n\phi^n\mathbf{det}\left(-\zeta\lambda U - \begin{bmatrix} J & Q \end{bmatrix}\begin{bmatrix} \frac{1}{1-\gamma\lambda}V^{-1} & \\ & \frac{1}{\phi(1-\rho\lambda)}G^{-1} \end{bmatrix}\begin{bmatrix} J^T \\ Q^T \end{bmatrix}\right), \\ &= (1-\gamma\lambda)^n(1-\rho\lambda)^n\phi^n\mathbf{det}\left(-\zeta\lambda U - \frac{1}{1-\gamma\lambda}JV^{-1}J^T - \frac{1}{\phi(1-\rho\lambda)}QG^{-1}Q^T\right), \\ &= (-1)^n\mathbf{det}\left(\zeta\phi\lambda(1-\gamma\lambda)(1-\rho\lambda)U + \phi(1-\rho\lambda)JV^{-1}J^T + (1-\gamma\lambda)QG^{-1}Q^T\right). \end{aligned} \tag{A.14}$$

Let U be parameterized by some scalar α and β such as

$$U = \alpha JV^{-1}J^T + \beta QG^{-1}Q^T.$$

Assuming J is not singular, (A.14) is further manipulated to

$$0 = \mathbf{det}\left([\alpha\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda) + \phi(1-\rho\lambda)]I + [\beta\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda) + (1-\gamma\lambda)]H\right), \tag{A.15}$$

where $H = (JV^{-1}J^T)^{-1}QG^{-1}Q^T$. Let μ be an eigenvalue of H. Then the determinant vanishes if

$$\begin{aligned} \frac{\alpha\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda) + \phi(1-\rho\lambda)}{\beta\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda) + (1-\gamma\lambda)} &= -\mu \quad \text{for } \mu \neq 0 \\ \alpha\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda) + \phi(1-\rho\lambda) &= 0 \quad \text{for } \mu = 0 \end{aligned}$$

This leads to the following theorem.

Theorem 5. *Let λ be an eigenvalue of $P_{gbd}^{-1}A$ and μ be an eigenvalue of $H = (JV^{-1}J^T)^{-1}QG^{-1}Q^T$. If $\mu = 0$, then λ satisfies*

$$\alpha\zeta\phi\gamma\rho\lambda^3 - \alpha\zeta\phi(\gamma+\rho)\lambda^2 + (\alpha\zeta\phi - \rho\phi)\lambda + \phi = 0$$

If $\mu \neq 0$, then λ obeys

$$(\alpha+\mu\beta)\zeta\phi\gamma\rho\lambda^3 - (\alpha+\mu\beta)\zeta\phi(\gamma+\rho)\lambda^2 + ((\alpha+\mu\beta)\zeta\phi - \rho\phi - \mu\gamma)\lambda + \phi + \mu = 0$$

$$\&$$

$$\beta\zeta\phi\gamma\rho\lambda^3 - \beta\zeta\phi(\gamma+\rho)\lambda^2 + (\beta\zeta\phi - \gamma)\lambda + 1 \neq 0.$$

Consider the following cases:

Case 1. *An eigenvalue of $P_{bd}^{-1}A$ (i.e., $\gamma = 1$, $\rho = 1$, $\zeta = 1$, $\alpha = 1$, and $\beta = \frac{1}{\phi}$) is*

$$\lambda = 1, \frac{1 \pm \sqrt{5}}{2} \quad \text{for both } \mu = 0 \text{ and } \mu \neq 0.$$

Case 2. *An eigenvalue of $P_{rees}^{-1}A$ (i.e., $\gamma = 1$, $\rho = 1$, $\zeta = 1$, $\alpha = 1$, and $\beta = 0$) is*

$$\lambda = 1, \frac{1 \pm \sqrt{5}}{2} \quad \text{for } \mu = 0$$

$$\lambda = 1, \frac{1 \pm \sqrt{5 + \frac{4\mu}{\phi}}}{2} \quad \text{for } \mu \neq 0.$$

Case 3. *An eigenvalue of $P_{bd2}^{-1}A$ (i.e., $\gamma = 1$, $\rho = 1$, $\zeta = 1$, $\alpha = 0$, and $\beta = \frac{1}{\phi}$) is*

$$\lambda = 1 \quad \text{for } \mu = 0$$

$$\lambda = 1, \frac{1 \pm \sqrt{5 + \frac{4\phi}{\mu}}}{2} \quad \text{for } \mu \neq 0.$$

Case 4. *An eigenvalue of $P_{sbd2}^{-1}A$ (i.e., $\gamma = 1$, $\rho = 1$, $\zeta = 1$, and $U = \frac{1}{\phi}I$) is*

$$\lambda = 1, \frac{1 \pm \sqrt{1 + 4\mu_s}}{2},$$

where μ_s is an eigenvalue of the Schur complement $S = JV^{-1}J^T + \frac{1}{\phi}QG^{-1}Q^T$. As claimed by Murphy, there are three distinct nonzero eigenvalues for the preconditioned system $P_{bd}^{-1}A$ (i.e., see case 1). Case 2 corresponds to Rees' block diagonal preconditioner P_{rees}. Note that ϕ is in the denominator and μ is in the numerator. For large ϕ, P_{rees} will act like P_{bd} since its eigenvalue distributions becomes similar to case 1. Furthermore, it is beneficial to have μ in the numerator for linear basis functions of quadrilateral or triangular elements. Due to the bound (A.7), refining

the mesh size does not increase the magnitude of an eigenvalue of $P_{lt1}^{-1}A$. However, note that ϕ is in the numerator and μ in the denominator for eigenvalues of $P_{lt2}^{-1}A$. This explains why P_{lt2} performs better for ϕ small, but it suffers when the mesh size is refined if linear finite element basis functions are used. This is shown in Table 7.2 and Table 7.7. Case 4 shows eigenvalues of $P_{sbd2}^{-1}A$. If S is not singular, then μ_s is not zero. Thus an eigenvalue of $P_{sbd2}^{-1}A$ is not zero. However, if S is singular, then μ_s is zero, resulting in a singular Schur complement S. If J is not singular, then S is not singular. However, for ϕ small, the Schur complement S will be dominated by the second part $\frac{1}{\phi}QG^{-1}Q^T$ and if Q is not a square full rank matrix, then S becomes nearly singular.

In order to verify the effects of the spectral analysis, the contrived example 1 was run with $m = n = 200$. Note that P_{bd2} does not make sense if $m \neq n$ since $QG^{-1}Q^T$ becomes singular. Figure A.1 shows the eigenvalue distributions and convergence of GMRES for the three preconditioners: P_{bd}, P_{rees}, and P_{bd2}. As predicted by both the preceeding analysis and [59], the three distinct real eigenvalues of $P_{bd}^{-1}A$ are 1 and $\frac{1\pm\sqrt{5}}{2}$ for both $\phi = 10^{-6}$ and $\phi = 10^{-2}$, although small imaginary parts are present for $\phi = 10^{-6}$. Also note that the eigenvalues of $P_{rees}^{-1}A$ are clustered around 1 and $\frac{1\pm\sqrt{5}}{2}$ for relatively large $\phi = 10^{-2}$, whereas they are spread out for relatively small $\phi = 10^{-6}$. On the other hand, the eigenvalues of $P_{bd2}^{-1}A$ are more spread out for large $\phi = 10^{-2}$ than for small $\phi = 10^{-6}$. The convergence of GMRES directly reflects the eigenvalue distribution results as shown in the bottom plots of Figure A.1. For $\phi = 10^{-6}$, the performance of GMRES in order of best to worst is P_{bd}, P_{bd2}, P_{rees}, and no preconditioner, while for $\phi = 10^{-2}$ the performance in order of best to worst is P_{bd}, P_{rees}, no preconditioner, and P_{bd2}. It is not surprising that the preconditioned system with P_{bd2} performs worse than with no preconditioner. Figure A.2 shows that eigenvalues become more spread out for $P_{bd2}^{-1}A$ than for no preconditioner.

For the case of $m \neq n$, P_{bd}, P_{rees}, and P_{sbd2} are applicable. The contrived example 1 with $n = 200$ and $m = 100$ was run. Figure A.3 shows the convergence of GMRES with these three preconditioners. As expected, P_{sbd2} does not perform well since the preconditioned system is nearly singular as the Figure on the right in A.3 shows.

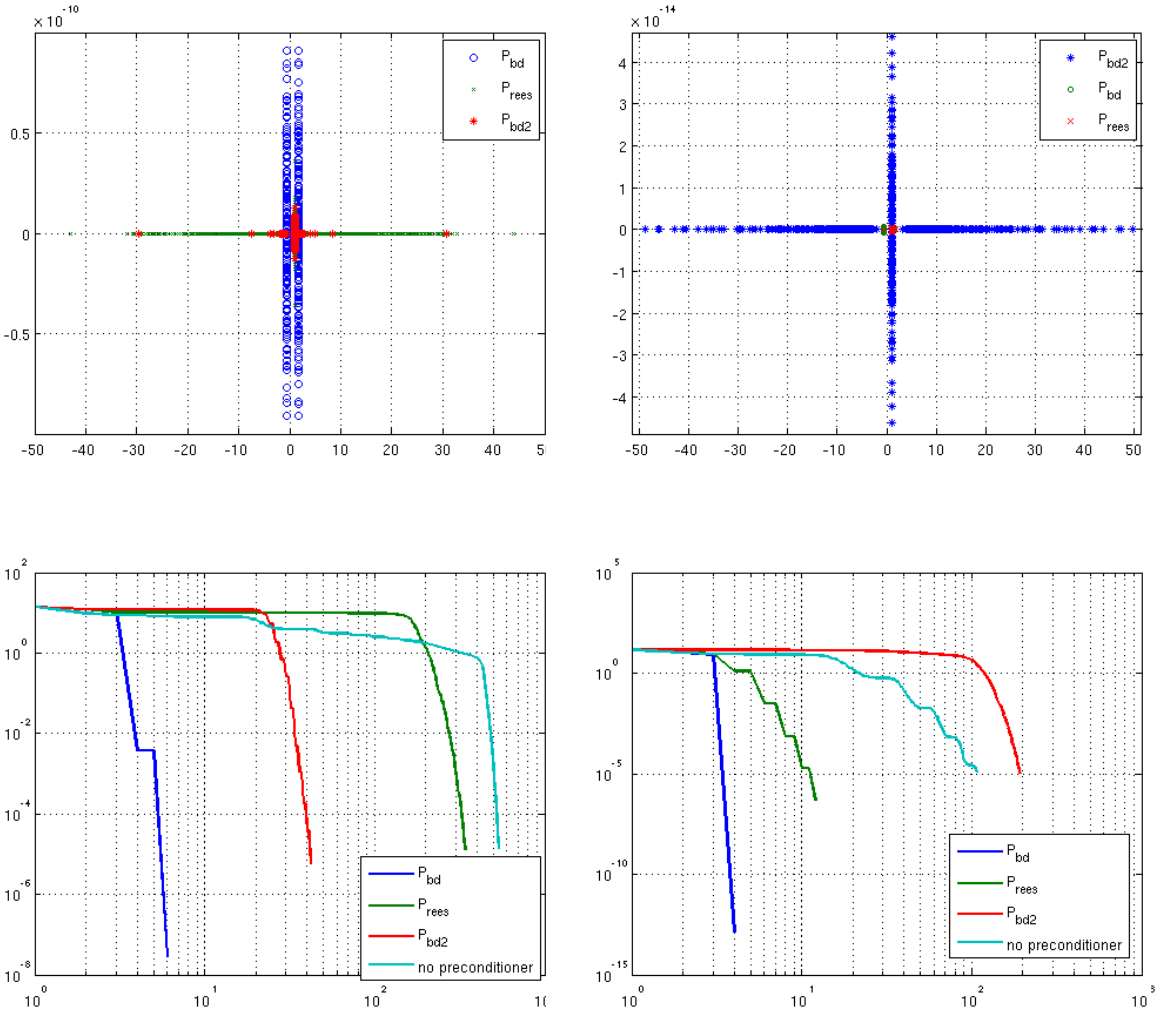


Figure A.1: The top two figures show the eigenvalue distributions in complex plane for three different cases of block diagonal preconditioners considered in Appendix A.3. The bottom two figures show the convergence of GMRES for block diagonal preconditioners. The y-axis of the bottom figures is the relative residual and the x-axis is the number of iterations. **Left**: for $\phi = 10^{-6}$. **Right**: for $\phi = 10^{-2}$

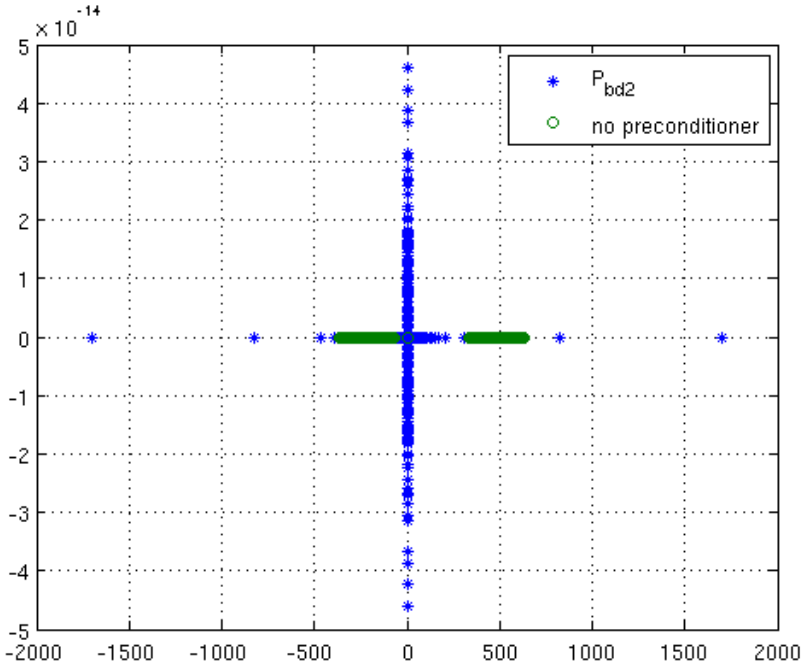


Figure A.2: Eigenvalue distributions of the preconditioned system for case 3 and the non-preconditioned system in the complex plane.

A.4 Block triangular preconditioner

In order to analyze the eigenvalue distributions of P_{lt1} and P_{lt2}, a generalized form of the block triangular preconditioner P_{glt} is defined as follows:

$$P_{glt} = \begin{bmatrix} \gamma V & & \\ & \rho\phi G & \\ J & Q & \zeta U \end{bmatrix}, \tag{A.16}$$

where ζ and γ are real constants. An eigenvalue λ of $P_{glt}^{-1}A$ can be found by solving

$$0 = \mathbf{det}(A - \lambda P_{glt}) = \begin{vmatrix} (1-\gamma\lambda)V & & J^T \\ & (1-\rho\lambda)\phi G & Q^T \\ (1-\lambda)J & (1-\lambda)Q & -\zeta\lambda U \end{vmatrix}. \tag{A.17}$$

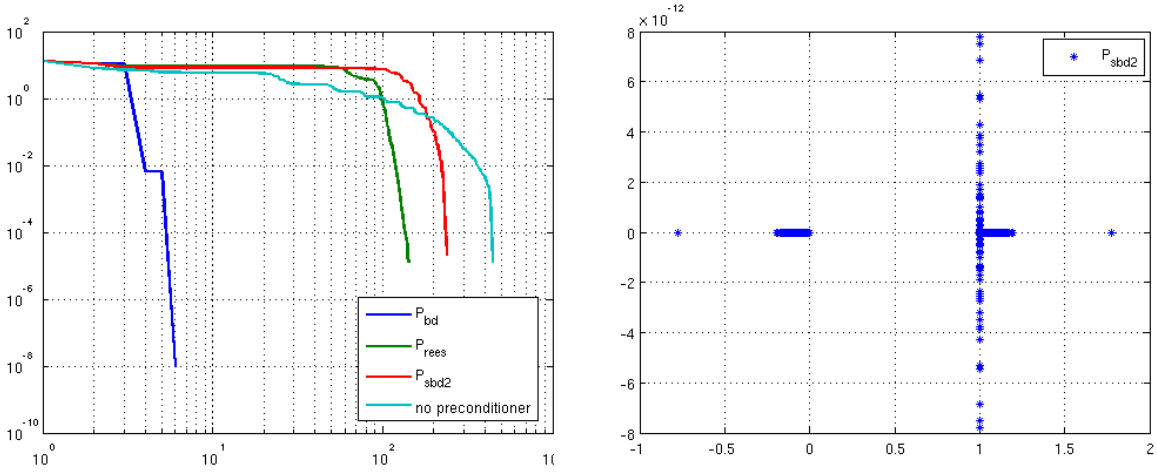


Figure A.3: The contrived example with $n = 200$ and $m = 100$ for block diagonal preconditioners. **Left**: the convergence plot of GMRES for $\phi = 10^{-6}$. **Right**: the eigenvalue distributions in the complex plane for case 4.

Due to identity (A.2), it is equivalent to

$$\begin{aligned}
0 &= (1-\gamma\lambda)^n(1-\rho\lambda)^n\phi^n\mathbf{det}\left(-\zeta\lambda U-(1-\lambda)\begin{bmatrix} J & Q\end{bmatrix}\begin{bmatrix} \frac{1}{1-\gamma\lambda}V^{-1} & \\ & \frac{1}{\phi(1-\rho\lambda)}G^{-1}\end{bmatrix}\begin{bmatrix} J^T \\ Q^T\end{bmatrix}\right),\\
&= (-1)^n(1-\gamma\lambda)^n(1-\rho\lambda)^n\phi^n\mathbf{det}\left(\zeta\lambda U+\frac{1-\lambda}{1-\gamma\lambda}JV^{-1}J^T+\frac{1-\lambda}{\phi(1-\rho\lambda)}QG^{-1}Q^T\right),\\
&= (-1)^n\mathbf{det}\left(-\zeta\phi\lambda(1-\gamma\lambda)(1-\rho\lambda)U+\phi(1-\lambda)(1-\rho\lambda)JV^{-1}J^T+(1-\gamma\lambda)(1-\lambda)QG^{-1}Q^T\right),
\end{aligned} \tag{A.18}$$

Let U be parameterized by some scalar α and β such as

$$U = \alpha JV^{-1}J^T + \beta QG^{-1}Q^T$$

Assuming J is not singular, (A.18) is further manipulated to

$$0 = \mathbf{det}\left([\alpha\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda)+\phi(1-\lambda)(1-\rho\lambda)]I+[\beta\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda)+(1-\gamma\lambda)(1-\lambda)]H\right), \tag{A.19}$$

where $H = (JV^{-1}J^T)^{-1}QG^{-1}Q^T$. Let μ be an eigenvalue of H. Then the determinant vanishes if

$$\begin{aligned}
\frac{\alpha\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda)+\phi(1-\lambda)(1-\rho\lambda)}{\beta\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda)+(1-\gamma\lambda)(1-\lambda)} &= -\mu \quad \text{for } \mu \neq 0\\
\alpha\phi\zeta\lambda(1-\gamma\lambda)(1-\rho\lambda)+\phi(1-\lambda)(1-\rho\lambda) &= 0 \quad \text{for } \mu = 0
\end{aligned}$$

This leads to the following theorem.

Theorem 6. *Let λ be an eigenvalue of $P_{glt}^{-1}A$ and μ be an eigenvalue of $H = (JV^{-1}J^T)^{-1}QG^{-1}Q^T$. If $\mu = 0$, then λ satisfies*

$$\alpha\zeta\phi\gamma\rho\lambda^3 + (\rho\phi - \alpha\zeta\phi(\gamma+\rho))\lambda^2 + (\alpha\zeta\phi - \phi(1+\rho))\lambda + \phi = 0.$$

If $\mu \neq 0$, then λ obeys

$$(\alpha+\mu\beta)\zeta\phi\gamma\rho\lambda^3 + (\rho\phi + \mu\gamma - (\alpha+\mu\beta)\zeta\phi(\gamma+\rho))\lambda^2 + ((\alpha+\mu\beta)\zeta\phi - \phi(1+\rho) - \mu(1+\gamma))\lambda + \phi + \mu = 0$$

&

$$\beta\zeta\phi\gamma\rho\lambda^3 i + (\gamma - \beta\zeta\phi(\gamma+\rho))\lambda^2 + (\beta\zeta\phi - (1+\gamma))\lambda + 1 \neq 0.$$

Consider the following cases:

Case 1. *An eigenvalue of $P_{lt}^{-1}A$ (i.e., $\gamma = 1$, $\rho = 1$, $\zeta = -1$, $\alpha = 1$, and $\beta = \frac{1}{\phi}$) is*

$$\lambda = 1, 1, 1 \quad \text{for both } \mu = 0 \text{ and } \mu \neq 0.$$

Case 2. *An eigenvalue of $P_{lt1}^{-1}A$ (i.e., $\gamma = 1$, $\rho = 1$, $\zeta = -1$, $\alpha = 1$, and $\beta = 0$) is*

$$\begin{aligned} \lambda &= 1, 1, 1 \quad && \text{for } \mu = 0 \\ \lambda &= 1, 1, 1 + \frac{\mu}{\phi} \quad && \text{for } \mu \neq 0. \end{aligned}$$

Case 3. *An eigenvalue of $P_{lt2}^{-1}A$ (i.e., $\gamma = 1$, $\rho = 1$, $\zeta = -1$, $\alpha = 0$, and $\beta = \frac{1}{\phi}$) is*

$$\begin{aligned} \lambda &= 1, 1 \quad && \text{for } \mu = 0 \\ \lambda &= 1, 1, 1 + \frac{\phi}{\mu} \quad && \text{for } \mu \neq 0. \end{aligned}$$

Case 4. *An eigenvalue of $P_{slt2}^{-1}A$ (i.e., $\gamma = 1$, $\rho = 1$, $\zeta = -1$, and $U = \frac{1}{\phi}I$) is*

$$\lambda = 1, 1, \mu_s,$$

where μ_s is an eigenvalue of the Schur complement $S = JV^{-1}J^T + \frac{1}{\phi}QG^{-1}Q^T$. Case 1 shows the spectral results for the preconditioned system with P_{lt}. As expected from factorization (4.27), every eigenvalue is equal to one. Similar to $P_{rees}^{-1}A$, μ is in the numerator for an eigenvalue of $P_{lt1}^{-1}A$. Thus, the clustering of eigenvalues does not deteriorate as a consequence of mesh size refinement, which is demonstrated by Table 7.7 if linear FEM basis functions are used. On the other hand, μ is in the denominator for an eigenvalue of $P_{lt2}^{-1}A$, which implies that this preconditioner will suffer when mesh size is refined for linear FEM basis functions as shown in Table 7.7. Case 4 shows the

spectral analysis for the preconditioned system with P_{slt2} where the Schur complement S in P_{lt} is replaced by $-\frac{1}{\phi}I$. Note that the eigenvalue μ_s is also an eigenvalue of S. As the case of P_{sbd2}, the preconditioned system $P_{slt2}^{-1}A$ becomes nearly singular if Q is not a square full rank matrix and ϕ small.

In order to verify the effects of the spectral analysis, the contrived Example 1 with $m = n = 200$ was run. Note that P_{lt2} is not appropriate if $m \neq n$ since $QG^{-1}Q^T$ becomes singular. Figure A.4 shows the eigenvalue distributions and convergence of GMRES for the three preconditioners: P_{lt}, P_{lt1}, and P_{lt2}. As expected from the analysis above, all the real eigenvalues of $P_{lt}^{-1}A$ are 1 for both $\phi = 10^{-6}$ and $\phi = 10^{-2}$ although small imaginary parts are present. Also note that the eigenvalues of $P_{lt1}^{-1}A$ are clustered around 1 for relatively large $\phi = 10^{-2}$, whereas they are spread out for relatively small $\phi = 10^{-6}$. On the other hand, the eigenvalues of $P_{lt2}^{-1}A$ are clustered around 1 for $\phi = 10^{-6}$, but spread out for $\phi = 10^{-2}$. The convergence of GMRES directly reflects the eigenvalue distribution results as shown in the bottom plots of Figure A.4. For $\phi = 10^{-6}$, the performance of GMRES in order of best to worst is P_{lt}, P_{lt2}, P_{lt1}, and no preconditioner, while for $\phi = 10^{-2}$ the performance in order of best to worst is P_{lt}, P_{lt1}, P_{lt2}, and no preconditioner.

For the case of $m \neq n$, P_{lt}, P_{lt1}, and P_{slt2} are applicable. The contrived example 1 with $n = 200$ and $m = 100$ was run. Figure A.5 shows the convergence of GMRES with these three preconditioners. As expected, P_{slt2} does not perform well since the preconditioned system is nearly singular as the figure on the right in A.5 shows.

One can define a generalized upper triangular preconditioner P_{gut} as it follows:

$$P_{gut} = \begin{bmatrix} \beta M & & K^T \\ & \rho\phi G & -G^T \\ & & \alpha G \end{bmatrix}.$$

It must have the same eigenvalues as P_{glt} because $\mathbf{det}(A - \lambda P_{gut}) = \mathbf{det}(A - \lambda P_{glt})$ due to the symmetry of A. Thus the spectral properties of $P_{gut}^{-1}A$ and $P_{glt}^{-1}A$ are identical. One should prefer P_{glt} to P_{gut} if the adjoint operator is not available since P_{glt} does not require an adjoint operator. However, if the adjoint operator is available and easier to apply than the forward operator, then one should prefer P_{gut} to P_{glt}.

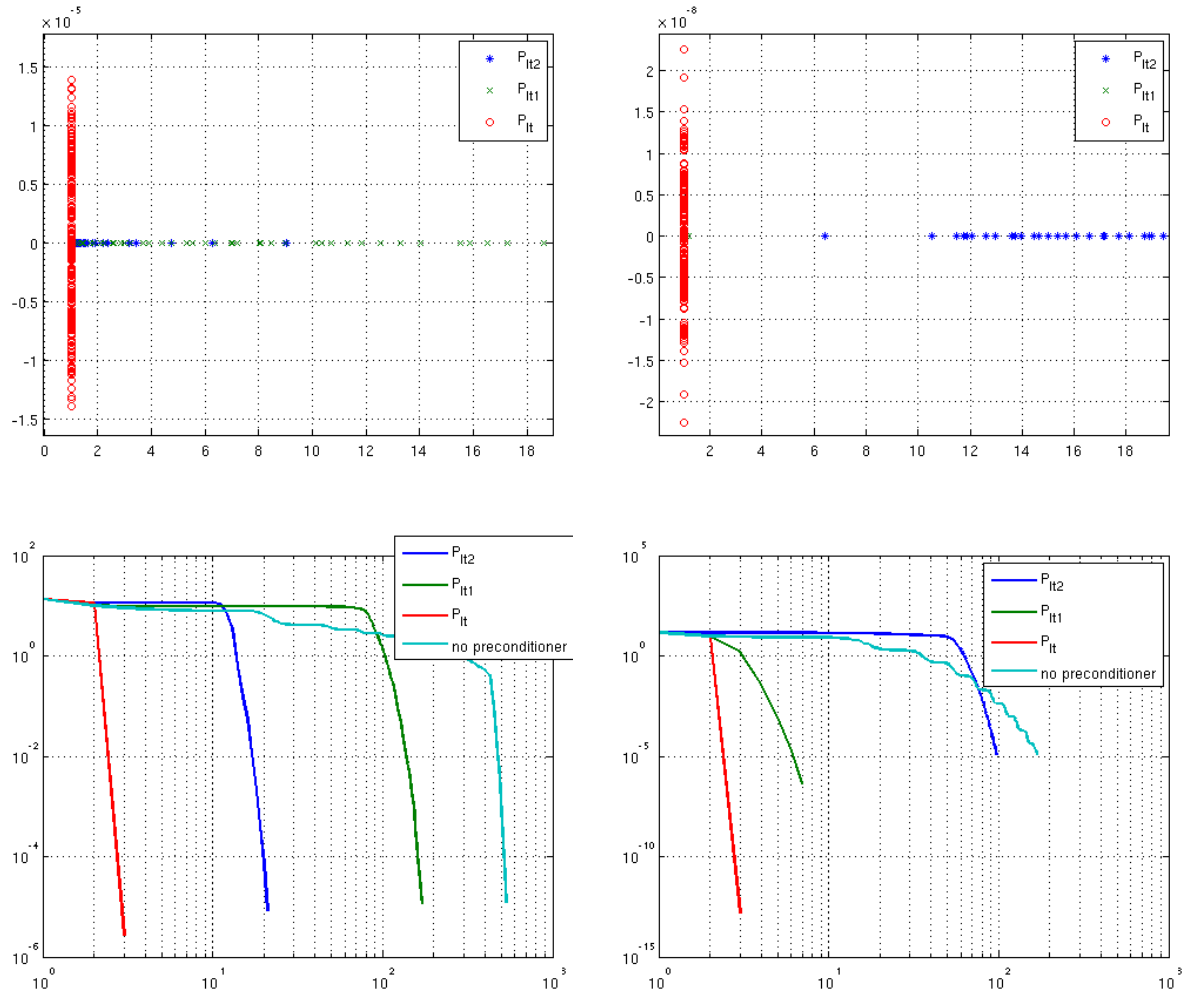


Figure A.4: The top two figures show the eigenvalue distributions in the complex plane for three different cases of block triangular preconditioners considered in Appendix A.4. The bottom two figures show convergence plots of GMRES for block diagonal preconditioners. **Left**: for $\phi = 10^{-6}$. **Right**: for $\phi = 10^{-2}$

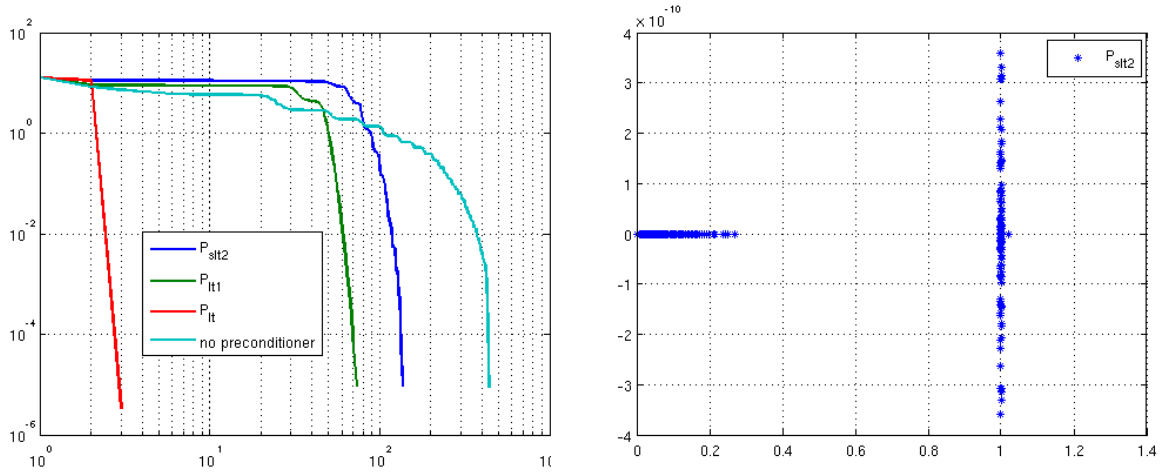


Figure A.5: The contrived example with $n = 200$ and $m = 100$ for block triangular preconditioners. **Left**: the convergence of GMRES for $\phi = 10^{-6}$. **Right**: the eigenvalue distributions in the complex plane for case 4

Bibliography

[1] Ken Alvin, Horacio M. de la Fuente, B. Haugen, and C. Felippa. Membrane triangles with corner drilling freedoms I. The EFF element. *Finite Elements in Analysis and Design*, 12:163–187, 1992.

[2] http://www.wired.com/dangerroom/2012/03/army-wings/.

[3] http://www.wired.com/dangerroom/2011/02/video-hummingbird-drone-can-perform-loops/.

[4] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific J. Math.*, 16:1–3, 1966.

[5] http://world.honda.com/ASIMO/.

[6] Ted Belytschko, Wing Kam Liu, and Brian Moran. *Nonlinear Finite Elements for Continua and Structures*. Wiley, 2000.

[7] Michele Benzi, Gene H. Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[8] J. T. Betts and W. P. Huffman. Sparse optimal control software, SOCS. Technical report, Boeing Information and Support Services, Seattle, Washington, July 1997.

[9] George Biros and Omar Ghattas. Parallel Lagrange-Newton-Krylov-Schur Methods for PDE-Constrained Optimization. Part I: The Krylov-Schur Solver. *SIAM Journal on Scientific Computing*, 27:687–713, 2005.

[10] George Biros and Omar Ghattas. Parallel Lagrange-Newton-Krylov-Schur Methods for PDE-Constrained Optimization. Part II: The Lagrange-Newton Solver and its Application to Optimal Control of Steady Viscous Flows. *SIAM Journal on Scientific Computing*, 27:714–739, 2005.

[11] Andrew Bradley and Walter Murray. Matrix-free approximate equilibration. submitted, 2012.

[12] Robert Bridson and Chen Greif. A multipreconditioned conjugate gradient algorithm. *J. on Matrix Analysis and Applications*, 27:1056–1068, 2006.

[13] Richard Byrd, Jorge Nocedal, and Robert Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Journal Mathematical Programming*, 63:129–156, 1994.

[14] Y. T. Chen. Iterative methods for linear least-squares problems. Technical report, University of Waterloo, 1975.

[15] Sou-Cheng Choi. *Iterative Methods for Singular Linear Equations and Least-Squares Problems.* PhD thesis, Stanford University, 2006.

[16] J. Chung and G. M. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method. *Journal of Applied Mechanics*, 60:371–375, 1993.

[17] S. S. Collis and M. Heinkenschloss. Analysis of the streamline upwind/petrov galerkin method applied to the solution of optimal control problems. Technical report, Department of Computational and Applied Mathematics, Rice University, 2002.

[18] H. A. Van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[19] H. Sue Dollar, Nicholas I. M. Gould, Martin Stoll, and Andrew J. Wathen. Preconditioning saddle-point systems with applications in optimization. *SIAM Journal on Scientific Computing*, 32:249–270, 2010.

[20] A.S. Drud. CONOPT - A Large-Scale GRG Code. *ORSA Journal on Computing*, pages 207–216, 1992.

[21] Howard C. Elman, David J. Silvester, and Andrew J. Wathen. *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics.* Oxford Science Publications, 2005.

[22] I. Kato et al. Pneumatically powered artificial legs walking automatically under various circumstances. In *Proceedings of 4th Int. Symposium in External Control of Human Extremities*, pages 458–470, 1972.

[23] D. K. Faddeev and V. N. Faddeeva. *Computational Methods of Linear Algebra.* W. H. Freeman and Co. San Francisco, 1963. Translated by Robert C. Williams.

[24] C. Farhat. *FEM Manual*, 0.1 edition, May 2010.

[25] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen. FETI-DP: a dual-primal unified FETI method. I. A faster alternative to the two-level FETI method. *Internat. J. Numer. Methods Engrg.*, 50:1523–1544, 2001.

[26] C. Farhat and F. X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32:1205–1227, 1991.

[27] C. A. Felippa and B. Haugen. A unified formulation of small-strain corotational finite elements: I. theory. *Computer Methods in Applied Mechanics and Engineering*, 194:2285–2335, 2005.

[28] R. Fletcher. Conjugate gradient methods for indefinite systems. *Numerical Analysis (Proc 6th Biennial Dundee Conf., Univ. Dundee, Dundee, 1975)*, 506:73–89, 1976. Lecture Notes in Math.

[29] D. C.-L. Fong and M. A. Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput*, 33:5:2950–2971, 2011.

[30] A. Forsgren, P. E. Gill, and J. D. Griffin. Iterative solution of augmented systems arising in interior methods. *SIAM J. Optim.*, 18:666–690, 2007.

[31] Anders Forsgren and Walter Murray. Newton methods for large-scale linear equality-constrained minimization. *SIAM J. Matrix Anal. Appl.*, 14(2):560–587, 1993.

[32] R. W. Freund. Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Stat. Comput.*, 13:425–448, 1992.

[33] R. W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14(2):470–482, 1993.

[34] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60(3):315339, 1991.

[35] R. W. Freund and N. M. Nachtigal. A new Krylov-subspace method for symmetric indefinite linear systems. In *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics*, pages 1253–1256, 1994.

[36] P. F. Gath and K. H. Well. Trajectory optimization using a combination of direct multiple shooting and collocation. In *AIAA Guidance, Navigation, and Control Conference*, August 2001.

[37] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic press, 1981.

[38] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

[39] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. Two steplength algorithms for numerical optimization. Technical report, Systems Optimization Laboratory, Stanford University, 1979.

[40] G. H. Golub and C. F. Van Loan. *Matrix Computations. 1996.* Johns Hopkins University Press, 1996.

[41] Chen Grief, Tyrone Rees, and Daniel B. Szyld. Multipreconditioned iterative methods for the solution of nonsymmetric linear systems of equations. Technical report, The University of British Columbia, December 2011.

[42] Max D. Gunzburger. *Perspectives in Flow Control and Optimization.* SIAM, 1987.

[43] Michael Hardt, Kenneth Kreutz-Delgado, and J. William Helton. Minimal energy control of a biped robot with numerical methods and a recursive symbolic dynamic model. In *Proceedings of the 37th IEEE conference on Decision and Control*, pages 413–416, 1998.

[44] C.R. Hargraves and S.W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance*, 10:338–342, 1987.

[45] Matthias Heinkenschloss. A time-domain decomposition iterative method for the solution of distributed linear quadratic optimal control problems. *Journal of Computational and Applied Mathematics*, 173:169–198, 2005.

[46] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[47] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of Honda humanoid robot. In *Proceedings of 4th Int. Symposium in External Control of Human Extremities*, pages 458–470, 1972.

[48] Thomas J. R. Hughes. *The finite element method: linear static and dynamic finite element analysis.* Dover Publications, 2000.

[49] Kaustuv. *IPSOL: An Interior Point Solver For Nonconvex Optimization Problems.* PhD thesis, Stanford University, 2008.

[50] Byounghyun Kim, Youngjoon Han, and Hernsoo Hahn. Lower energy gait pattern generation in 5-link biped robot using image processing. *World Academy of Science, Engineering and Technology 50*, pages 543–550, 2009.

[51] Philip A. Knight and Daniel Ruiz. A fast algorithm for matrix balancing. In Andreas Frommer, Michael W. Mahoney, and Daniel B. Szyld, editors, *Web Information Retrieval and Linear Algebra Algorithms*, number 07071 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[52] D. Lahaye, H. De Gersem, S. Vandewalle, and K. Hameyer. Algebraic multigrid for complex symmetric systems. *IEEE Transactions on Magnetics*, 36:1535–1538, 2000.

[53] Oren E. Livne and Gene H. Golub. Scaling by binormalization. *Numerical Algorithms*, 35:97–120, 2004.

[54] Tarek Mathew, Marcus Sarkis, and Christian Schaerer. Analysis of block parareal preconditioners for parabolic optimal control problems. *SIAM Journal on Scientific Computing*, 32:1180–1200, 2010.

[55] D. Q. Mayne and N. Maratos. A first order, exact penalty function algorithm for equality constrained optimization problems. *Math. Programming*, 16(3):303–324, 1979.

[56] Carmelo Militello and Carlos A. Felippa. The first andes elements: 9-dof plate bending triangles. *Computer Methods in Applied Mechanics and Engineering*, 93:217–246, 1991.

[57] Jorge J. More and David J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Software*, 20(3):286–307, 1994.

[58] Xiuping Mu and Qiong Wu. Synthesis of a composite sagittal gait cycle for a five-link biped robot. *Robotica*, 21:581–587, 2003.

[59] Malcolm F. Murphy, Gene H. Golub, and Andrew J. Wathen. A note on preconditioning for indefinite linear systems, 1999.

[60] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[61] Y. Notay. Flexible conjugate gradients. *SIAM J. Sci. Comput.*, 22:1444–1460, 2000.

[62] H. J. Oberle and W. Grimm. *BNDSCO-A Program for the Numerical Solution of Optimal Control Problems*. Institute for Flight Systems Dynamics, 1989.

[63] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Nuer. Anal.*, 12:617–624, 1975.

[64] C. C. Paige and M. A. Saunders. Algorithm 583; LSQR: Sparse linear equations and least-squares problems. *ACM Trans. Math. Software*, 8(2):195–209, 1982.

[65] C. C. Paige and M. A. Saunders. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71, 1982.

[66] John W. Pearson and Andrew J. Wathen. A new approximation of the Schur complement in preconditioners for PDE-constrained optimization. *Numerical Linear Algebra with Applications*, pages 816–829, 2012.

[67] Tomasz Pietrzykowski. An exact potential method for constrained maxima. *SIAM J. Numer. Anal.*, 6:299–304, 1969.

[68] Ernesto Prudencio, Richard Byrd, and Xiao-Chuan Cai. Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems. *SIAM Journal on Scientific Computing*, 27:13051328, 2006.

[69] Alfio Quarteroni and Alberto Valli. *Numerical Approximation of Partial Differential Equations*, volume 23. Springer-Verlag, 1994.

[70] A. V. Rao, D. A. Benson, G. T. Huntington, C. Francolin, C. L. Darby, and M. A. Patterson. User's manual for GPOPS: A MATLAB package for dynamic optimization using the gauss pseudospectral method. Technical report, University of Florida, August 2008.

[71] Tyrone Rees, H. Sue Dollar, and Andrew J. Wathen. Optimal solvers for PDE-constrained optimization. *SIAM J. on Scientific Computing*, 32:271–298, 2010.

[72] I. M. Ross and F. Fahroo. User's manual for DIDO: A MATLAB package for dynamic optimization. Technical report, Dept. of Aeronautics and Astronautics, Naval Postgraduate School, 2002.

[73] P. Rutquist and M. Edvall. PROPT - MATLAB optimal control software. Technical report, Tomlab Optimization, Inc, 2010.

[74] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 1986.

[75] Youcef Saad. A flexible inner-outer preconditioned gmres algorithm, 1993.

[76] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. PWS, 1993.

[77] Adam Schwartz. *Theory and Implementation of Methods based on Runge-Kutta Integration for Solving Optimal Control Problems*. PhD thesis, University of California at Berkeley, 1996.

[78] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10(1):36–52, 1989.

[79] Martin Stoll and Andy Wathen. All-at-once solution of time-dependent pde-constrained optimization problems. Technical report, University of Oxford, 2011.

[80] Martin Stoll and Andy Wathen. Combination preconditioning and the Bramble-Pasciak$^+$ preconditioner. *SIAM Journal on Matrix Analysis and Applications*, 2011.

[81] Aye Aye Thant and Khaing Khaing Aye. Application of cubic spline interpolation to walking patterns of biped robot. *World Academy of Science, Engineering and Technology*, 2009.

[82] Stefan Ulbrich. Generalized SQP methods with "parareal time-domain decomposition for time-dependent pde-constrained optimization. In Lorenz T. Biegler, Omar Ghattas, Matthias Heinkenschloss, David Keyes, and Bart van Bloemen Waanders, editors, *Real-Time PDE-Constrained Optimization*, chapter 7, pages 145–168. SIAM, 2007.

[83] A. van der Sluis. Condition numbers and equilibration of matrices. *Numer. Math.*, 14:14–23, 1969.

[84] M. Vasile, F. Bernelli-Zazzera, N. Fornasari, and P. Masarati. Design of interplanetary and lunar missions combining low-thrust and gravity assists. Technical report, ESA/ESOC, September 2002.

[85] Oskar von Stryk. User's guide for DIRCOL (version 2.1): A direct collocation method for the numerical solution of optimal control problems. Technical report, Technische Universitt Darmstadt, 1999.