Preconditioning Iterative Methods for PDE Constrained Optimization

Tyrone Rees

New College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2010

This thesis is dedicated to my wife, Kristal.

Acknowledgements

First and foremost I would like to thank my supervisor, Andy Wathen. It was through his lectures I was first introduced to numerical analysis as an undergraduate, and later as his DPhil student he initiated me in the rich field of numerical linear algebra and PDE-constrained optimization. He has been a constant source of support both academically and personally. I would also like to thank my co-authors, Martin Stoll and Sue Thorne, with whom it has been a pleasure to work. Having these three people to bounce ideas off and get feedback from has enabled me to write this thesis.

I would like to thank the EPSRC for funding my research. I am especially grateful to all organizations and bodies that helped me travel to conferences and research visits. These include New College, Oxford Computing Laboratory, Oxford Mathematical Institute, WIAS Berlin, the Copper Mountain conference on Iterative Methods, SIAM, BIRS and the LMS. I am particularly grateful to the organizers of the first Elgersburg School on Mathematical Systems Theory, and to Volkswagen Stiftung, who funded my place. It was here that I learned much of the material that would go on to form the first chapter of this thesis.

I would like to thank the Numerical Analysis group here in Oxford for making the department such a stimulating place to study. I have got to talk with so many inspiring people as they pass through Oxford. A special mention has to go to the students in the department – in particular my office mates, Siobhan Burke, Nick Hale and Ricardo Pachón, who have always been on hand to answer a 'quick question'.

Last, but certainly not least, I have to thank my family. Without my parents' constant love and support I would certainly not be where I am today, and for that I cannot thank them enough. Finally I must thank my wife, Kristal, for being amazing and a constant source of strength. She has had to put up with me working late, being absent at conferences, and

missing two Easters over the last four years. She deserves an extra special thanks for spending our first wedding anniversary at a maths conference!

Abstract

Methods for solving PDE-constrained optimization problems generally require the solution of a large linear system in saddle point form. This system is sparse, symmetric and indefinite, so it is amenable to solution by certain Krylov subspace methods, such as MINRES or modified versions of Conjugate Gradients. Such methods need good preconditioners to be effective.

In this thesis we describe block preconditioners, which make use of our knowledge of the individual blocks that make up the system. These are based on approximations to the mass matrix and the Schur complement. We show that a fixed number of steps of the Chebyshev semi-iteration is spectrally equivalent to the mass matrix, so can be used in the preconditioner. The only component that is needed to approximate the Schur complement is, at least for larger values of the regularization parameter, a suitable preconditioner for the forward PDE problem.

We explicitly describe preconditioners to solve control problems with three PDEs – Poisson's equation, the convection-diffusion equation and the Stokes equations – and consider both distributed and boundary control. We prove the effectiveness of the methods proposed by showing eigenvalue bounds for the preconditioned system, and also give a range of numerical examples.

Contents

List of Figures

Chapter 1

Introduction

Let Ω denote some body of mass. Suppose that we have some desired heat distribution, or desired state, which we will denote $\hat{y}$, which we would like the mass to possess. Suppose the heat distribution satisfies a partial differential equation,

$$\mathcal{L}y = u,$$

and we have some mechanism of heating the body – for example, microwaves or electromagnetic induction. Mathematically, this means that we are free to choose the forcing term, u, on the right hand side of the PDE. A practial situation would be, for example, a tumor in a body; here we would like to heat the tumor to a high enough temperature to cure it, yet keep the tissue surrounding the tumor at a low temperature. The question is, how do we choose the heat to apply to the body – i.e. the function u – to give us our desired distribution.

The answer to this question depends on which sense we mean that our state y is close to the desired state $\hat{y}$? If we take this as being in the $L^2(\Omega)$ norm – which makes the theory work well, as we shall see in Chapter 2 – the question can be formulated as the following minimization problem:

$$\begin{gathered}\min_y \frac{1}{2}\|y-\hat{y}\|^2_{L^2(\Omega)} \\ \text{s.t. } \mathcal{L}y = u.\end{gathered}$$

We shall see in Chapter 2 that this problem is, in general, ill-posed. We can make the solution well-defined by the addition of the norm of the control, and a Tychonoff regularization parameter to the cost functional, so our problem becomes

$$\begin{gathered}\min_{y,u} \frac{1}{2}\|y-\hat{y}\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} \\ \text{s.t. } \mathcal{L}y = u.\end{gathered}$$

Since u is now included in the cost functional its size in the L^2 norm is prevented from becoming too large, and u has better regularity properties. We call the functions u and y that acheive this minimum the optimal control and optimal state respectively.

The problem above is called a *distributed control* problem, as the quantity you can change is distributed throughout the domain Ω. A related problem is *boundary control*, where you can only change the boundary conditions of the PDE.

The mathematical foundations for problems of this type were set down by J.L. Lions in the 1960s [79]. Many problems in many fields can be formulated in this way – there are examples in medicine [2, 76], aerodynamics [46, 105] and mathematical finance [14, 36] to name but a few. It is therefore very important that we have good techniques for solving such problems. Our aim was to develop methods which are factorization free, and for which the time needed to solve such problems scales linearly with the problem size.

In Chapter 2 we review some of the theory behind such control problems. We describe the finite element method for solving PDEs numerically, and show how applying this method to a PDE-constrained optimization problem leads to a discrete optimality system which is in saddle point form.

Chapter 3 describes some of the standard iterative methods for solving linear systems of equations. These are presented as linear methods – such as Jacobi iteration, and the Chebyshev semi-iteration – and non-linear methods, such as Krylov subspace methods. We also give a brief introduction to multigrid methods.

We look at solving systems with saddle point structure in Chapter 4. We again consider linear methods – such as the inexact Uzawa method – and non-linear Krylov subspace methods. This chapter includes an original derivation of bounds for the eigenvalues, λ, satisfying the generalized eigenvalue problem

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} C & 0 \\ B & D \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}.$$

These confirm those previously reported in the literature for the case when $A-C > 0$ – i.e. when the eigenvalues are real. The bounds proved for the general case differ from those already published, and could offer new insights.

Chapter 5 introduces the paradigm that is the main contribution of this thesis: we can develop optimal block preconditioners for PDE constrained optimization problems by using (suitable) preconditioners for the forward problem and an approximation to the mass matrix. We show that the Chebyshev semi-iteration gives a good approximation to the mass matrix in block preconditioners of this type. We present

eigenvalue bounds to justify our claims, as well as numerical examples for a variety of control problems where the constraint is Poisson's equation.

In Chapter 6 we look at a PDE that is not self-adjoint – the convection diffusion equation. Here we have to deal with the question of whether to optimize first and then discretize, or do the discretization before the optimization. We give numerical results, based on the methods introduced in the previous chapter, for both techniques.

Finally we will look at a flow control problem in Chapter 7, where the constraints are the Stokes equations. Again, the methods introduced in Chapter 5 can be applied, although in this case we must be careful with the approximation to the PDE that we use in our preconditioner. We show theoretically that we can derive effective preconditioners for this problem, and give numerical results which support our theory.

Chapter 2

Optimal control of PDEs: theoretical background

2.1 The Finite Element method

Before we consider how to solve the control problem introduced in Chapter 1, we give a brief review of how to discretize the forward problem using the finite element method. Suppose we want to solve Laplace's equation in some domain Ω,

$$-\nabla^2 u = f, \tag{2.1}$$

together with mixed boundary conditions

$$u = g_1 \text{ on } \partial\Omega_D, \quad \frac{\partial u}{\partial n} = g_2 \text{ on } \partial\Omega_N, \tag{2.2}$$

where $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$, $\partial\Omega_D \cap \partial\Omega_N = \varnothing$, and $\partial u/\partial n$ denotes the directional derivative in the direction normal to $\partial\Omega$. We call boundary conditions of the type on $\partial\Omega_D$ Dirichlet boundary conditions, and those on $\partial\Omega_N$ Neumann boundary conditions.

Let $\mathcal{H}^1(\Omega)$ be the Sobolev space of functions u such that $u \in L^2(\Omega)$ and it possesses a weak first derivative. Then, if we define solution and test spaces as

$$\mathcal{H}^1_E = \{u \in \mathcal{H}^1(\Omega) | u = g_D \text{ on } \partial\Omega_D\}$$
$$\mathcal{H}^1_{E_0} = \{v \in \mathcal{H}^1(\Omega) | v = 0 \text{ on } \partial\Omega_D\}$$

the weak formulation of (2.1) and (2.2) is:
Find $u \in \mathcal{H}^1_E(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega v f + \int_{\partial\Omega_N} v g_2 \quad \forall\ v \in \mathcal{H}^1_{E_0}.$$

We refer the reader to the books by Evans [42] or Adams and Fournier [1], for example, for more detail about weak formulations of PDEs and Sobolev spaces.

Assume that $V_0^h \subset \mathcal{H}^1_{E_0}$ is an n-dimensional vector space of test functions with $\{\phi_1, ..., \phi_n\}$ as a basis. In order to satisfy the Dirichlet boundary condition on $\partial\Omega_D$ we extend the basis by defining functions $\phi_{n+1}, \ldots, \phi_{n+n_\partial}$ and coefficients U_j such that $\sum_{j=n+1}^{n+n_\partial} U_j\phi_j$ interpolates the boundary data. We define

$$V_E^h := \text{span}\{\phi_1, \ldots, \phi_n\} + \sum_{j=n+1}^{n+n_\partial} U_j\phi_j.$$

Then, if $u_h \in V_E^h$, it is uniquely determined by $\mathbf{u} = (U_1 \cdots U_n)^T$ in

$$u_h = \sum_{j=1}^{n} U_j\phi_j + \sum_{j=n+1}^{n+n_\partial} U_j\phi_j. \tag{2.3}$$

The functions in the first sum on the right hand side of (2.3) are called the trial functions. We have therefore constructed a space V_E^h, of which the trial functions and the basis of the set of test functions V_0^h coincide. This is referred to as the *Galerkin* method. If you choose an approximation where the trial and test spaces are different you get a *Petrov-Galerkin* method; we discuss an example of such a method in Chapter 6.

Using the Galerkin approximation we can write the finite-dimensional version of the weak formulation: find $u_h \in V_E^h$ such that

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega v_h f + \int_{\partial\Omega_N} v_h g_2 \quad \forall\ v_h \in V_0^h.$$

Since the ϕ_i form a basis, this is the same as finding U_j, $j = 1, \ldots, n$ such that

$$\sum_{j=1}^{n} U_j \int_\Omega \nabla\phi_j \cdot \nabla\phi_i = \int_\Omega \phi_i f + \int_{\partial\Omega_N} \phi_i g_2 - \sum_{j=n+1}^{n+n_\partial} U_j \int_\Omega \nabla\phi_j \cdot \nabla\phi_i \quad \forall\ v_h \in V_0^h,$$

for $i = 1, \ldots, n$, where we have taken the known coefficients $U_{n+1}, \ldots, U_{n+n_\partial}$ to the right hand side. This is then equivalent to solving the linear system

$$K\mathbf{u} = \widehat{\mathbf{f}}, \tag{2.4}$$

where

$$K = [k_{i,j}] \in \mathbb{R}^{n\times n},\ k_{i,j} = \int_\Omega \nabla\phi_j \cdot \nabla\phi_i$$

$$\widehat{\mathbf{f}} = [f_i] \in \mathbb{R}^n,\ f_i = \int_\Omega \phi_i f + \int_{\partial\Omega_N} \phi_i g_2 - \sum_{j=n+1}^{n+n_\partial} U_j \int_\Omega \nabla\phi_j \cdot \nabla\phi_i.$$

The matrix K is commonly referred to as the *stiffness matrix*. It is clear from the definition that this is symmetric. Moreover, let $\mathbf{v}$ be any vector in $\mathbb{R}^n$. Then

$$\begin{aligned}\mathbf{v}^T K \mathbf{v} &= \sum_{j=1}^{n}\sum_{i=1}^{n} \mathbf{v}_j \left(\int_\Omega \nabla\phi_j \cdot \nabla\phi_i\right) \mathbf{v}_i \\ &= \int_\Omega \left(\sum_{j=1}^{n} \mathbf{v}_j \nabla\phi_j\right) \cdot \left(\sum_{i=1}^{n} \mathbf{v}_i \nabla\phi_i\right) \\ &= \int_\Omega \nabla v_h \cdot \nabla v_h \\ &\geq 0,\end{aligned}$$

for $v_h := \sum_{i=1}^{n} \mathbf{v}_i \phi_i \in V_0^h$. Therefore K is at least positive semi-definite. Now suppose $\mathbf{v}^T K \mathbf{v} = 0$. Then this must mean that $\nabla v_h = 0$, i.e. v_h must be a constant function. Since $v_h \in V_0^h$, it is zero on $\partial\Omega_D$. Since the function is continuous up to the boundary, and as long as $\partial\Omega_D \neq \varnothing$, we must have that $v_h = 0$, and hence $\mathbf{v} = \mathbf{0}$. Therefore if the problem has a Dirichlet boundary condition applied to part of the boundary, the stiffness matrix K is positive definite.

In the case of a purely Neumann boundary condition, this argument will not work. In this case, K will be only semi-definite, and will have a null-space of dimension one corresponding to the constant functions. If, as is usual, our trial functions define a partition of unity – i.e. the sum of the trial functions is identically one – then this corresponds to the constant vectors. Therefore, in order for the purely Neumann problem to have a solution, we require that $\mathbf{1}^T \widehat{\mathbf{f}} = \mathbf{0}$, where $\mathbf{1} = [1, \ldots, 1]^T$. This is equivalent to the necessary condition for the continuous solution to exist, namely

$$\int_\Omega f + \int_{\partial\Omega} g_2 = 0.$$

Let us return to (2.4). Calculating the right hand side, $\widehat{\mathbf{f}}$, requires us to evaluate integrals of the form $\int f\phi_i$. In general, calculating these integrals exactly would be infeasible, so one way to do this would be using numerical quadrature. Another way, which we will now explore, would be to discretize the right hand side function, f, using the trial space. Then we'd have, by interpolation or otherwise,

$$f_h = \sum_{i=1}^{n} F_i \phi_i.$$

Then we can write (2.4) in the form

$$K\mathbf{u} = Q\mathbf{f} + \mathbf{d}, \tag{2.5}$$

where $\mathbf{f} = [F_1, \ldots, F_n]^T$, $\mathbf{d}$ contains the boundary data and

$$Q = [q_{i,j}] \in \mathbb{R}^{n \times n},\ q_{i,j} = \int_\Omega \phi_j \phi_i.$$

The matrix Q is usually called the *mass matrix*. As with the stiffness matrix, this is clearly symmetric and for any $\mathbf{v} \in \mathbb{R}^n$,

$$\mathbf{v}^T Q \mathbf{v} = \sum_{j=1}^{n} \sum_{i=1}^{n} \mathbf{v}_j \left(\int_\Omega \phi_j \phi_i \right) \mathbf{v}_i = \int_\Omega v_h v_h \geq 0,$$

where $v_h = \sum_{j=1}^{n} \mathbf{v}_i \phi_i$. It is clear that $\mathbf{v}^T Q \mathbf{v} = 0 \iff v_h = 0$, hence $\mathbf{v} = \mathbf{0}$. Therefore the mass matrix too is symmetric positive definite.

Now we just have to choose basis functions ϕ_i. If we pick functions with a small support, then most of the entries in K and M will be zero, making these sparse matrices, hence easier to solve computationally. It is well known that any smooth function can be approximated to arbitrary accuracy using piecewise polynomials [39, p 20], so this is where we look for our basis.

Consider first the case where the domain Ω is square. Then we can simply divide it into smaller squares, the corners of which are some distance h apart[1]. We want to define our basis elements such that if the nodes are at positions $\mathbf{x}_j$, $j = 1, \ldots, n$, then

$$\phi_i(\mathbf{x}_j) = \delta_{i,j},$$

where δ denotes the Kronecker delta. We can define bilinear polynomials of the form $(ax + b)(cy + d)$, where $a,\ b,\ c,\ d \in \mathbb{R}$, that are one at each of the corners, and zero at the others. For example, for the square $[0, h] \times [0, h]$ such functions would be

$$(1 - x/h)(1 - y/h),\ x/h(1 - y/h),\ xy/h^2,\ (1 - x/h)y/h.$$

Then, at each node, we simply patch together the relevant functions from the four squares the touch that node, giving the basis function which is made out of four piecewise-bilinear functions. A regular $\mathbf{Q}_1$ grid is shown in Figure 2.1.

Arbitrary quadrilaterals can be treated in essentially the same way – see e.g. [39, Section 1.3] for details. Such elements are known as $\mathbf{Q}_1$ elements. The ideas above generalize to three dimensions – here we have $\mathbf{Q}_1$ 'brick' elements.

We can get a more accurate approximation by using higher-order elements – that is, by using functions that are piecewise biquadratic, for example. This would be known as a $\mathbf{Q}_2$ approximation.

[1] More generally we can spilt the domain into rectangles – the discussion here generalizes in the obvious way.

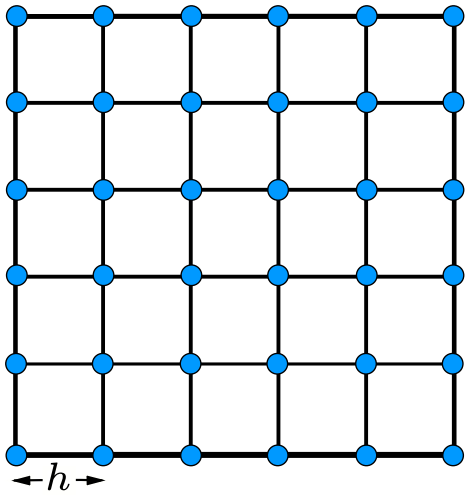


Figure 2.1: A $\mathbf{Q}_1$ grid.

For less convenient domains, it is usual to use a triangular mesh – known as a triangulation. We call bases with triangles (or tetrahedra) that correspond to those defined with rectangles (or bricks) above $\mathbf{P}_1$ and $\mathbf{P}_2$ elements. We will only use quadrilateral elements here – we refer to [39, Section 1.3] for a more detailed discussion about other possibilities.

We now turn our attention to the calculation of the stiffness matrix. This is usually done by working out the stiffness matrix on a standard element and then assembling all the element stiffness matrices together into a much bigger matrix. Let n_e be the number of nodes on an element – for example, in 2D n_e is four if using $\mathbf{Q}_1$ elements – and, as above, let n be the number of nodes corresponding to unknown coefficients in the triangulation. Let an element stiffness matrix on element e be labelled $K_e \in \mathbb{R}^{n_e \times n_e}$. If we use a regular $\mathbf{Q}_1$ discretization with mesh size h, as shown in Figure 2.1, then the element stiffness matrix is given by

$$K_e = \frac{1}{6} \begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}.$$

Now consider the Boolean matrix $L_e \in \mathbb{R}^{n_e \times n}$ which maps a vector from the global nodal variables to the (local) nodal variables on the e^{th} element. Then, if we have N

elements overall, we can write the stiffness matrix as

$$\begin{aligned} K &= \sum_{i=1}^{N} L_e^T K_e L_e \\ &= \begin{bmatrix} L_1^T & \dots & L_N^T \end{bmatrix} \begin{bmatrix} K_1 & & 0 \\ & \ddots & \\ 0 & & K_N \end{bmatrix} \begin{bmatrix} L_1 \\ \vdots \\ L_N \end{bmatrix} \\ &:= L^T \text{blkdiag}(K_e) L. \end{aligned}$$

Note that the matrix L, which is made up of the element transfer matrices, has at most one non-zero entry in each row.

The mass matrix is assembled in the same way - and so we can write

$$Q = L^T \text{blkdiag}(Q_e) L, \tag{2.6}$$

where Q_e is the element mass matrix on element e. For a $\mathbf{Q}_1$ discretization with mesh size h this is given by

$$Q_e = \frac{h^2}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix}.$$

Now that we have defined possible bases for V_E^h we would like to say more about the eigenvalues of the stiffness and mass matrices than we could in the general case. First, let $\mathcal{T}_h$ be a set of triangles or quadrilaterals that tessellate Ω. Define $\underline{h} := \min_{\Delta_k \in \mathcal{T}_h} h_k$ and $h := \max_{\Delta_k \in \mathcal{T}_h} h_k$, where h_k denotes the longest edge of the triangle. Then a sequence of such grids $\{\mathcal{T}_h\}$ is said to be *quasi-uniform* if there exists a constant $\rho > 0$ such that $\underline{h} \geq \rho h$ for every grid in the sequence. Also, $\{\mathcal{T}_h\}$ is said to be *shape-regular* if there exists a minimum angle $\theta_* \neq 0$ such that every element in $\mathcal{T}_h$ has it's smallest angle greater than or equal to θ_*.

We are now in a poisition to state the following two theorems, which are proved in [39].

Theorem 2.1.1. *[39, Theorem 1.32] For $\mathbf{Q}_m$ or $\mathbf{P}_m$ approximation on a shape regular, quasi-uniform subdivision of $\mathbb{R}^2$, the Galerkin matrix K in (2.5) satisfies*

$$c_m h^2 \leq \frac{\mathbf{v}^T K \mathbf{v}}{\mathbf{v}^T \mathbf{v}} \leq C_m \ \forall \ \mathbf{v} \in \mathbb{R}^n,$$

where h is the length of the longest edge in the grid, and c_m and C_m are positive constants independent of h, but depending on m.

In three dimensions, under the same assumptions, the corresponding bound is

$$c_m h^3 \leq \frac{\mathbf{v}^T K \mathbf{v}}{\mathbf{v}^T \mathbf{v}} \leq C_m h \ \forall \ \mathbf{v} \in \mathbb{R}^n.$$

Theorem 2.1.2. *[39, Theorem 1.29] For $\mathbf{Q}_m$ or $\mathbf{P}_m$ approximation on a shape regular, quasi-uniform subdivision of $\mathbb{R}^2$, the mass matrix Q in (2.5) approximates the scaled identity matrix in the sense that*

$$c_m h^2 \leq \frac{\mathbf{v}^T Q \mathbf{v}}{\mathbf{v}^T \mathbf{v}} \leq C_m h^2 \ \forall \ \mathbf{v} \in \mathbb{R}^n,$$

where h is the length of the longest edge in the grid, and c_m and C_m are positive constants independent of h, but depending on m.

In three dimensions, under the same assumptions, the corresponding bound is

$$c_m h^3 \leq \frac{\mathbf{v}^T Q \mathbf{v}}{\mathbf{v}^T \mathbf{v}} \leq C_m h^3 \ \forall \ \mathbf{v} \in \mathbb{R}^n.$$

We briefly mention the accuracy of finite element methods as presented here. Any error analysis is dependent on the norm in which one would like to measure convergence. Deriving error bounds for the Dirichlet problem where $g_1 \neq 0$ – the inhomogeneous problem – requires a non-conforming analysis since the boundary condition is interpolated, and is technical – see, for example, Brenner and Scott for details [24]. In the homogeneous case, where $g_1 = 0$, it can be shown, for example, that

$$\|u - u_h\|_{\mathcal{H}^1(\Omega)} \leq ch\|f\|_2. \tag{2.7}$$

See [19, Theorem 7.5] for a proof. If we know more about the regularity of u this can be strengthened. For example, if $f \in L^2(\Omega)$, so $u \in \mathcal{H}^2(\Omega)$, then

$$\|u - u_h\|_2 \leq Ch^2\|u\|_2. \tag{2.8}$$

This is Corollary 7.7 in Braess [19].

The discussion in this section broadly follows the approach presented in [39, Chapter 1]. For more details – including more results on the accuracy of such methods – we refer the reader to, for example, the books by Braess [19], Brenner and Scott [24] or Ciarlet [28].

2.2 Distributed control

Recall from Chapter 1 that we are interested in solving problems of the form

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||_2^2 + \frac{\beta}{2}||u||_2^2$$

$$\text{s.t.} \qquad \mathcal{L}y = u \text{ in } \Omega$$
$$y = f_1 \text{ on } \partial\Omega_D, \quad \frac{\partial y}{\partial n} = f_2 \text{ on } \partial\Omega_N.$$

We will introduce the theory for a specific application of the problem, namely where $\mathcal{L} = -\nabla^2$, $\partial\Omega_D = \partial\Omega$ and $f_1 = 0$. We look for a control $u \in L^2(\Omega)$ and $y \in \mathcal{H}_0^1(\Omega)$. This problem is given explicitly as

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||_2^2 + \frac{\beta}{2}||u||_2^2 \tag{2.9}$$

$$\text{s.t.} \qquad -\nabla^2 y = u \text{ in } \Omega \tag{2.10}$$
$$y = 0 \text{ on } \partial\Omega. \tag{2.11}$$

If we want to solve such a problem numerically, it is clear that we will have to discretize the quantities involved at some point. Here there are two schools of thought – do you derive the optimality conditions first, and then discretize from there (*optimize-then-discretize*) or do you discretize the cost functional and the PDE first and then optimize that (*discretize-then-optimize*)? This can be an important choice, e.g. Collis and Heinkenschloss in [29] show that you can get different answers depending on which method you choose. As we shall see, both approaches are equivalent when looking at the problem just introduced, but this will not always be the case – for example, see Chapter 6.

2.2.1 Discretize-then-optimize

First, let's consider the discretize-then-optimize method. Here we discretize the PDE using the finite element method. Let $\{\phi_i\}$ be some set of basis functions such that $\text{span}\{\phi_1, \ldots, \phi_n\} = Y_0^h \subset \mathcal{H}_0^1$. Then we look for a finite dimensional approximation to y, $y_h = \sum_{i=1}^n Y_i\phi_i \in Y_0^h$. We call the vector of unknown coefficients $\mathbf{y} := (Y_1, \ldots, Y_n)$.

We also need to discretize the control, u. Let $\{\psi_i\}$ be some basis – generally, but not necessarily, different from the ϕ_i – such that $\psi_i \in L^2(\Omega)$. Then we can define $U^h := \text{span}\{\psi_1, \ldots, \psi_m\}$. The discretized control u_h is therefore uniquely determined by the vector of coefficients $\mathbf{u} = (U_1, \ldots, U_m)$.

The discrete analogue of the minimization problem (2.9)-(2.11) is therefore

$$\min_{y_h, u_h} \frac{1}{2}||y_h - \hat{y}||_2^2 + \frac{\beta}{2}||u_h||_2^2 \tag{2.12}$$

$$\text{s.t.} \quad \int_\Omega \nabla y_h \cdot \nabla v_h = \int_\Omega u_h v_h \quad \forall\ v_h \in Y_0^h. \tag{2.13}$$

As we saw in the previous section, (2.13) can be written as

$$K\mathbf{y} = \widehat{Q}\mathbf{u},$$

where K is the stiffness matrix, and here

$$\widehat{Q} = [\widehat{q}_{i,j}] \in \mathbb{R}^{n\times m},\ \widehat{q}_{i,j} = \int_\Omega \phi_i \psi_j.$$

Also, we have:

$$\begin{aligned} ||y_h - \hat{y}||_2^2 &= \int_\Omega (y_h - \hat{y})^2 \\ &= \sum_i \sum_j Y_i Y_j \int_\Omega \phi_i \phi_j - 2\sum_j Y_j \int_\Omega \phi_j \hat{y} + \int_\Omega \hat{y}^2 \\ &= \mathbf{y}^T Q_y \mathbf{y} - 2\mathbf{y}^T \mathbf{b} + C \end{aligned}$$

where

$$Q_y = \left[q_{i,j}^y\right] \in \mathbb{R}^{n\times n},\ q_{i,j}^y = \int_\Omega \phi_i \phi_j,$$

$$\mathbf{b} = [b_i] \in \mathbb{R}^n,\ b_i = \int_\Omega \hat{y}\phi_i,$$

and C is a constant. We call Q_y the state mass matrix. Similarly,

$$||u_h||_2^2 = \mathbf{u}^T Q_u \mathbf{u}$$

where

$$Q_u = \left[q_{i,j}^u\right] \in \mathbb{R}^{m\times m},\ q_{i,j}^u = \int_\Omega \psi_i \psi_j,$$

which we call the control mass matrix.

Solving the discretized equations (2.12)-(2.13) is therefore equivalent to solving

$$\min_{\mathbf{y},\mathbf{u}} \frac{1}{2}\mathbf{y}^T Q_y \mathbf{y} - \mathbf{y}^T \mathbf{b} + \frac{\beta}{2}\mathbf{u}^T Q_u \mathbf{u} \tag{2.14}$$

$$\text{s.t. } K\mathbf{y} = \widehat{Q}\mathbf{u}. \tag{2.15}$$

We will sometimes refer to this cost function as $J(\mathbf{y}, \mathbf{u}) := \frac{1}{2}\mathbf{y}^T Q_y \mathbf{y} - \mathbf{y}^T \mathbf{b} + \frac{\beta}{2}\mathbf{u}^T Q_u \mathbf{u}$. Theorem 2.1.1 tells us that K is invertible, and so $\mathbf{y} = K^{-1}\widehat{Q}\mathbf{u}$. Therefore we can

think of $\mathbf{y}$ as a function of $\mathbf{u}$, and so $J(\mathbf{y}, \mathbf{u}) =: F(\mathbf{u})$, and our problem can be re-phrased as

$$\min_{\mathbf{u}} F(\mathbf{u}).$$

It is well known – see, e.g. [89, Theorem 12.2], [68, Theorem 1.93] – that if $\mathbf{u}^*$ is optimal for problems such as this, then $\mathbf{u}^*$ must satisfy the variational inequality

$$\nabla(F(\mathbf{u}^*)) \cdot (\mathbf{u} - \mathbf{u}^*) \geq 0 \quad \forall\ \mathbf{u} \in \mathbb{R}^m,$$

which tells us that $\nabla(F(\mathbf{u}^*)) = \mathbf{0}$.

For the cost functional defined above, we have

$$\nabla(F(\mathbf{u})) = \widehat{Q}^T K^{-T}(Q_y K^{-1}\widehat{Q}\mathbf{u} - \mathbf{b}) + \beta Q_u \mathbf{u} = \widehat{Q}^T K^{-T}(Q_y \mathbf{y} - \mathbf{b}) + \beta Q_u \mathbf{u}. \quad (2.16)$$

If we introduce a new variable $\mathbf{p} := -K^{-T}(Q_y\mathbf{y} - \mathbf{b})$, then the optimality condition becomes

$$(\beta Q_u \mathbf{u} - \widehat{Q}^T \mathbf{p}) = \mathbf{0}. \quad (2.17)$$

We call the vector $\mathbf{p}$ the adjoint state, since it satisfies

$$K^T \mathbf{p} = \mathbf{b} - Q_y \mathbf{y}. \quad (2.18)$$

The system that needs to be solved to find the minimum is therefore given by equations (2.15,2.18,2.17), and can be written as

$$\begin{bmatrix} \beta Q_u & 0 & -\widehat{Q}^T \\ 0 & Q_y & K^T \\ -\widehat{Q} & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \quad (2.19)$$

Notice that the cost function is convex by construction, so the critical point found by solving this system will always be a minimum.

Since Q_u and Q_y are symmetric, this matrix is itself symmetric. Also, in the case where the control is Poisson's equation, as long as we use a Galerkin finite element method – i.e. the trial and test spaces are the same - K is symmetric, so $K^T = K$. In the case where we discretize the control and the state using the same bases, then the equation will simplify to

$$\begin{bmatrix} \beta Q & 0 & -Q \\ 0 & Q & K \\ -Q & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \\ \mathbf{0} \end{bmatrix}, \quad (2.20)$$

so $Q := Q_u = Q_p = \widehat{Q}$. Note in particular that, in this case, the generally rectangular matrix $\widehat{Q}$ will be square, and invertible.

The matrices above have what is called a *saddle point* structure, that is they are of the form

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix},$$

where, in the general case, $A = \begin{bmatrix} \beta Q_u & 0 \\ 0 & Q_y \end{bmatrix}$ and $B = [-\widehat{Q}\ K]$. We shall discuss the properties of such matrices in Chapter 4.

Returning to the general case we give another way to derive the optimality system (2.19). Recall that we have to find $\mathbf{y}$ and $\mathbf{u}$ satisfying (2.14,2.15):

$$\min_{\mathbf{y},\mathbf{u}} \frac{1}{2}\mathbf{y}^T Q_y \mathbf{y} - \mathbf{y}^T \mathbf{b} + \frac{\beta}{2}\mathbf{u}^T Q_u \mathbf{u}$$
$$\text{s.t. } K\mathbf{y} = \widehat{Q}\mathbf{u}.$$

Another way to do this is by introducing the Lagrangian function

$$\mathscr{L}(\mathbf{u}, \mathbf{y}, \mathbf{p}) := \frac{1}{2}\mathbf{y}^T Q_y \mathbf{y} - \mathbf{y}^T \mathbf{b} + \frac{\beta}{2}\mathbf{u}^T Q_u \mathbf{u} + \mathbf{p}^T (K\mathbf{y} - \widehat{Q}\mathbf{u}),$$

where here $\mathbf{p}$ denotes a vector of Lagrange multipliers. Then it is well known that a stationary point (and hence, by convexity, the minimum) is given by finding $\mathbf{u}$, $\mathbf{y}$ and $\mathbf{p}$ such that

$$\begin{aligned} \nabla_{\mathbf{u}} \mathscr{L}(\mathbf{u}, \mathbf{y}, \mathbf{p}) &= \beta Q_u \mathbf{u} - \widehat{Q}^T \mathbf{p} = 0 \\ \nabla_{\mathbf{y}} \mathscr{L}(\mathbf{u}, \mathbf{y}, \mathbf{p}) &= Q_y \mathbf{y} - \mathbf{b} + K^T \mathbf{p} = 0 \\ \nabla_{\mathbf{p}} \mathscr{L}(\mathbf{u}, \mathbf{y}, \mathbf{p}) &= K\mathbf{y} - \widehat{Q}\mathbf{u} = 0. \end{aligned}$$

This is just the optimality system (2.19) derived above. Therefore the adjoint variable we introduced earlier can be thought of as a Lagrange multiplier.

2.2.2 Optimize-then-discretize

Now we consider the other alternative – to find an optimality system for the continuous problem, and discretize these equations. To do this, we need to have a definition of the derivative of a functional.

Let $u_0 \in U$, an open set, and take $t > 0$ sufficiently small so that $u + th \in U$ for some h. We define the directional derivative of f at u_0 in the direction h as

$$\delta f(u_0, h) := \lim_{t \to 0} \frac{1}{t}(f(u_0 + th) - f(u_0)).$$

If the directional derivative exists for each $h \in U$, then the map

$$\begin{aligned} \delta f(u, \cdot) : U &\longrightarrow V \\ h &\longmapsto \delta f(u_0, h) \end{aligned}$$

is called the first variation of f at u. If this is a bounded linear operator, then this is the Gâteaux derivative of f at u_0.

In addition, f is Fréchet differentiable at u_0 if and only if there exists a bounded linear operator $D : U \longrightarrow V$ such that if $h \neq 0$,

$$\|h\|_U \to 0 \Rightarrow \frac{\|f(u_0 + h) - f(u_0) - Dh\|_V}{\|h\|_U} \to 0.$$

We say that f is Gâteaux (resp. Fréchet) differentiable on U if and only if f is Gâteaux (resp. Fréchet) differentiable for each $u_0 \in U$.

Let us consider the continuous problem (2.9–2.31):

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||_2^2 + \frac{\beta}{2}||u||_2^2$$

$$\begin{aligned} \text{s.t.} \qquad & -\nabla^2 y = u \text{ in } \Omega \\ & y = 0 \text{ on } \partial\Omega. \end{aligned}$$

We will find the optimality system using the Lagrangian technique, as described above for the discrete case. It has been shown (see, e.g. [83, 118]) that this is a rigorous way to derive the optimality system. A treatment from first principles can get very technical, so we just consider the Lagrangian technique here, and refer the interested reader to, for example, [79].

Introduce two Lagrange multipliers, p_1 and p_2, which we assume to have the appropriate smoothness, and formally consider the Lagrangian

$$\mathcal{L} = \frac{1}{2}\|y - \hat{y}\|_2^2 + \frac{\beta}{2}\|u\|_2^2 - \int_\Omega (-\nabla^2 y - u)p_1 \, dx - \int_{\partial\Omega} y p_2 \, ds. \tag{2.21}$$

Consider the Fréchet derivative with respect to y in the direction h:

$$\begin{aligned} D_y\mathcal{L}(y, u, p_1, p_2)h &= \int_\Omega (\bar{y} - \hat{y})h \, dx - \int_\Omega -\nabla^2 h \, p_1 \, dx - \int_{\partial\Omega} p_2 h \, ds \\ &= \int_\Omega (\bar{y} - \hat{y})h \, dx + \int_\Omega h \, \nabla^2 p_1 \, dx - \int_{\partial\Omega} \frac{\partial h}{\partial n} p_1 \, ds \\ &\quad + \int_{\partial\Omega} h \frac{\partial p_1}{\partial n} \, ds - \int_{\partial\Omega} p_2 h \, ds. \end{aligned}$$

There are no restrictions on the state, so for a minimum we must have that the optimal control and state, denoted by $\bar{u}$ and $\bar{y}$ respectively, must satisfy

$$D_y\mathcal{L}(\bar{y}, \bar{u}, p_1, p_2)h = 0 \quad \forall h \in \mathcal{H}^1(\Omega). \tag{2.22}$$

In particular, we must have $D_y\mathcal{L}(\bar{y}, \bar{u}, p_1, p_2)h = 0$ for all $h \in C_0^\infty(\Omega)$. In this case $h|_{\partial\Omega} = 0 = \frac{\partial h}{\partial n}|_{\partial\Omega}$, and so the expression above reduces to

$$\int_\Omega (\bar{y} - \hat{y} + \nabla^2 p_1)h \, dx \quad \forall\ h \in C_0^\infty(\Omega),$$

and so, applying the fundamental lemma of the Calculus of Variations, we get that

$$-\nabla^2 p_1 = y - \hat{y} \quad \text{in } \Omega.$$

Now consider $h \in H_0^1(\Omega)$, so that $h|_{\partial\Omega} = 0$. Then we get

$$\int_{\partial\Omega} \frac{\partial h}{\partial n} p_1 \, ds = 0 \quad \forall h \in H_0^1(\Omega)$$

so we have

$$p_1 = 0 \quad \text{on } \partial\Omega.$$

The remaining equations give us the link between p_1 and p_2, namely

$$p_2 = \frac{\partial p_1}{\partial n} \quad \text{on } \partial\Omega.$$

If we label $p_1 = p$, then we can write the adjoint equation as

$$-\nabla^2 p = y - \hat{y} \quad \text{in } \Omega \tag{2.23}$$

$$p = 0 \qquad \text{on } \partial\Omega, \tag{2.24}$$

which is the continuous analogue of the discrete adjoint equation obtained above.

Now consider optimality with respect to the control, u. The optimal control and state satisfy

$$D_u\mathcal{L}(\bar{y}, \bar{u}, p_1, p_2)h = 0 \quad \forall h \in L^2(\Omega).$$

This gives us that

$$\int_\Omega (\beta\bar{u} + p)h \, dx = 0 \quad \forall h \in L^2(\Omega),$$

and hence, almost everywhere,

$$\beta\bar{u} + p = 0. \tag{2.25}$$

Now consider the case where $\beta = 0$. Equation (2.25) gives us that $p = 0$ a.e., and substituting this into the adjoint equation (2.23) tells us that $\bar{y} = \hat{y}$ a.e. Therefore there exists an optimal control for the continuous problem if and only if $\hat{y}$ is a solution to the differential equation (2.10–2.11). In particular, if $\hat{y} \in L^2(\Omega)\backslash H_0^1(\Omega)$, then there does not exist an optimal control.

In order to get around this difficulty we pick a regularization parameter $\beta > 0$. In this case, we get the optimal control $\bar{u}$ and optimal state $\bar{y}$ satisfy the optimality system given by

$$\begin{array}{rcll} -\nabla^2 \bar{y} & = & \bar{u} & \text{in } \Omega \\ \bar{y} & = & 0 & \text{on } \partial\Omega \end{array} \quad \left| \quad \begin{array}{rcll} -\nabla^2 p & = & \bar{y} - \hat{y} & \text{in } \Omega \\ p & = & 0 & \text{on } \partial\Omega \end{array} \right. \quad \left| \quad \beta\bar{u} + p = 0 \quad \text{in } \Omega. \right.$$

To solve the optimality system numerically, we need to discretize these using, say, finite elements. Note that here we have three separate – yet coupled – equations.

The first two of these are just partial differential equations, so we can discretize them using the finite element method. Note that here it is possible to discretize the three equations independently of the other – e.g. using a different finite element space to discretize the state in the state equation as in the adjoint equation. However, we will consider only the case where we use the same discretization of the state, control and adjoint variables for all three equations.

Suppose, as in Section 2.1, there are spaces $\{\phi_i\}$, $\{\psi_i\}$ and $\{\chi_i\}$ such that

$$y = \sum_{i=1}^{n} Y_i \phi_i, \; u = \sum_{i=1}^{m} U_i \psi_i \text{ and } p = \sum_{i=1}^{k} P_i \chi_i.$$

First, let us consider the state equation, $-\nabla^2 y = u$, with homogeneous Dirichlet boundary conditions. As shown in Section 2.1, if we use a Galerkin finite element method we get

$$K_y \mathbf{y} = Q_{uy} \mathbf{u},$$

where $K_y = \{\int_\Omega \nabla\phi_i \cdot \nabla\phi_j\}_{i,j} \in \mathbb{R}^{m\times m}$ and $Q_{uy} = \{\int_\Omega \phi_i\psi_j\}_{i,j} \in \mathbb{R}^{n\times m}$.

Now consider the adjoint equation, $-\nabla^2 p = \hat{y} - \bar{y}$, again with homogeneous Dirichlet boundary conditions. Discretizing with a Galerkin finite element method here gives

$$K_p \mathbf{p} = -Q_{yp} \mathbf{y} + \mathbf{b},$$

where $K_p = \{\int_\Omega \nabla\chi_i \cdot \nabla\chi_j\}_{i,j} \in \mathbb{R}^{k\times k}$, $Q_{yp} = \{\int_\Omega \psi_i\chi_j\}_{i,j} \in \mathbb{R}^{m\times k}$ and $\mathbf{b} = \{\int_\Omega \hat{y}\chi_j\}_j$. Note that if we discretize u and p using the same elements then, in the notation of Section 2.2.1, $Q_{uy} = \widehat{Q}$ and $Q_{yp} = \widehat{Q}^T$.

We now turn our attention to the relation $\beta u + p = 0$. Using our approximations for u and p we get

$$\beta \sum_{i=1}^{m} U_i \psi_i - \sum_{i=1}^{k} P_i \chi_i = 0.$$

To get a weak formulation of this we clearly have a choice of trial space. If we pick $\text{span}\{\psi_i\}$, then we get the finite dimensional equation

$$\beta Q_u \mathbf{u} - Q_{pu}\mathbf{p} = \mathbf{0},$$

where $Q_u = \{\int_\Omega \psi_i\psi_j\}_{i,j} \in \mathbb{R}^{m\times m}$ and $Q_{pu} = \{\int_\Omega \chi_i\psi_j\}_{i,j} \in \mathbb{R}^{k\times m}$.

Putting these three equations together, we see that when we optimize first we get to an equation of the form

$$\begin{bmatrix} \beta Q_u & 0 & -Q_{pu} \\ 0 & Q_{yp} & K_p \\ -Q_{uy} & K_y & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \tag{2.26}$$

It is easy to see that if we choose the same finite element basis to discretize the variables y and p (so $k = n$ and $\chi_i = \phi_i$ for all $i = 1, \ldots, n$), then we get (2.19) – i.e. this is the same as when we did the discretization first. This will not be true in general – it holds here because the Laplacian is self-adjoint, so the discretization of the analytic adjoint and the discrete adjoint coincide. This will be the case whenever we employ an adjoint-consistent discretization. We consider a case where the two methods do not coincide in Chapter 6.

We briefly mention how error estimates are obtained for these problems. Let us explicitly introduce the solution map $S : L^2(\Omega) \to L^2(\Omega)$, $S(u) = y$ and the discrete solution map $S_h(u) = y_h$ $S_h : L^2(\Omega) \to L^2(\Omega)$, $S_h(u) = y_h$. Then from the FEM error estimate (2.8) we get $\|Su - S_h u\|_2 \leq Ch^2\|u\|_2$, which can be thought of as a bound on the operator norm

$$\|S - S_h\|_2 \leq Ch^2.$$

We can write the discretized problem (2.12) as

$$\min_u \frac{1}{2}\|S_h u - \hat{y}\|_2^2 + \frac{\beta}{2}\|u\|_2^2,$$

which, by the theory above, has a unique optimal control $\bar{u}_h$ and state $\bar{y}_h$ almost. Then, analogously to (2.16), the necessary optimality conditions for both the discrete control and the continuous control respectively are

$$S^*(S\bar{u} - \hat{y}) + \beta\bar{u} = 0$$
$$S_h^*(S_h\bar{u}_h - \hat{y}) + \beta\bar{u}_h = 0.$$

If we take inner products with $\bar{u} - \bar{u}_h$ and subtract, we get

$$\langle S^*(S\bar{u} - \hat{y}) - S_h^*(S_h\bar{u}_h - \hat{y}) + \beta(\bar{u} - \bar{u}_h), \bar{u} - \bar{u}_h\rangle = 0.$$

Rearranging, we get

$$\begin{aligned}\|S_h(\bar{u}-\bar{u}_h)\|_2^2+\beta\|\bar{u}-\bar{u}_h\|_2^2 &\leq |\,\langle\hat{y},(S-S_h)(\bar{u}-\bar{u}_h\rangle\,| \\ &\leq \|\hat{y}\|_2 Ch^2\|\bar{u}-\bar{u}_h\|_2.\end{aligned}$$

Finally, we have that the error satisfies

$$\|\bar{u}-\bar{u}_h\|_2 \leq C\frac{h^2}{\beta}\|\hat{y}\|_2, \tag{2.27}$$

where C is a constant independent of h and β. We see that the accuracy is dependent on the size of h, as we would expect, but also is inversely proportional to the regularization parameter β. In particular, if β is less than h^2 then the difference between the discrete and continuous optimal controls may be considerable.

2.2.3 Bound constraints

Most practical applications also require bound constraints to be satisfied on the control and/or the state, and we will briefly show how these can be handled. We only consider here the case where we have simple bound constraints, which we append to our standard problem:

$$\min_{y,u}\frac{1}{2}||y-\hat{y}||_2^2+\frac{\beta}{2}||u||_2^2 \tag{2.28}$$

$$\text{s.t.} \qquad -\nabla^2 y = u \text{ in } \Omega, \tag{2.29}$$

$$y = 0 \text{ on } \partial\Omega, \tag{2.30}$$

$$u_a \leq u \leq u_b \text{ a.e. in } \Omega, \tag{2.31}$$

$$y_a \leq y \leq y_b \text{ a.e. in } \Omega, \tag{2.32}$$

where u_a, u_b, y_a, y_b are constants. Consider first the case with just constraints on the control, i.e. $y_a = -\infty$ and $y_b = \infty$. We outline the theory for the optimize-then-discretize technique only[2].

Define the space of all admissible controls as

$$U_{ad} := \{u \in L^2(\Omega) \mid u_a \leq u \leq u_b\}.$$

We now want to find a critical point of the Lagrangian (2.21) subject to the constraint that $u \in U_{ad}$. Minimizing this with respect to the state will give, as before, the adjoint

[2]The discretize-then-optimize case is more straightforward – see, e.g. [97].

equation (2.23). Since we have box constraints on the control, u, the optimal control and state will, as in e.g. [68, Theorem 1.93], satisfy

$$D_u\mathcal{L}(\bar{y}, \bar{u}, p_1, p_2)(u - \bar{u}) \geq 0 \quad \forall u \in U_{ad}.$$

This gives us the variational inequality

$$\int_\Omega (\beta\bar{u} + p)(u - \bar{u})\, dx \geq 0 \quad \forall u \in U_{ad}. \tag{2.33}$$

This problem can be efficiently solved numerically using a primal-dual active set method, as introduced in this context by Bergounioux, Ito and Kunisch [10]. We first need to discretize the problem. As previously, let $\{\psi_i\}$ be some basis – generally, but not necessarily, different from the ϕ_i – such that $\psi_i \in L^2(\Omega)$. Then we can define $U^h_{ad} := \text{span}\{\psi_1, \ldots, \psi_m\} \cap U_{ad}$. Then if $u_h \in U^h_{ad}$, $u_h = \sum_{i=1}^m U_i\psi_i$. The function u_h is therefore uniquely determined by the vector of coefficients $\mathbf{u} = (U_1, \ldots, U_m)$. Let us further define $\mathcal{S}^h := \{\mathbf{u} \mid \sum_{i=1}^m U_i\psi_i \in U^h_{ad}\}$. Then the variational inequality (2.33) can be discretized as

$$(\beta Q_u\mathbf{u}^* - \widehat{Q}^T\mathbf{p}) \cdot (\mathbf{u} - \mathbf{u}^*) \geq 0 \quad \forall\ \mathbf{u} \in \mathcal{S}^h, \tag{2.34}$$

where we use the same notation as in the previous section. Suppose that we can consider the box constraints pointwise, so there exist vectors $\mathbf{u}^a$ and $\mathbf{u}^b$ such that

$$\mathcal{S}^h = \{\mathbf{u} \in \mathbb{R}^m : \mathbf{u}^a \leq \mathbf{u} \leq \mathbf{u}^b\}.$$

Note that if the control is discretized using piecewise constant elements then a pointwise description is reasonable, as everything will be local to the elements. If we use higher order elements, then we may be committing a variational crime [112, Chapter 4], and have to consider the resulting reduction in the accuracy of the solution. We will not consider these issues any further here, but refer the interested reader to, e.g. [117].

We can rewrite the discretized variational inequality (2.34) as

$$(\beta Q_u\mathbf{u}^* - \widehat{Q}^T\mathbf{p}) \cdot \mathbf{u}^* \leq (\beta Q_u\mathbf{u}^* - \widehat{Q}^T\mathbf{p}) \cdot \mathbf{u} \quad \forall\ \mathbf{u} \in \mathcal{S}^h,$$

and hence

$$\min_{\mathbf{u}\in\mathcal{S}^h}(\beta Q_u\mathbf{u}^* - \widehat{Q}^T\mathbf{p}) \cdot \mathbf{u} = \min_{\mathbf{u}\in\mathcal{S}^h} \sum_{i=1}^m (\beta Q_u\mathbf{u}^* - \widehat{Q}^T\mathbf{p})_i\mathbf{u}_i.$$

Since the components, $\mathbf{u}_i$ are independent, this means that

$$(\beta Q_u\mathbf{u}^* - \widehat{Q}^T\mathbf{p})_i\mathbf{u}^*_i = \min_{\mathbf{u}^a_i \leq \mathbf{u}_i \leq \mathbf{u}^b_i} (\beta Q_u\mathbf{u}^* - \widehat{Q}^T\mathbf{p})_i\mathbf{u}_i.$$

Hence, we must have that either $\beta Q_u \mathbf{u}^* - \widehat{Q}^T \mathbf{p} = 0$ (as before), or

$$\mathbf{u}_i^* = \begin{cases} \mathbf{u}_i^b & \text{if } (\beta Q_u \mathbf{u}^* - \widehat{Q}^T \mathbf{p})_i < 0 \\ \mathbf{u}_i^a & \text{if } (\beta Q_u \mathbf{u}^* - \widehat{Q}^T \mathbf{p})_i > 0 \end{cases} .$$

This characterization is not immediately helpful in practice, as it requires knowledge of the optimal control $\mathbf{u}^*$. It can be reformulated by defining

$$\boldsymbol{\mu} = \widehat{Q}^T \mathbf{p} - (\beta Q_u)\mathbf{u}.$$

Then, for $i = 1, \ldots, m$, since Q_u is positive definite and $\beta > 0$,

$$\mathbf{u}_i = \begin{cases} \mathbf{u}_i^a & \text{if } \mathbf{u}_i + \boldsymbol{\mu}_i < \mathbf{u}_i^a \\ \boldsymbol{\mu}_i & \text{if } \mathbf{u}_i + \boldsymbol{\mu}_i \in [\mathbf{u}_i^a, \mathbf{u}_i^b] \\ \mathbf{u}_i^b & \text{if } \mathbf{u}_i + \boldsymbol{\mu}_i > \mathbf{u}_i^b \end{cases} .$$

This formulation leads to an active set strategy. For some starting vectors $\mathbf{u}^{(0)}$ and $\boldsymbol{\mu}^{(0)}$, define at the n^{th} step

$$\begin{aligned} \mathcal{A}_n^a &= \{i \in \{1, \ldots, m\} : \mathbf{u}_i^{(n-1)} + \boldsymbol{\mu}_i^{(n-1)} < \mathbf{u}_i^a\} \\ \mathcal{A}_n^b &= \{i \in \{1, \ldots, m\} : \mathbf{u}_i^{(n-1)} + \boldsymbol{\mu}_i^{(n-1)} > \mathbf{u}_i^b\} \\ \mathcal{I}_n &= \{1, \ldots, m\} \backslash (\mathcal{A}_n^a \cup \mathcal{A}_n^b). \end{aligned}$$

Then an active set method based on the active (and inactive) sets defined above will converge in finitely many steps when applied to the discrete problem[118]. It can be shown [41, 111] that this strategy is equivalent to solving the system

$$\begin{bmatrix} \beta Q_u^{\mathcal{I}_n} & 0 & -(\widehat{Q}^{\mathcal{I}_n})^T \\ 0 & Q_y & K^T \\ -\widehat{Q}^{\mathcal{I}_n} & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{\mathcal{I}_n}^{(n)} \\ \mathbf{y}^{(n)} \\ \mathbf{p}^{(n)} \end{bmatrix} = \begin{bmatrix} \mathbf{a}^{(n)} \\ \mathbf{b} \\ \mathbf{d}^{(n)} \end{bmatrix}, \tag{2.35}$$

for appropriate (known) right hand side vectors $\mathbf{a}^{(n)}$ and $\mathbf{d}^{(n)}$ and where $Q_u^{\mathcal{I}} = Q_u(\mathcal{I}_n, \mathcal{I}_n)$ and $\widehat{Q}^{\mathcal{I}} = \widehat{Q}(:, \mathcal{I}_n)$. For more information, see Engel and Greibel [41] or Stoll and Wathen [111].

Note that this system, which needs to be solved at each step, is also a saddle point matrix. The efficient solution of such a system is one of the main bottlenecks in current algorithms for problems of this type [57, 41].

The issue of constraints on the state is much more difficult. Consider a case which has only bound constraints on the state – i.e. $u_a = -\infty$ and $u_b = \infty$ in (2.31). Then the Lagrange multipliers[3] which are associated with the state constraint will now be

[3]We implicitly assumed that such Lagrange multipliers exist and are well-behaved functions in the analysis of the control contrained case above.

measures. In can be shown – e.g. [68, Section 1.7.3.5] that the optimality conditions for the continuous problem in this case are given by

$-\nabla^2 \bar{y} = \bar{u}$ in Ω $\bar{y} = 0$ on $\partial\Omega$	$-\nabla^2 p = \bar{y} - \hat{y} + \mu_a - \mu_b$ in Ω $p = 0$ on $\partial\Omega$
$\int_{\bar{\Omega}} (\bar{y} - y_b)\, \mathrm{d}\mu_\mathrm{b} = \int_{\bar{\Omega}} (\bar{y} - y_a)\, \mathrm{d}\mu_\mathrm{a} = 0$	$\beta \bar{u} + p = 0$ in Ω,

where μ_a and μ_b – the Lagrange multipliers for the state constraints – are nonnegative Borel measures. See [118] for examples illustrating this.

The presence of measures in the optimality conditions presents significant difficulties, and this is the topic of much recent research. We mention here two methods of getting around this problem currently available in the literature. The first involves Laurentiev-type regularization, where the state constraints are turned into more manageable control constraints – see, e.g. Meyer, Rösch and Tröltzsch [86]. The second method – Moreau-Yosida relaxation – works by dropping the state constraint and adding a penalty term to the cost functional – see, e.g. Hintermüller and Kunisch [66, 67].

Although the analysis is more involved in the case of bound constraints of both type, from a linear algebra view the matrices we need to solve are essentially the same as the matrix (2.26), and algorithms for the solution of that system will also be applicable here. Hence we will only consider the problem without bound constraints in the remainder of this thesis.

Proving error estimates for bound constrained problems is the topic of much current research – see Trölzsch [117] and the references therein for an overview of the state-of-the-art.

2.2.4 Different boundary conditions

The above analysis was done only for the case of a state equation with homogeneous Dirichlet boundary conditions. Below we will consider some other boundary conditions. We only consider the equations without bound constraints here; if such constraints are necessary, we can use the obvious extensions of the methods described above.

First, one can extend the results to inhomogeneous boundary conditions. The extension is trivial using the Lagrange multiplier technique, but a bit more technical to derive from first principles. In this case, if the discrete state equation is

$$K\mathbf{u} = \widehat{Q}\mathbf{y} + \mathbf{d},$$

then the optimality system analogous to (2.19) is

$$\begin{bmatrix} \beta Q_u & 0 & -\widehat{Q}^T \\ 0 & Q_y & K \\ -\widehat{Q} & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}, \tag{2.36}$$

where the only change is the addition of the vector containing the boundary data on the right hand side. This is the same system whether we optimize-then-discretize or discretize-then-optimize (providing we discretize y and p the same way).

We can also replace the Dirichlet boundary condition with a Neumann boundary condition, giving the problem

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||_2^2 + \frac{\beta}{2}||u||_2^2$$

$$\begin{aligned} \text{s.t.} \qquad -\nabla^2 y &= u \text{ in } \Omega \\ \frac{\partial y}{\partial n} &= g \text{ on } \partial\Omega. \end{aligned}$$

This, if we optimize-then-discretize using the method described above, will give the optimality system

$$\begin{array}{rcll} -\nabla^2 \bar{y} & = & \bar{u} & \text{in } \Omega \\ \frac{\partial \bar{y}}{\partial n} & = & g & \text{on } \partial\Omega \end{array} \quad \left| \quad \begin{array}{rcll} -\nabla^2 p & = & \hat{y} - \bar{y} & \text{in } \Omega \\ \frac{\partial p}{\partial n} & = & 0 & \text{on } \partial\Omega \end{array} \right. \quad \beta\bar{u} + p = 0 \quad \text{in } \Omega. \tag{2.37}$$

When discretized, this system will also be of the form (2.36), where K now denotes a discrete Neumann Laplacian matrix, and the 'mass' matrices have been enlarged to include the boundary. It is interesting to note that here K is singular – there is a one dimension kernel corresponding to the constant vectors – but the optimal control problem still has a unique solution, since the saddle point system is invertible (see Chapter 4 for conditions for invertibility of a saddle point system). It is trivial to see that the discretize-then-optimize technique gives the same system here.

The results above can be extended in the obvious way to give the optimality system for a problem where we have different boundary conditions on different parts of the boundary, or Robin boundary conditions.

2.3 Boundary control

We now consider a problem where the control only acts on the boundary, not distributed throughout the domain, so that the control $u \in L^2(\partial\Omega)$. First of all, consider a Neumann boundary control problem

$$\min_{y,u} \frac{1}{2} \int_\Omega (y - \hat{y})^2 \, \mathrm{dx} + \frac{\beta}{2} \int_{\partial\Omega} u^2 \, \mathrm{ds}$$

$$\text{s.t.} \qquad -\nabla^2 y = g \text{ in } \Omega$$
$$\frac{\partial y}{\partial n} = u \text{ on } \partial\Omega.$$

Firstly, let us consider the discretize-then-optimize technique. The weak formulation of the PDE constraint is: find $y \in \mathcal{H}^1(\Omega)$ such that

$$\int_\Omega \nabla y \cdot \nabla v \,\mathrm{dx} - \int_{\partial\Omega} u\gamma(v)\,\mathrm{ds} = \int_\Omega gv\,\mathrm{dx} \qquad \forall v \in \mathcal{H}^1(\Omega),$$

where γ is the trace operator, which restricts a function to the boundary $\partial\Omega$.

Let $\{\phi_i\}_{1\le i\le m}$ be a set of finite element basis functions defined on some triangulation of Ω. Suppose this mesh has n nodes on $\partial\Omega$, and define another finite element space with basis $\{\psi_i\}_{1\le i\le n}$ at these nodes. Then, as in the Dirichlet control case, we can define approximations to the state and control by

$$y_h = \textstyle\sum_{i=1}^m Y_i\phi_i, \quad u_h = \sum_{i=1}^n U_i\psi_i.$$

The discretized constraint can therefore be written in matrix form as

$$K\mathbf{y} = \widehat{Q}\mathbf{u} + \mathbf{g},$$

where

$$K = [k_{i,j}] \in \mathbb{R}^{m\times m},\ k_{i,j} = \int \nabla\phi_i \cdot \nabla\phi_j$$
$$\widehat{Q} = [\widehat{q}_{i,j}] \in \mathbb{R}^{m\times n},\ \widehat{q}_{i,j} = \int \phi_i\psi_j$$
$$\mathbf{g} = [g_i] \in \mathbb{R}^n,\ g_i = \int g\phi_i.$$

Now, by introducing a vector of Lagrange multipliers $\mathbf{p}$ and following the same arguments as in the previous section it is easy to see that the discrete optimality system is

$$\begin{bmatrix} \beta Q_u & 0 & -\widehat{Q}^T \\ 0 & Q_y & K \\ -\widehat{Q} & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{g} \end{bmatrix}, \tag{2.38}$$

where

$$Q_u = \left[q^u_{i,j}\right] \in \mathbb{R}^{n\times n},\ q^u_{i,j} = \int \phi_i\phi_j$$
$$Q_y = \left[q^y_{i,j}\right] \in \mathbb{R}^{m\times m},\ q^y_{i,j} = \int \psi_i\psi_j$$
$$\mathbf{b} = [b_i] \in \mathbb{R}^n,\ b_i = \int \hat{y}\phi_i.$$

Now consider the optimize-then-discretize approach. Define the Lagrangian

$$\mathcal{L} = \frac{1}{2}\int_\Omega (y-\hat{y})^2 \,\mathrm{dx} + \frac{\beta}{2}\int_{\partial\Omega} u^2 \,\mathrm{ds} + \int_\Omega (-\nabla^2 y - g)p_1 \,\mathrm{dx} + \int_{\partial\Omega} (\frac{\partial y}{\partial n} - u)p_2 \,\mathrm{ds}.$$

As in the previous section, consider the Fréchet derivative with respect to y in the direction h:

$$\begin{aligned} D_y\mathcal{L}(y,u,p_1,p_2)h &= \int_\Omega (\bar{y}-\hat{y})h \,\mathrm{dx} - \int_\Omega h\,\nabla^2 p_1 \,\mathrm{dx} + \int_{\partial\Omega} \frac{\partial h}{\partial n} p_1 \,\mathrm{ds} \\ &\quad - \int_{\partial\Omega} h\frac{\partial p_1}{\partial n} \,\mathrm{ds} + \int_{\partial\Omega} \frac{\partial h}{\partial n} p_2 \,\mathrm{ds}. \end{aligned}$$

For a minimum we must have that the optimal control and state, dentoted by $\bar{u}$ and $\bar{y}$ respectively, must satisfy

$$D_y\mathcal{L}(\bar{y},\bar{u},p_1,p_2)h = 0 \quad \forall h \in \mathcal{H}^1(\Omega). \tag{2.39}$$

In particular, we must have $D_y\mathcal{L}(\bar{y},\bar{u},p_1,p_2)h = 0$ when $h|_{\partial\Omega} = 0 = \frac{\partial h}{\partial n}|_{\partial\Omega}$, and so applying the fundamental lemma of the Calculus of Variations we get that

$$-\nabla^2 p_1 = \hat{y} - \bar{y} \quad \text{in } \Omega.$$

Now consider h so that $\frac{\partial h}{\partial n}|_{\partial\Omega} = 0$. Then we get

$$\int_{\partial\Omega} h\frac{\partial p_1}{\partial n} \,\mathrm{ds} = 0 \quad \forall h \in \mathcal{H}_0^1(\Omega)$$

so we have

$$\frac{\partial p_1}{\partial n} = 0 \quad \text{on } \partial\Omega.$$

The remaining equations give us the link between p_1 and p_2, namely

$$p_1 + p_2 = 0 \quad \text{on } \partial\Omega.$$

If we label $p_1 = p$, then we can write the adjoint equation as

$$-\nabla^2 p = \hat{y} - \bar{y} \quad \text{in } \Omega \tag{2.40}$$

$$\frac{\partial p}{\partial n} = 0 \qquad \text{on } \partial\Omega, \tag{2.41}$$

which is again the continuous analogue of the discrete adjoint equation obtained above.

Since there are no box constraints on the control, u, the optimal control and state satisfy

$$D_u\mathcal{L}(\bar{y},\bar{u},p_1,p_2)h = 0 \quad \forall\ h \in L^2(\partial\Omega).$$

This gives us that

$$\int_\Omega (\beta\bar{u} - p)h \,\mathrm{dx} = 0 \quad \forall\ h \in H^1(\Omega) \tag{2.42}$$

and hence,

$$\beta\bar{u} - p = 0. \tag{2.43}$$

On discretization, the usual choice of bases will again give us the discrete optimality system

$$\begin{bmatrix} \beta Q_u & 0 & -\widehat{Q}^T \\ 0 & Q_y & K \\ -\widehat{Q} & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{g} \end{bmatrix},$$

which was the same as was obtained by the discretize-then-optimize method.

Now consider the Dirichlet boundary control problem

$$\min_{y,u} \frac{1}{2}\int_\Omega (y - \hat{y})^2 \,\mathrm{dx} + \frac{\beta}{2}\int_{\partial\Omega} u^2 \,\mathrm{ds}$$

$$\begin{aligned} \text{s.t.} \qquad -\nabla^2 y &= g \text{ in } \Omega \\ y &= u \text{ on } \partial\Omega. \end{aligned}$$

Getting first order optimality conditions for this problem is somewhat harder, since it does not have a variational formulation [124]. However, by following the optimize-then-discretize method we can see what the discrete optimality system must look like. A more rigorous approach, which leads to the same optimality system, can be found in Neittaanmaki *et al.* [88, Example 3.2.3] or Lions [79, Chapter II.5].

Again, consider the Lagrangian

$$\mathcal{L} = \frac{1}{2}\int_\Omega (y - \hat{y})^2 \,\mathrm{dx} + \frac{\beta}{2}\int_{\partial\Omega} u^2 \,\mathrm{ds} + \int_\Omega (-\nabla^2 y - g)p_1 \,\mathrm{dx} + \int_{\partial\Omega} (u - y)p_2 \,\mathrm{ds}.$$

The Fréchet derivative with respect to y in the direction h is given by

$$\begin{aligned} D_y\mathcal{L}(y, u, p_1, p_2)h = &\int_\Omega (\bar{y} - \hat{y})h \,dx + \int_\Omega h\,\nabla^2 p_1 \,dx - \int_{\partial\Omega} \frac{\partial h}{\partial n} p_1 \,ds \\ &+ \int_{\partial\Omega} h\frac{\partial p_1}{\partial n} \,ds - \int_{\partial\Omega} p_2 h \,ds, \end{aligned}$$

and so an argument as above tells us that p_1 and p_2 satisfy

$$\begin{aligned} -\nabla^2 p_1 &= y - \hat{y} \quad \text{in } \Omega \\ p_1 &= 0 \quad \text{on } \partial\Omega \\ p_2 &= \frac{\partial p_1}{\partial n} \quad \text{on } \partial\Omega. \end{aligned}$$

The Fréchet derivative with respect to u in the direction h is given by

$$D_u\mathcal{L}(y,u,p_1,p_2)h = \beta\int_{\partial\Omega}\bar{u}h\,ds - \int_{\partial\Omega}p_2h\,ds,$$

and so, since this must vanish for all $h \in L^2(\partial\Omega)$, we have

$$\beta u = p_2 \quad \text{on } \partial\Omega.$$

Therefore, if we eliminate p_2 and relabel p_1 as p, the optimality system here is given by

$$\begin{array}{rcll} -\nabla^2\bar{y} & = & g & \text{in } \Omega \\ y & = & \bar{u} & \text{on } \partial\Omega \end{array} \;\Bigg|\; \begin{array}{rcll} -\nabla^2 p & = & \hat{y}-\bar{y} & \text{in } \Omega \\ p & = & 0 & \text{on } \partial\Omega \end{array} \;\Bigg|\; \beta\bar{u} - \tfrac{\partial p}{\partial n} = 0 \quad \text{on } \partial\Omega. \tag{2.44}$$

We include this example for completeness, and will only consider the Neumann boundary control problem in the remainder of this thesis.

Chapter 3

The Numerical Solution of Linear Systems of Equations

The discretization techniques presented in Chapter 2 find an approximation to the optimal control for a partial differential equation by solving a linear system of equations, which is in general of very large dimension. The main goal of this thesis is to develop efficient algorithms to solve systems such as (2.19). When looking at algorithms for solving linear systems of equations there are two main philosophies – one can either solve the system directly, or by some iterative procedure.

Direct methods are based on factorizing the system into a product of matrices, each posessing a structure which is easier to solve. Most direct methods are, at the simplest level, based on some kind of *LU-factorization*. MATLAB's `backslash` command is a 'black box' direct solver which chooses a different algorithm depending on the properties of the matrix one would like to solve. As we will see from the results in later chapters, `backslash` performs extremely well for small- to medium-sized two dimensional problems, but for large problems, and especially in three dimensions, it starts to struggle.

The usual Gaussian elimination algorithm for a dense $n \times n$ matrix takes $\frac{2}{3}n^3 + \mathcal{O}(n^2)$ flops [115, Lecture 20]. For a sparse matrix there are sparse direct algorithms that can reduce this work. See, for example, Duff, Erisman and Reid [35] for more details.

Iterative methods work by giving successively better approximations to the solution of the system. In exact arithmetic, provided the matrix is non-singular, direct methods always give you the exact solution, whereas iterative methods will generally never quite give you this. However, in floating point arithmetic the situation is less clear. The forward error in the solution with a direct method not only depends on the accuracy of the machine, $\epsilon_{\text{machine}}$, but also the condition number of the matrix –

see, for example Higham [65, Chapter 9]. Iterative methods, however, do not suffer from such a limitation – although they take more iterations to converge for an ill-conditioned problem, you can get accuracy of the order of $\epsilon_{\text{machine}}$; this is the principle behind iterative refinement methods [65, Chapter 12].

Even for well conditioned systems, if you know that your discretization is such that the solution of your linear system and the solution of the infinite-dimensional problem that you are interested in only agree to, say, three decimal places, then knowing the solution of the linear system to fifteen decimal places is considerable overkill. With iterative methods you can stop once you have reached the desired accuracy, which has the promise to save significant computing time.

Also, if n is large the work to solve a linear system by a direct method may be prohibitive. By exploiting the structure of the problem it is often possible to find an *optimal* iterative method – i.e. an iterative method that takes $\mathcal{O}(1)$ steps and $\mathcal{O}(n)$ work per step to solve an $n \times n$ system. This means that, in practice, the iterative method takes a fixed number of iterations, independent of the problem size, to converge to a fixed tolerance and the time to solve the system will scale linearly with the problem size. Figure 3.1 shows a schematic diagram which illustrates this difference. Of course, the constants involved in the work estimate mean it is usually the case that a direct method is faster for smaller matrices, but as n gets larger an optimal iterative method will be more efficient.

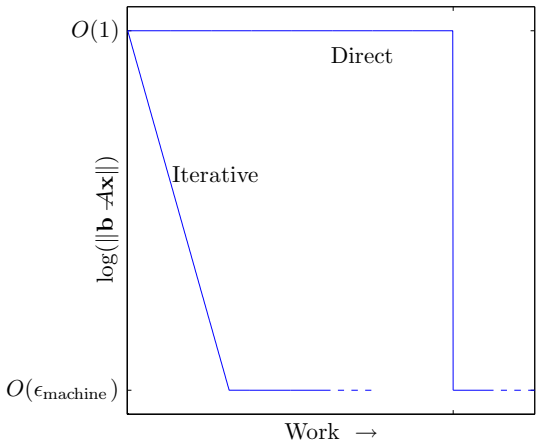


Figure 3.1: Schematic illustration of convergence of a direct and an optimal iterative method. (*after Trefethen and Bau [115, Figure 32.1]*)

The remainder of this thesis is dedicated to developing optimal iterative methods for systems of the type we introduced in Chapter 2. In the sections below we discuss some iterative methods for solving general linear systems of equations.

3.1 General Iterative Methods

3.1.1 Simple Iteration

Suppose we want to solve the equation

$$A\mathbf{x} = \mathbf{b} \tag{3.1}$$

by an iterative method. One of the simplest ways is to use a splitting $A = M - N$ to give you the simple iteration: given $\mathbf{x}^{(0)}$

$$M\mathbf{x}^{(k)} = N\mathbf{x}^{(k-1)} + \mathbf{b}. \tag{3.2}$$

Here the choice of matrix, M, is free, and depending on the splitting used we will get different iterative methods. We have to solve with M at each iteration, so it should be a matrix such that this is inexpensive. Many of the obvious choices of M have names, for example, $M = \text{diag}(A)$ would be Jacobi iteration, M being the lower triangular part of A would be Gauss-Seidel, etc. It is clear that if $\mathbf{x}^{(m)} = \mathbf{x}$ for some m, then $\mathbf{x}^{(m+1)} = \mathbf{x}$ – i.e. if we have found the exact solution, doing further iterations will not take us away from it. This leads us to ask what conditions do we require of M so that we can be sure the iteration converges?

Note that we can write the iteration (3.2) as

$$\mathbf{x}^{(k)} = M^{-1}N\mathbf{x}^{(k-1)} + M^{-1}\mathbf{b},$$

or, alternatively,

$$\mathbf{x}^{(k)} = (I - M^{-1}A)\mathbf{x}^{(k-1)} + M^{-1}\mathbf{b}, \tag{3.3}$$

since $N = M - A$. We call the matrix $I - M^{-1}A$ the *iteration matrix.* Recall that the exact solution, $\mathbf{x}$, also satisfies

$$\mathbf{x} = (I - M^{-1}A)\mathbf{x} + M^{-1}\mathbf{b}. \tag{3.4}$$

If we subtract (3.3) from (3.4) we get

$$\mathbf{e}^{(k)} = (I - M^{-1}A)\mathbf{e}^{(k-1)} = \cdots = (I - M^{-1}A)^k\mathbf{e}^{(0)}, \tag{3.5}$$

where $\mathbf{e}^{(k)} := \mathbf{x} - \mathbf{x}^{(k)}$ denotes the error at the k^{th} iteration. Let $\|\cdot\|$ denote any vector norm, and we define the matrix norm induced by this by

$$\|A\| = \sup_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|.$$

We have that the error at the k^{th} iteration satisfies

$$\|\mathbf{e}^{(k)}\| \leq \|(I - M^{-1}A)^k\|\|\mathbf{e}^{(0)}\|.$$

It can be shown [53, Lemma 2.1.1] that the norm of the error in (3.2) will approach zero for every initial error $\mathbf{e}^{(0)}$ if and only if

$$\lim_{k\to\infty} \|(I - M^{-1}A)^k\| = 0.$$

This gives us a necessary and sufficient condition for convergence, but it is not very practical. First, in general it is hard to get information about the norm of a matrix such as the iteration matrix. Second, it is norm-dependent – ideally we'd like a condition that gives us convergence in any norm.

The spectral radius of a matrix C, denoted $\rho(C)$, is defined as

$$\rho(C) = \max\{|\lambda| \ : \ \lambda \text{ is an eigenvalue of } C\}.$$

The spectral radius of a matrix is related to the norm of powers of a matrix by Gelfand's formula [53, Corollary 1.3.1],

$$\rho(C) = \lim_{k\to\infty} \|C^k\|^{1/k},$$

from which we can see that the simple iteration will converge in any norm, for every starting vector $\mathbf{x}^{(0)}$, if and only if $\rho(I - M^{-1}A) < 1$.

Although we now have a condition which tells us that our splitting will be convergent, it does not give us any information regarding how long we'll have to wait for a reasonable approximation to our solution. Let us first consider normal matrices[1], that is, matrices C where $CC^T = C^TC$, or, equivalently, matrices whose eigenvectors form an orthonormal basis. In this case, if we look at the 2-norm, it is well known (see, e.g. [53, p. 28]) that

$$\|C\|_2 = \rho(C).$$

Therefore if the iteration matrix $I - M^{-1}A$ is normal the necessary and sufficient condition for convergence, $\rho(I - M^{-1}A) < 1$, also tells us that convergence (in the 2-norm) would be monotonic, as for every k,

$$\|\mathbf{e}^{(k)}\|_2 \leq \rho(I - M^{-1}A)\|\mathbf{e}^{(k-1)}\|_2.$$

[1]For simplicity we only consider the case where C has real entries, since this is the situation with the matrices coming from optimization problems of the type we are considering; the complex case can be treated analogously, with the transpose replaced by the conjugate transpose.

We can generalize the concept of normality slightly. Given a matrix C and a positive-definite matrix $\mathcal{H}$, we can define an inner product $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}} := \mathbf{x}^T \mathcal{H} \mathbf{y}$. Then

$$\langle C\mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}} = \mathbf{x}^T C^T \mathcal{H} \mathbf{y} = \mathbf{x}^T \mathcal{H} (\mathcal{H}^{-1} C^T \mathcal{H}) \mathbf{y} = \left\langle \mathbf{x}, C^\dagger \mathbf{y} \right\rangle_{\mathcal{H}},$$

where $C^\dagger := \mathcal{H}^{-1} C^T \mathcal{H}$, the $\mathcal{H}-$adjoint. Then we say that C is $\mathcal{H}-$normal if and only if

$$C^\dagger C = C C^\dagger.$$

Note that C is self adjoint in the $\mathcal{H}-$inner product if and only if $\mathcal{H}C = C^T \mathcal{H}$, and in this case C is clearly $\mathcal{H}-$normal.

We can now apply these ideas to the simple iteration. If our iteration matrix is $\mathcal{H}-$normal, then using a standard result from the theory of linear operators (see, e.g. [113, Theorem 6.2-E]) we have that

$$\|I - M^{-1}A\|_{\mathcal{H}} = \rho(I - M^{-1}A),$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the matrix norm induced by the vector norm $\|\cdot\|_{\mathcal{H}} = \langle \cdot, \cdot \rangle_{\mathcal{H}}^{1/2}$. Hence $\mathcal{H}-$normality ensures monotonic convergence in the $\mathcal{H}$-norm.

If the iteration matrix is nonnormal, however, it is often true that $\|I - M^{-1}A\| > 1$, even if $\rho(I - M^{-1}A) < 1$. In this case, although the method is guaranteed to converge eventually, there may be a significant transient period where the error may increase or stagnate. One way to understand this effect is using the concept of pseudospectra to quantify nonnormality, as described by Trefethen and Embree in [116].

Let $C \in \mathbb{R}^{n \times n}$ and $\epsilon > 0$. Then the ϵ-pseudospectrum, $\sigma_\epsilon(C)$, is defined by

$$\sigma_\epsilon(C) := \left\{ z \in \mathbb{C} \ : \ \|(z - C)^{-1}\| > 1/\epsilon \right\}.$$

We further define the $\epsilon-$pseudospectral radius of C by

$$\rho_\epsilon(C) := \sup\{|z| \ : \ z \in \sigma_\epsilon(C)\}.$$

The matrix $(z - C)^{-1}$ is called the resolvent of C, and, following the convention used in [116], we assume $\|(z - C)^{-1}\| = \infty$ for $z \in \sigma(C)$. Note that for a normal matrix the ϵ-pseudospectrum is just the union of open ϵ-balls around the points of the spectrum.

We know that, if we plot convergence on a log-scale, the initial slope is simply going to be $\log(\|I - M^{-1}A\|)$, and also that the asymptotic convergence rate will be $\rho(I - M^{-1}A)$. We would like to know what happens between these two limits. To

do this, we want to have bounds (both upper and lower) on $\|C^k\|$. One such bound [116, Theorem 16.4] is that

$$\sup_{k\geq 0} \|C^k\| \geq \frac{\rho_\epsilon(C) - 1}{\epsilon} \qquad \forall\ \epsilon > 0.$$

If we define the Kreiss constant of C with respect to the unit disk as

$$\mathcal{K}(C) = \sup_{\epsilon>0} \frac{\rho_\epsilon(C) - 1}{\epsilon} = \sup_{|z|>1} (|z| - 1)\|(z - C)^{-1}\|,$$

then we can write

$$\sup_{k\geq 0} \|C^k\| \geq \mathcal{K}(C).$$

A converse of this is that [116, Theorem 16.2]

$$\|C^k\| \leq \frac{\rho_\epsilon(C)^{k+1}}{\epsilon} \qquad \forall \epsilon > 0,\ \forall k \geq 0.$$

This can be weakened to

$$\|C^k\| \leq \mathrm{e}N\mathcal{K}(C),$$

where C is a matrix of dimension N.

Note that the upper bound tells us that if the pseudospectra significantly protrude outside the unit disk, i.e. $\rho_\epsilon(C) > 1 + \epsilon$ for some ϵ, then there must be some transient growth.

There are codes written for analyzing pseudospectra of particular matrices, for example EigTool [129] by Thomas Wright, but it is in general hard to get quantifiable numerical bounds of this type for, say, a given class of matrices. Nevertheless, looking at the pseudospectra does give a good insight into what is happening with regards to convergence of a nonnormal matrix.

3.1.2 Richardson Iteration

Suppose we want to speed up convergence of a simple iteration, as defined in the preceding section. Note that we can write (3.3) in the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + M^{-1}\mathbf{r}^{(k)},$$

where $\mathbf{r}^{(k)}$ denotes the residual, $\mathbf{b} - A\mathbf{x}^{(k)}$. One way to adapt this method would be to introduce a parameter, α, and consider the iteration

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha M^{-1}\mathbf{r}^{(k)}.$$

Note that this is itself a simple iteration, with splitting matrix $\widehat{M} = \frac{1}{\alpha}M$. Let us first consider the case where $M = I$, so we get

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{r}^{(k)}. \tag{3.6}$$

This is called the *Richardson iteration*. The error will satisfy

$$\mathbf{e}^{(k+1)} = \mathbf{e}^{(k)} - \alpha A \mathbf{e}^{(k)},$$

where we have used the fact that $\mathbf{r}^{(k)} = A\mathbf{e}^{(k)}$. We can pick an optimal value of α, which will depend on the quantity we would like minimized. For example, if we want to minimize the error in the 2-norm, we have that

$$\|\mathbf{e}^{(k+1)}\|_2 \leq \|I - \alpha A\|_2 \|\mathbf{e}^{(k)}\|_2.$$

Now suppose that A is positive definite. Then $\|I - \alpha A\|_2 = \max_i |\lambda_i(I - \alpha A)|$, so the optimal value of α is the value for which the smallest and the largest eigenvalues of $I - \alpha A$ are the same in absolute value, i.e. α satisfies

$$\begin{aligned} \alpha\lambda_{\max} - 1 &= 1 - \alpha\lambda_{\min}, \\ \text{so} \quad \alpha &= 2/(\lambda_{\min} + \lambda_{\max}), \end{aligned} \tag{3.7}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of A respectively. For this optimal value of α,

$$\|I - \alpha A\|_2 = 1 - \frac{2}{\lambda_{\min} + \lambda_{\min}}\lambda_{\min} = \frac{\kappa - 1}{\kappa + 1},$$

where $\kappa := \lambda_{\max}/\lambda_{\min}$ is the condition number of A. Hence we have shown that the Richardson iteration with a parameter chosen to minimize the 2-norm of the error satisfies

$$\|\mathbf{e}^{(k+1)}\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right) \|\mathbf{e}^{(k)}\|_2.$$

We therefore have that

$$\|\mathbf{e}^{(k)}\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\mathbf{e}^{(0)}\|_2. \tag{3.8}$$

However, we highlight that this bound is when we use the optimal value of α, which requires you to know some information about the eigenvalues of A. Calculating these will be expensive in itself for a general matrix.

So far we have only considered a simple Richardson iteration - i.e., the case where $M = I$. Of course, the system

$$A\mathbf{x} = \mathbf{b}$$

is equivalent to the system

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b},$$

and if we choose M such that $M^{-1}A$ has a smaller condition number than the original matrix, A, then we should expect convergence to be faster. The bound above, however, relies on the matrix being positive definite. If our matrix M is positive definite, then we can write its Cholesky decomposition, $M = HH^T$, and thus formally solve the equivalent system

$$\widehat{A}\widehat{\mathbf{x}} = \widehat{\mathbf{b}},$$

where $\widehat{A} = H^{-1}AH^{-T}$, $\widehat{\mathbf{x}} = H^T\mathbf{x}$ and $\widehat{\mathbf{b}} = H^{-1}\mathbf{b}$. This guarantees that the matrix $\widehat{A}$ is positive definite. We do this formally, as in practice we need not know the Cholesky decomposition of A, and all that is needed is the matrix M. The algorithm is given in Algorithm 1. Because M is often chosen to reduce the condition number of the system, we call M a *preconditioner*.

Algorithm 1 Preconditioned Richardson iteration to solve $A\mathbf{x} = \mathbf{b}$ with preconditioner M

Choose $\mathbf{x}^{(0)}$
$\alpha = (\lambda_{\min} + \lambda_{\max})/2$
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
Solve $M\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$
for $k = 0, 1, \ldots$ **until** convergence **do**
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{z}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha A\mathbf{z}^{(k)}$
  Solve $M\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$
end for

Consider the special case where $M = D = \mathrm{diag}(A)$. Recall that if we take $\alpha = 1$ in the Richardson iteration this is called the Jacobi iteration. By taking a different value of the parameter α we get the *relaxed Jacobi* iteration. The optimal value of α can be chosen as above, provided we have knowledge of the maximum and minimum eigenvalues of $D^{-1}A$. It is easily seen that this scheme is equivalent (in exact arithmetic) to taking a weighted average of two successive Jacobi iterations, i.e.

$$\begin{aligned} D\widehat{\mathbf{x}}^{(k+1)} &= -(L+U)\mathbf{x}^{(k)} + \mathbf{b} \\ \mathbf{x}^{(k+1)} &= \alpha\widehat{\mathbf{x}}^{(k+1)} + (1-\alpha)\mathbf{x}^{(k)}, \end{aligned}$$

where L and U are matrices formed from the strictly lower and upper triangular parts of A respectively.

Similar ideas can be applied to the Gauss-Seidel iteration, where $M = L + D$ and $N = U$. Here, we generally relax by making the iteration

$$(D + \alpha L)\mathbf{x}^{(k+1)} = -(\alpha U - (1 - \alpha)D)\mathbf{x}^{(k)} + \alpha\mathbf{b},$$

which we call the *Successive Over Relaxation (SOR)* method. Note that the iteration matrix here is not symmetric positive definite, so the error bounds derived above will not hold. We can adapt the method to preserve symmetry by doing the following variation of the above iteration:

$$\begin{aligned}(D + \alpha L)\mathbf{x}^{(k+\frac{1}{2})} &= -(\alpha U + (1 - \alpha)D)\mathbf{x}^{(k)} + \alpha\mathbf{b} \\ (D + \alpha U)\mathbf{x}^{(k+1)} &= -(\alpha L + (1 - \alpha)D)\mathbf{x}^{(k+\frac{1}{2})} + \alpha\mathbf{b}.\end{aligned}$$

This is called the *Symmetric Successive Over Relaxation (SSOR)* method.

For more information about these methods, see, for example, Varga [123] or Meurant [84, Sections 5.5 & 5.6].

3.1.3 Steepest Descent

In the previous section we picked a fixed parameter α, the value of which was dependent on the eigenvalues of the iteration matrix. If we were to allow the parameter α to change at each step we could possibly improve upon this method. Again, first consider the case where $M = I$, then the iteration will become the generalized Richarson iteration,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{r}^{(k)}. \tag{3.9}$$

We would like to find some way to choose the parameter α_k. Suppose that A is symmetric positive definite. Then a quantity that we might want to minimize is the error in the A^m-norm at each step, i.e. $\|\mathbf{e}^{(k)}\|_{A^m} = \left\langle \mathbf{e}^{(k)}, A^m\mathbf{e}^{(k)} \right\rangle^{1/2}$, where m is an integer to be determined. From (3.9) we get

$$\mathbf{e}^{(k+1)} = \mathbf{e}^{(k)} - \alpha_k\mathbf{r}^{(k)}. \tag{3.10}$$

Now,

$$\left\langle \mathbf{e}^{(k+1)}, \mathbf{e}^{(k+1)} \right\rangle_{A^m} = \left\langle \mathbf{e}^{(k)}, \mathbf{e}^{(k)} \right\rangle_{A^m} - 2\alpha_k \left\langle \mathbf{e}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{A^m} + \alpha_k^2 \left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{A^m},$$

and it is easy to see that the minimum of this is when

$$\alpha_k = \frac{\left\langle \mathbf{e}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{A^m}}{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{A^m}}.$$

We want to choose m such that this parameter is easy to calculate. We don't know what $\mathbf{e}^{(k)}$ is, but $A\mathbf{e}^{(k)} = \mathbf{r}^{(k)}$, so

$$\alpha_k = \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{A^{m-1}}}{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{A^m}}.$$

We therefore see that the simplest value we can take is $m = 1$, which will give

$$\alpha_k = \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle}{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle_A}.$$

This choice of α_k minimizes the error in the $A-$norm at each step.

Note that solving $A\mathbf{x} = \mathbf{b}$ is equivalent to minimizing the quadratic functional

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}, \tag{3.11}$$

and the gradient of ϕ at $\mathbf{x} = \mathbf{x}^{(k)}$ is the negative of the residual, $-\mathbf{r}^{(k)}$. The value α_k is then the optimal step length in this direction. For this reason this method – which is given as Algorithm 2 – is known as the method of *steepest descent*.

Algorithm 2 Method of steepest descent to solve $A\mathbf{x} = \mathbf{b}$

Choose $\mathbf{x}^{(0)}$
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
for $k = 0, 1, \ldots$ **until** convergence **do**
  $\alpha_k = \left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle / \left\langle A\mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{r}^{(k)}$
end for

Note that $\mathbf{r}^{(k)} = A\mathbf{e}^{(k)}$, so we can write (3.10) as

$$\mathbf{e}^{(k+1)} = (I - \alpha_k A)\mathbf{e}^{(k)}.$$

Therefore we can look at the error in the $A-$norm at the k^{th} step:

$$\begin{aligned}
\|\mathbf{e}^{(k)}\|_A^2 =& \|(I-\alpha_{k-1}A)\mathbf{e}^{(k-1)}\|_A^2 \\
=& \left\langle (I-\alpha_{k-1}A)\mathbf{e}^{(k-1)}, (I-\alpha_{k-1}A)\mathbf{e}^{(k-1)} \right\rangle_A \\
=& \left\langle \mathbf{e}^{(k-1)}, \mathbf{e}^{(k-1)} \right\rangle_A - 2\alpha_{k-1}\left\langle A\mathbf{e}^{(k-1)}, \mathbf{e}^{(k-1)} \right\rangle_A + \alpha_{k-1}^2 \left\langle A\mathbf{e}^{(k-1)}, A\mathbf{e}^{(k-1)} \right\rangle_A \\
=& \left\langle \mathbf{e}^{(k-1)}, \mathbf{e}^{(k-1)} \right\rangle_A - 2\frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle}{\left\langle A\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle} \left\langle A\mathbf{e}^{(k-1)}, \mathbf{e}^{(k-1)} \right\rangle_A \\
&+ \left(\frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle}{\left\langle A\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle} \right)^2 \left\langle A\mathbf{e}^{(k-1)}, A\mathbf{e}^{(k-1)} \right\rangle_A \\
=& \left\langle \mathbf{e}^{(k-1)}, \mathbf{e}^{(k-1)} \right\rangle_A - 2\frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle^2}{\left\langle A\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle} + \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle^2}{\left\langle A\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle} \\
=& \left(1 - \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle^2}{\left\langle A\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle \left\langle \mathbf{e}^{(k-1)}, \mathbf{e}^{(k-1)} \right\rangle_A} \right) \|\mathbf{e}^{(k-1)}\|_A^2 \\
=& \left(1 - \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle^2}{\left\langle A\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle \left\langle A^{-1}\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle} \right) \|\mathbf{e}^{(k-1)}\|_A^2.
\end{aligned}$$

Now, we apply the Kantorovich inequality [100, Lemma 5.1], which states

$$\frac{\langle \mathbf{v}, \mathbf{v} \rangle^2}{\langle A\mathbf{v}, \mathbf{v} \rangle \langle A^{-1}\mathbf{v}, \mathbf{v} \rangle} \geq \frac{4\lambda_{\min}\lambda_{\max}}{(\lambda_{\min}+\lambda_{\max})^2},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalues of A respectively, to get

$$\begin{aligned}
\|\mathbf{e}^{(k)}\|_A^2 &\leq \left(1 - \frac{4\lambda_{\min}\lambda_{\max}}{(\lambda_{\min}+\lambda_{\max})^2} \right) \|\mathbf{e}^{(k-1)}\|_A^2 \\
&= \left(\frac{\kappa-1}{\kappa+1} \right)^2 \|\mathbf{e}^{(k-1)}\|_A^2,
\end{aligned}$$

where, again, κ denotes the condition number of A, $\lambda_{\max}/\lambda_{\min}$. We can therefore say that

$$\|\mathbf{e}^{(k)}\|_A \leq \left(\frac{\kappa-1}{\kappa+1} \right)^k \|\mathbf{e}^{(0)}\|_A. \tag{3.12}$$

Compare this with the error bound for Richardson iteration in (3.8) – the upper bound is identical when you measure the error in the norm in which you are minimizing. However, whereas the Richardson iteration requires you to have some knowledge of the eigenvalues of the system beforehand, steepest descent requires no such information. The algorithm generates the optimal value of the parameter at each step.

Steepest descent can do this as it is a non-linear algorithm in the following sense. Suppose we want to find $\mathbf{x}_1$ and $\mathbf{x}_2$ satisfying

$$A\mathbf{x}_1 = \mathbf{b}_1 \quad \text{and} \quad A\mathbf{x}_2 = \mathbf{b}_2.$$

If we use steepest descent[2] then, for $i = 1, 2$, $\mathbf{r}_i^{(0)} = \mathbf{b}_i$ and

$$\mathbf{x}_i^{(1)} = \frac{\langle \mathbf{b}_i, \mathbf{b}_i \rangle}{\langle A\mathbf{b}_i, \mathbf{b}_i \rangle} \mathbf{b}_i.$$

Now, consider the equation

$$A\widehat{\mathbf{x}} = \mathbf{b}_1 + \mathbf{b}_2,$$

then clearly $\widehat{\mathbf{x}} = \mathbf{x}_1 + \mathbf{x}_2$. Here $\widehat{\mathbf{r}}^{(0)} = \mathbf{b}_1 + \mathbf{b}_2$, and

$$\begin{aligned}
\widehat{\mathbf{x}}^{(1)} &= \frac{\langle \mathbf{b}_1 + \mathbf{b}_2, \mathbf{b}_1 + \mathbf{b}_2 \rangle}{\langle A(\mathbf{b}_1 + \mathbf{b}_2), \mathbf{b}_1 + \mathbf{b}_2 \rangle} (\mathbf{b}_1 + \mathbf{b}_2) \\
&= \frac{\langle \mathbf{b}_1 + \mathbf{b}_2, \mathbf{b}_1 + \mathbf{b}_2 \rangle}{\langle A(\mathbf{b}_1 + \mathbf{b}_2), \mathbf{b}_1 + \mathbf{b}_2 \rangle} \mathbf{b}_1 + \frac{\langle \mathbf{b}_1 + \mathbf{b}_2, \mathbf{b}_1 + \mathbf{b}_2 \rangle}{\langle A(\mathbf{b}_1 + \mathbf{b}_2), \mathbf{b}_1 + \mathbf{b}_2 \rangle} \mathbf{b}_2 \\
&\neq \mathbf{x}_1^{(1)} + \mathbf{x}_2^{(1)} \quad \text{in general.}
\end{aligned}$$

The same is also clearly true for the higher iterations. In this sense, steepest descent is a non-linear method – i.e., there is no matrix S_i such that $\mathbf{x}^{(i)} = S_i \mathbf{b}$. In contrast, for the Richardson iteration we have a fixed α, and so such a corresponding matrix R_i does exist in the case where $\mathbf{x}^{(0)} = \mathbf{0}$, which is given explicitly by

$$R_i = (I - (I - \alpha A)^i) A^{-1}.$$

Of course we can combine steepest descent with a preconditioner, as we did with the Richardson iteration; this will give us Algorithm 3. Again, the preconditioner M must be symmetric positive-definite.

Algorithm 3 Preconditioned steepest descent to solve $A\mathbf{x} = \mathbf{b}$ with preconditioner M.

Choose $\mathbf{x}^{(0)}$
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
Solve $M\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$
for $k = 0, 1, \ldots$ **until** convergence **do**
  $\alpha_k = \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle / \left\langle A\mathbf{z}^{(k)}, \mathbf{z}^{(k)} \right\rangle$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{z}^{(k)}$
  Solve $M\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$
end for

[2]for simplicity we take $\mathbf{0}$ as the starting vector in both cases

3.1.4 Chebyshev Semi-iteration

Consider again the simple iteration in the form (3.3),

$$\mathbf{x}^{(k+1)} = (I - M^{-1}A)\mathbf{x}^{(k)} + M^{-1}\mathbf{b}.$$

When using this we keep only the last iterate, i.e. at the k^{th} step we only consider $\mathbf{x}^{(k)}$. We might be able to improve on the methods above by taking some linear combination of the previous iterates,

$$\mathbf{y}^{(k)} = \sum_{j=0}^{k} \beta_j^{(k)} \mathbf{x}^{(j)}. \tag{3.13}$$

What properties must the coefficients $\beta_j^{(k)}$ have so that this converges quickly?

We can write a simple iteration (3.3) in the form

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}, \tag{3.14}$$

where $T = I - M^{-1}A$ and $\mathbf{c} = M^{-1}\mathbf{b}$. If the initial guess was the exact solution, $\mathbf{x}$, then since we can write the error at the k^{th} iteration as

$$\mathbf{x} - \mathbf{x}^{(k)} = T^k(\mathbf{x} - \mathbf{x}^{(0)}), \tag{3.15}$$

we have that $\mathbf{x}^{(k)} = \mathbf{x}$ for all k. Furthermore, it's reasonable to expect that $\mathbf{y}^{(k)}$ should also be the exact solution for all k. By the definition of $\mathbf{y}^{(k)}$ (3.13) this corresponds to the constraint that

$$\sum_{j=0}^{n} \beta_j^{(n)} = 1. \tag{3.16}$$

With this constraint in mind, we look at the error at the k^{th} step:

$$\begin{aligned}
\mathbf{x} - \mathbf{y}^{(k)} &= \mathbf{x} - \sum_{j=0}^{n} \beta_j^{(k)} \mathbf{x}^{(j)} \\
&= \sum_{j=0}^{k} \beta_j^{(k)} \mathbf{x} - \sum_{j=0}^{k} \beta_j^{(k)} \mathbf{x}^{(j)} \qquad \text{[using(3.16)]} \\
&= \sum_{j=0}^{k} \beta_j^{(k)} (\mathbf{x} - \mathbf{x}^{(j)}) \\
&= \sum_{j=0}^{k} \beta_j^{(k)} T^j (\mathbf{x} - \mathbf{x}^{(0)}) \qquad \text{[using(3.15)]}, \\
\Rightarrow \quad \mathbf{x} - \mathbf{y}^{(k)} &= p_k(T)(\mathbf{x} - \mathbf{x}^{(0)})
\end{aligned} \tag{3.17}$$

where $p_k(z) = \sum_{j=0}^{k} \beta_j^{(k)} z^j$, a polynomial of degree k.

As in Section 3.1.1, consider the case where the iteration matrix is normal, i.e. there exists a basis of eigenvectors $\{\mathbf{v}_i\}$ such that $T\mathbf{v}_i = \lambda_i \mathbf{v}_i$. Then we can write $\mathbf{x} - \mathbf{x}^{(0)} = \sum \alpha_i \mathbf{v}_i$ and so

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}^{(k)}\|_2 &\leq \|\sum \alpha_i p_k(T)\mathbf{v}_i\|_2 \\ &= \|\sum \alpha_i p_k(\lambda_i)\mathbf{v}_i\|_2. \end{aligned}$$

The polynomial p_k is uniquely defined by the coefficients in (3.13). If we pick a polynomial which is small on the eigenvalues, then the error at the k^{th} step should also be small.

In the case where T is symmetric, we can make this more precise. A symmetric matrix is orthogonally diagonalizable, which means we can write $T = Q\Lambda Q^T$, where Q is orthogonal – its columns are eigenvectors – and Λ is diagonal – its entries are eigenvalues. Then $T^k = Q\Lambda^k Q^T$ for all k, and so

$$p(T) = Qp(\Lambda)Q^T$$

for any polynomial p, where $p(\Lambda) = \text{diag}(p(\lambda_1), p(\lambda_2), \ldots, p(\lambda_n))$. Thus

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}^{(k)}\|_2 &= \|p_k(T)(\mathbf{x} - \mathbf{x}^{(0)})\|_2 \\ &\leq \|p_k(T)\|_2 \|\mathbf{x} - \mathbf{x}^{(0)}\|_2 \\ &= \|Qp_k(\Lambda)Q^T\|_2 \|\mathbf{x} - \mathbf{x}^{(0)}\|_2 \\ &= \|p_k(\Lambda)\|_2 \|\mathbf{x} - \mathbf{x}^{(0)}\|_2 \qquad [Q \text{ orthogonal}] \\ &\leq \max_i |p_k(\lambda_i)| \|\mathbf{x} - \mathbf{x}^{(0)}\|_2. \end{aligned}$$

So if we have a p_k that is small on the eigenvalues, and such that $p_k(1) = 1$ (from condition (3.16)), then we should have fast convergence.

If we have explicit knowledge of the eigenvalues of T then finding such a polynomial would be easy, but what about the general case, where such information is not at hand?

The Chebyshev polynomials are defined for $t \in [-1, 1]$ by $T_0(t) = 1$ and

$$T_k(t) = \cos(k \cos^{-1} t).$$

This does, in fact, define a polynomial - using the identity

$$\cos(k+1)\theta + cos(k-1)\theta = 2\cos\theta\cos k\theta$$

and setting $\theta = \cos^{-1} t$ we get the recurrence relation

$$T_{k+1}(t) = 2tT_k(t) - T_{k-1}(t), \quad k = 1, 2, 3, \ldots, \ t \in [-1, 1]. \tag{3.18}$$

Applying this gives $T_1 = t$, $T_2 = 2t^2 - 1$, $T_3 = 4t^3 - 3t$ etc. Figure 3.2 shows plots of these polynomials T_i for various i. The Chebyshev polynomials can also be defined for $|t| > 1$ using the hyperbolic cosine. These polynomials define an orthonormal set with respect to the inner product $\langle p, q \rangle := \int_{-1}^{1} \frac{p(x)\,q(x)}{\sqrt{1-x^2}}\,dx$. For more information about Chebyshev polynomials see, for example, Mason and Handscomb [82].

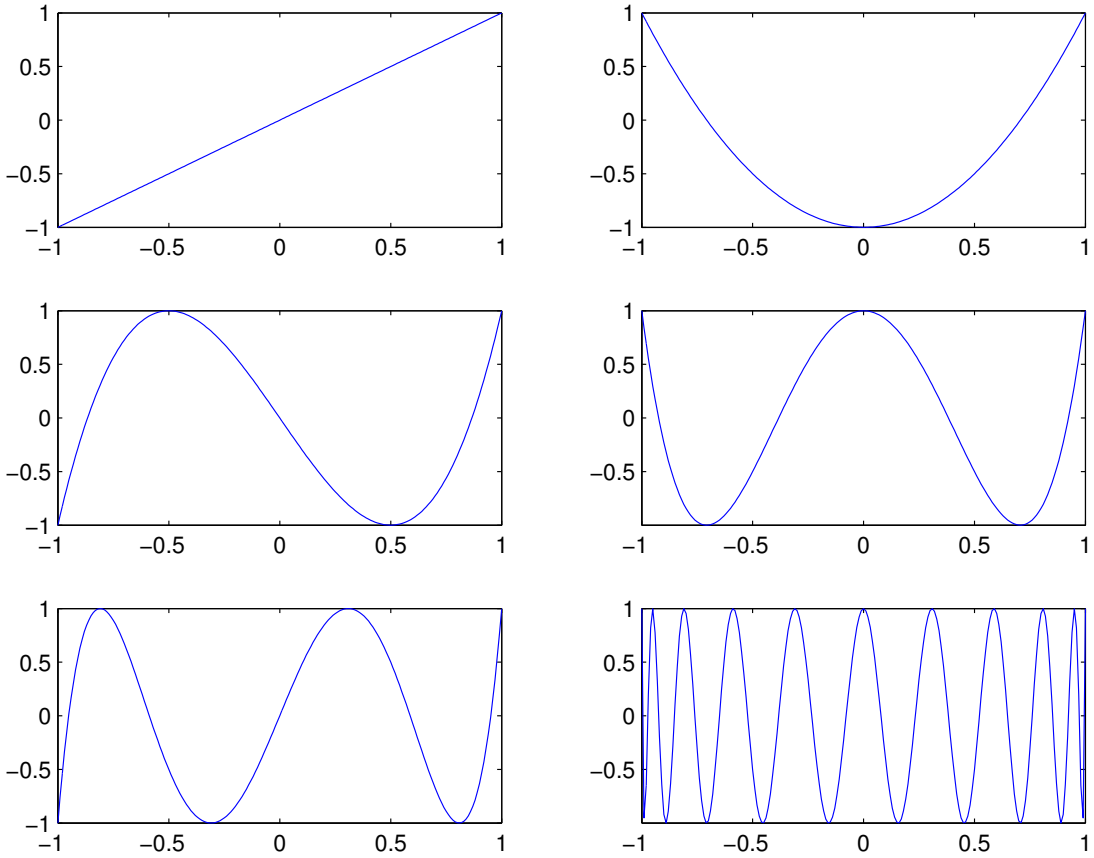


Figure 3.2: Plots of T_i, where $i = 1, 2, 3, 4, 5, 20$

The Chebyshev polynomials have many nice properties, but the property that makes them interesting in this context is that the polynomial which minimizes

$$\min_{p \in \Pi_k, p(1)=1} \left(\max_{t \in [-1,1]} |p(t)| \right)$$

is precisely the Chebyshev polynomial defined above.

It's unlikely that our iteration matrix T will have $\lambda(T) \in [-1, 1]$ precisely. However, suppose we know that $\lambda(T) \in [a, b]$, where 1 is not contained in $[a, b]$. Then

the optimal polynomial in this case will simply be the shifted and scaled Chebyshev polynomial, namely

$$\widehat{T_k}(s) := \frac{T_k\left(\frac{2s-a-b}{a-b}\right)}{T_k\left(\frac{2-a-b}{a-b}\right)}. \tag{3.19}$$

Then if we take $p_k = \widehat{T_k}$, then, without knowing anything more about the spectrum of T, this is the best we can do. Note that this polynomial tends to infinity very quickly outside $[a, b]$, so we must have good knowledge of the extremal values for the method to be useful. This choice of polynomial gives us the *Chebyshev semi-iteration* [51]. For more information, see Golub and van Loan [49, Section 10.1.5], Varga [123, Chapter 5] or [84, Section 5.8].

We now turn our attention to the error. Recall that if T is symmetric,

$$\begin{aligned}\|\mathbf{x} - \mathbf{y}^{(k)}\|_2 &\leq \max_i |p_k(\lambda_i)| \|\mathbf{x} - \mathbf{x}^{(0)}\|_2 \\ &\leq \max_{s\in[a,b]} |\widehat{T_k}(s)| \|\mathbf{x} - \mathbf{x}^{(0)}\|_2, \qquad [\text{if } \lambda_i \in [a,b] \ \forall\ i]\end{aligned}$$

and, further, $\max_{s\in[a,b]} |\widehat{T_k}(s)| = |\widehat{T_k}(a)| = |\widehat{T_k}(b)| = \left|\frac{T_k(1)}{T_k\left(\frac{2-a-b}{b-a}\right)}\right| = \frac{1}{\left|T_k\left(\frac{2-a-b}{b-a}\right)\right|}$, as the maximum error is always attained at the end points (see the plots in Figure 3.2). Therefore the convergence of the Chebyshev semi-iteration satisfies

$$\|\mathbf{x} - \mathbf{y}^{(k)}\|_2 \leq \frac{1}{\left|T_k\left(\frac{2-a-b}{b-a}\right)\right|} \|\mathbf{x} - \mathbf{x}^{(0)}\|_2. \tag{3.20}$$

Note that this bound only holds for problems where the iteration matrix, $T = I - M^{-1}A$ is symmetric. Similar behaviour will be observed if T is normal. If T is non-normal, this method is not guaranteed to converge [80].

Since $\lambda(T) \in [a, b]$, and $T = I - M^{-1}A$, we see that $\lambda(M^{-1}A) \in [1 - b, 1 - a]$. If we write $\hat{a} = 1 - b$ and $\hat{b} = 1 - a$, then we can write (3.20) as

$$\|\mathbf{x} - \mathbf{y}^{(k)}\|_2 \leq \frac{1}{\left|T_k\left(\frac{\hat{a}+\hat{b}}{\hat{b}-\hat{a}}\right)\right|} \|\mathbf{x} - \mathbf{x}^{(0)}\|_2. \tag{3.21}$$

It is well known – see e.g. [82, Section 1.4.2] – that we can write the k^{th} Chebyshev polynomial as

$$T_k(z) = \frac{1}{2}\left[(z + \sqrt{z^2 - 1})^k + (z - \sqrt{z^2 - 1})^k\right],$$

and so, if we write $\kappa := \kappa(M^{-1}A) = \hat{b}/\hat{a}$, the condition number of $M^{-1}A$, we get

$$\begin{aligned} T_k\left(\frac{\hat{b}+\hat{a}}{\hat{b}-\hat{a}}\right) &= T_k\left(\frac{\kappa+1}{\kappa-1}\right) \\ &= \frac{1}{2}\left[\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k\right] \\ &\geq \frac{1}{2}\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k. \end{aligned}$$

We have therefore shown that the bound (3.20) can be written in terms of the condition number of $M^{-1}A$ as

$$\|\mathbf{x}-\mathbf{y}^{(k)}\|_2 \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|\mathbf{x}-\mathbf{x}^{(0)}\|_2. \tag{3.22}$$

We can now estimate how fast this method converges, and (3.19) gives a way to compute the coefficients $\beta_j^{(k)}$ from the Chebyshev polynomial of the required degree. However, we can use the recurrence relation for Chebyshev polynomials (3.18) to give a more efficient algorithm. Suppose first that the eigenvalues of the iteration matrix lie in an interval which is symmetric about the origin, so $\lambda_i \in [-\rho, \rho]$ for some ρ. Substituting (3.19) into (3.18) gives

$$T_{k+1}\left(\frac{1}{\rho}\right) p_{k+1}(s) = \frac{2s}{\rho} T_k\left(\frac{1}{\rho}\right) p_k(s) - T_{k-1}\left(\frac{1}{\rho}\right) p_{k-1}(s).$$

Multiplying this on the right by the initial error, $\mathbf{e}^{(0)}$, gives

$$T_{k+1}\left(\frac{1}{\rho}\right) \boldsymbol{\eta}^{(k+1)} = \left(\frac{2}{\rho}\right) T_k\left(\frac{1}{\rho}\right) T\boldsymbol{\eta}^{(k)} - T_{k-1}\left(\frac{1}{\rho}\right) \boldsymbol{\eta}^{(k-1)},$$

where $\boldsymbol{\eta}^{(k)} = \mathbf{x} - \mathbf{y}^{(k)}$. We can simplify this to obtain

$$\mathbf{y}^{(k+1)} = \frac{2T_k\left(\frac{1}{\rho}\right)}{\rho T_{k+1}\left(\frac{1}{\rho}\right)}(T\mathbf{y}^{(k)} + \mathbf{c}) - \frac{T_{k-1}\left(\frac{1}{\rho}\right)}{T_{k+1}\left(\frac{1}{\rho}\right)}\mathbf{y}^{(k-1)},$$

where we have used the fact that

$$\frac{2T_k\left(\frac{1}{\rho}\right)}{\rho T_{k+1}\left(\frac{1}{\rho}\right)}\mathbf{x} = \frac{T_{k-1}\left(\frac{1}{\rho}\right)}{T_{k+1}\left(\frac{1}{\rho}\right)}\mathbf{x} + \mathbf{x}. \tag{3.23}$$

This shows us that we can use the Chebyshev semi-iteration without having to calculate the iterates $\mathbf{x}^{(k)}$ of the underlying iterative method. This can be further simplified

by noting that

$$\mathbf{y}^{(k+1)} = \frac{2T_k\left(\frac{1}{\rho}\right)}{\rho T_{k+1}\left(\frac{1}{\rho}\right)}(T\mathbf{y}^{(k)} + \mathbf{c} - \mathbf{y}^{(k-1)}) - \left(\frac{2T_k\left(\frac{1}{\rho}\right)}{\rho T_{k+1}\left(\frac{1}{\rho}\right)} - \frac{T_{k-1}\left(\frac{1}{\rho}\right)}{T_{k+1}\left(\frac{1}{\rho}\right)}\right)\mathbf{y}^{(k-1)},$$

and so, again using (3.23), we get

$$\mathbf{y}^{(k+1)} = w_{k+1}(T\mathbf{y}^{(k)} + \mathbf{c} - \mathbf{y}^{(k-1)}) + \mathbf{y}^{(k-1)},$$

where $w_{k+1} = \frac{2T_k\left(\frac{1}{\rho}\right)}{\rho T_{k+1}\left(\frac{1}{\rho}\right)}$ and $w_1 = 1$.

Note also that

$$w_{k+1} = \frac{2T_k\left(\frac{1}{\rho}\right)}{2T_k\left(\frac{1}{\rho}\right) - \rho T_{k-1}\left(\frac{1}{\rho}\right)} = \frac{1}{1 - \frac{\rho^2 w_k}{4}},$$

so we do not even need to compute the Chebyshev polynomials to apply this method. Algorithm 4 gives this procedure.

Algorithm 4 Chebyshev semi-iteration to solve $A\mathbf{x} = \mathbf{b}$, where $\lambda(T) \in [-\rho, \rho]$

Choose $\mathbf{y}^{(0)}$, $w_0 = 0$, $(\mathbf{y}^{(-1)} = \mathbf{0})$
for $k = 0, 1, \ldots$ **until** convergence **do**
  $w_{k+1} = \frac{1}{1 - \frac{\rho^2 w_k}{4}}$
  $M\mathbf{z}^{(k)} = \mathbf{b} - A\mathbf{y}^{(k)}$
  $\mathbf{y}^{(k+1)} = w_{k+1}(\mathbf{z}^{(k)} + \mathbf{y}^{(k)} - \mathbf{y}^{(k-1)}) + \mathbf{y}^{(k-1)}$
end for

Of course, the above algorithm is only for the case where the largest and smallest eigenvalues of the iteration matrix are symmetric about the origin; in general, if $\lambda(T) \in [a, b]$, then the above argument can be modified to give Algorithm 5, which is the algorithm as presented in [84].

The main drawback of the Chebyshev semi-iteration is, of course, that you need to know explicit information about the spectrum of the iteration matrix. Because of the oscillatory nature of the Chebyshev polynomials, even if we have an estimate of the largest and smallest eigenvalues, that may not be enough to guarantee us fast convergence. Nevertheless, as we shall see in Section 5.2, for example, there are situations in which the Chebyshev semi-iteration is well suited.

So far we have considered the Chebyshev semi-iteration applied to a symmetric matrix, which has real eigenvalues. Manteuffel [80, 81] showed that the Chebyshev semi-iteration can be applied to a non-symmetric matrix, as long as the ellipse that

Algorithm 5 Chebyshev semi-iteration to solve $A\mathbf{x} = \mathbf{b}$, where $\lambda(T) \in [a, b]$

Choose $\mathbf{y}^{(0)}$, $(\mathbf{y}^{(-1)} = \mathbf{0})$
$w_0 = 0$, $w = \frac{2-(a+b)}{b-a}$
for $k = 0, 1, \ldots$ **until** convergence **do**
  if $k = 1$ **then**
    $w_{k+1} = 1/(1 - \frac{1}{2w^2})$
  else
    $w_{k+1} = 1/(1 - \frac{w_k^2}{4w^2})$
  end if
  $M\mathbf{z}^{(k)} = \mathbf{b} - A\mathbf{y}^{(k)}$
  $\mathbf{y}^{(k+1)} = w_{k+1}(\frac{2}{2-(a+b)}\mathbf{z}^{(k)} + \mathbf{y}^{(k)} - \mathbf{y}^{(k-1)}) + \mathbf{y}^{(k-1)}$
end for

contains the eigenvalues (or in general the field of values) is known. He also proposed an adaptive method to determine such an ellipse from a given non-symmetric matrix, which was later implemented by Ashby [4].

Note that as the scaled and shifted Chebyshev polynomials only depend on the extremal eigenvalues of the matrix this method is – unlike Steepest Descent – a linear method, in the same sense as described in Section 3.1.2.

3.1.5 Conjugate Gradients

We saw in the previous section that the Chebyshev semi-iteration, as given in Algorithm 5, is a method that finds iterates which satisfy

$$\mathbf{x}^{(k+1)} = w_{k+1}(\alpha\mathbf{z}^{(k)} + \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) + \mathbf{x}^{(k-1)},$$

where $M\mathbf{z}^{(k+1)} = \mathbf{b} - A\mathbf{x}^{(k+1)}$ and for a specific choice of w_{k+1} and α. In much the same way that we generalized the Richardson iteration to obtain the steepest descent algorithm in Section 3.1.3, we could allow the constant α above to vary at each step gives us

$$\mathbf{x}^{(k+1)} = w_{k+1}(\alpha_k\mathbf{z}^{(k)} + \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) + \mathbf{x}^{(k-1)},$$

where $\mathbf{z}^{(k)}$ is defined as before. Is there some choice of w_{k+1} and α_k we can choose so that we get good convergence properties here?

As we have done in the previous sections, let us first consider the case where we have no preconditioner, i.e. $M = I$. The iteration now becomes

$$\mathbf{x}^{(k+1)} = w_{k+1}(\alpha_k\mathbf{r}^{(k)} + \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) + \mathbf{x}^{(k-1)},$$

where, as usual, $\mathbf{r}^{(k)}$ is the residual. One thing we could try to look for is a method that makes the residual vectors all mutually orthogonal, i.e.

$$\left\langle \mathbf{r}^{(i)}, \mathbf{r}^{(j)} \right\rangle = 0 \quad \text{if } i \neq j. \tag{3.24}$$

It can be shown [84, Theorem 6.4] that if A is symmetric positive definite, the choice of parameters

$$\alpha_k = \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle}{\left\langle A\mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle}, \; w_{k+1} = 1/\left(1 + \alpha_k \frac{\left\langle A\mathbf{r}^{(k-1)}, \mathbf{r}^{(k)} \right\rangle}{\left\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle}\right),$$

generates a sequence of iterates with the required orthogonality properties. As shown in [84, Section 6.1], for example, this can be made more efficient for computation by taking the equivalent definition

$$w_{k+1} = 1/\left(1 - \frac{\alpha_k \left\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k)} \right\rangle}{w_k \alpha_{k-1} \left\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle}\right).$$

The resulting algorithm is called the Conjugate Gradient algorithm, and was first developed by Hestenes and Steifel in 1952 [64]. We give the method as Algorithm 6.

Algorithm 6 Conjugate Gradients (version 1) to solve $A\mathbf{x} = \mathbf{b}$

Choose $\mathbf{x}^{(0)}$, $(\mathbf{x}^{(-1)} = \mathbf{0})$
$w_1 = 1$
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
for $k = 0, 1, \ldots$ **until** convergence **do**
  $\alpha_k = \left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle / \left\langle A\mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle$
  $w_{k+1} = 1/\left(1 - \frac{\alpha_k \left\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k)} \right\rangle}{w_k \alpha_{k-1} \left\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \right\rangle}\right)$
  $\mathbf{x}^{(k+1)} = w_{k+1}(\alpha_k \mathbf{r}^{(k)} + \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) + \mathbf{x}^{(k-1)}$
  $\mathbf{r}^{(k+1)} = \mathbf{b} - A\mathbf{x}^{(k+1)}$
end for

Conjugate gradients is therefore a generalization of acceleration techniques, such as the Chebyshev semi-iteration, in much the same way that the steepest descent algorithm is a generalization of the Richardson iteration.

The above formulation of conjugate gradients was shown by Concus, Golub and O'Leary [30]. However, as was the case with the steepest descent algorithm, the conjugate gradient algorithm can also thought of as a method that finds the minimum of a quadratic functional, and that's the context in which Hestenes and Steifel first considered the algorithm in 1952 [64]. This second derivation gives rise to the version of the algorithm that is generally used in practice.

As we noted in (3.11), if A is symmetric positive definite, finding the solution of $A\mathbf{x} = \mathbf{b}$ is equivalent to finding the minimum of the quadratic functional

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}.$$

The method of steepest descent can be derived by taking the residual $\mathbf{r}^{(k)}$ as the search direction at each step. For conjugate gradients we look for alternative search directions, $\mathbf{p}^{(k)}$, in the hope of getting faster convergence. We therefore find the iterate at the k^{th} step by $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{p}^{(k)}$, where α_k is the optimal step-length in the direction $\mathbf{p}^{(k)}$. When we think of solving the linear system as a minimization problem, the optimal α_k is found by solving the one-dimensional minimization problem

$$\alpha_k = \min_{\alpha\in\mathbb{R}} \phi(\mathbf{x}^{(k)} + \alpha\mathbf{p}^{(k)}).$$

Simple calculus shows that solution of this, for any search direction $\mathbf{p}^{(k)}$, is given by

$$\alpha_k = \frac{\left\langle \mathbf{p}^{(k)}, \mathbf{r}^{(k)} \right\rangle}{\left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle}.$$

We now turn our attention to finding a good choice of search direction, $\mathbf{p}^{(k)}$. For steepest descent, we took $\mathbf{p}^{(k)} = \mathbf{r}^{(k)}$, and this made $\left\langle \mathbf{e}^{(k+1)}, \mathbf{r}^{(k)} \right\rangle_A = 0$ for all k. However, $\mathbf{e}^{(k+1)}$ is not $A-$orthogonal to the previous search direction, $\mathbf{r}^{(k-1)}$. If we instead take $\mathbf{p}^{(k)}$ to be

$$\mathbf{p}^{(k)} = \mathbf{r}^{(k)} - \frac{\left\langle A\mathbf{r}^{(k)}, \mathbf{p}^{(k-1)} \right\rangle}{\left\langle A\mathbf{p}^{(k-1)}, \mathbf{p}^{(k-1)} \right\rangle}\mathbf{p}^{(k-1)}$$

then this choice of $\mathbf{p}^{(k)}$ has the property that the error at the $(k+1)^{\text{th}}$ step is $A-$orthogonal to the previous two search directions, i.e.

$$\left\langle \mathbf{e}^{(k+1)}, \mathbf{p}^{(k)} \right\rangle_A = \left\langle \mathbf{e}^{(k+1)}, \mathbf{p}^{(k-1)} \right\rangle_A = 0.$$

This choice of search direction therefore minimizes $\|\mathbf{e}^{(k)}\|_A$ over the two-dimensional affine space $\mathbf{e}^{(k)} + \text{span}\{\mathbf{r}^{(k)}, \mathbf{p}^{(k-1)}\}$. Moreover, since we have assumed that A is positive definite it can be shown [53, Theorem 2.3.2] that $\|\mathbf{e}^{(k)}\|_A$ is actually minimized over the much larger space $\mathbf{e}^{(0)}+\text{span}\{\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(k)}\}$. We can replace the intuitive formula above by the equivalent

$$\alpha_k = \frac{\left\langle \mathbf{p}^{(k)}, \mathbf{r}^{(k)} \right\rangle}{\left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle} = \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle}{\left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle}, \quad \frac{\left\langle A\mathbf{r}^{(k+1)}, \mathbf{p}^{(k)} \right\rangle}{\left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle} = -\frac{\left\langle A\mathbf{r}^{(k+1)}, \mathbf{p}^{(k)} \right\rangle}{\left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle} := -\beta_k,$$

then we get the conjugate gradient algorithm in a more computationally efficient from, Algorithm 7.

Algorithm 7 Conjugate Gradients to solve $A\mathbf{x} = \mathbf{b}$

Choose $\mathbf{x}^{(0)}$
Compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
Set $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$
for $k = 0, 1, 2...$**until** convergence **do**
  $\alpha_k = \left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle / \left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}$
  TEST FOR CONVERGENCE
  $\beta_k = \left\langle \mathbf{r}^{(k+1)}, \mathbf{r}^{(k+1)} \right\rangle / \left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle$
  $\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta_k \mathbf{p}^{(k)}$
end for

One can show [53, Theorem 2.3.2] that the method described in Algorithm 7 satisfies

$$\left\langle A\mathbf{e}^{(k+1)}, \mathbf{p}^{(j)} \right\rangle = \left\langle A\mathbf{p}^{(k+1)}, \mathbf{p}^{(j)} \right\rangle = \left\langle \mathbf{r}^{(k+1)}, \mathbf{r}^{(j)} \right\rangle = 0 \quad \forall j \leq k.$$

Note that the last of these is the same as (3.24), which was the property we required to derive the iterates in Algorithm 6. In fact, it can be shown that the two methods are indeed equivalent [84, Theorem 6.8]. Also we can see that $\text{span}\{\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(k)}\} = \text{span}\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, \ldots, A^{k-1}\mathbf{r}^{(0)}\}$. We call spaces of this type *Krylov subspaces*, and introduce the notation

$$\mathcal{K}_k(A, \mathbf{z}) = \text{span}\{\mathbf{z}, A\mathbf{z}, \ldots, A^{k-1}\mathbf{z}\}.$$

Therefore, at the k^{th} step, Algorithm 7 finds the unique vector $\mathbf{x}^{(k)}$ that minimizes $\|\mathbf{e}^{(k)}\|_A$ over all vectors in the affine space $\mathbf{x}^{(0)} + \mathcal{K}_k(A, \mathbf{r}^{(0)})$. For this reason, the conjugate gradient method (or CG) is referred to as a *Krylov subspace method*.

The conjugate gradient method implicitly defines a sequence of polynomials $q_j \in \Pi_j$ – where Π_m denotes the set of polynomials of degree at most m – such that the iterates

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + q_{k-1}(A)\mathbf{r}^{(0)}$$

are successively closer to $\mathbf{x} = A^{-1}\mathbf{b}$. We can write the error at the k^{th} step as

$$\mathbf{e}^{(k)} = \mathbf{e}^{(0)} - q_{k-1}(A)\mathbf{r}^{(0)} = (I - Aq_{k-1}(A))\mathbf{e}^{(0)}.$$

We can therefore write the error as

$$\mathbf{e}^{(k)} = p_k(A)\mathbf{e}^{(0)},$$

where $p_k(z)$ is a polynomial of degree k with constant term 1. Recall that we have defined the next iterate of the conjugate gradient method as that which minimizes the error in the A norm, so we have

$$\|\mathbf{e}^{(k)}\|_A = \min_{p_k \in \Pi_k, p_k(0)=1} \|p_k(A)\mathbf{e}^{(0)}\|_A,$$

and so if we expand $\mathbf{e}^{(0)}$ in terms of a basis consisting of eigenvectors of A then we get $\mathbf{e}^{(0)} = \sum_{j=1}^n a_i \mathbf{v}_j$ (where $A\mathbf{v}_j = \lambda_j \mathbf{v}_j$). Now,

$$\begin{aligned}
\|\mathbf{e}^{(k)}\|_A &= \min_{p_k \in \Pi_k, p_k(0)=1} \left\| \sum_{j=1}^n a_i p_k(\lambda_j)\mathbf{v}_j \right\|_A \\
&\leq \min_{p_k \in \Pi_k, p_k(0)=1} \left\| \max_j |p_k(\lambda_j)| \sum_{j=1}^n a_i \mathbf{v}_j \right\|_A \\
&= \min_{p_k \in \Pi_k, p_k(0)=1} \max_j |p_k(\lambda_j)| \ \|\mathbf{e}^{(0)}\|_A. \qquad (3.25)
\end{aligned}$$

Thus we can say that

$$\frac{\|\mathbf{e}^{(k)}\|_A}{\|\mathbf{e}^{(0)}\|_A} = \min_{p_k \in \Pi_k, p_k(0)=1} \max_j |p_k(\lambda_j)|. \qquad (3.26)$$

To get an error bound, suppose that the eigenvalues are such that $\lambda_i \in [a, b]$, and the interval is tight (i.e. $a = \lambda_{\min}(A)$ and $b = \lambda_{\max}(A)$). Then we must have

$$\frac{\|\mathbf{e}^{(k)}\|_A}{\|\mathbf{e}^{(0)}\|_A} \leq \min_{p_k \in \Pi_k, p_k(0)=1} \max_{z \in [a,b]} |p_k(z)|,$$

and, as in Section 3.1.4, the polynomial that achieves this minimum is the scaled and shifted Chebyshev polynomial, this time given by

$$p_k(z) = T_k\left(\frac{b+a-2z}{b-a}\right) \Big/ T_k\left(\frac{b+a}{b-a}\right).$$

As argued with the Chebyshev semi-iteration

$$\max_{z \in [a,b]} T_k\left(\frac{b+a-2t}{b-a}\right) \Big/ T_k\left(\frac{b+a}{b-a}\right) = 1 \Big/ T_k\left(\frac{b+a}{b-a}\right).$$

This is the same form as (3.21), and we can use the same argument here to say

$$1/T_k\left(\frac{b+a}{b-a}\right) \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k,$$

where $\kappa = \kappa(A)$, the condition number of A. Therefore we have shown that the error for the CG iterates satisfies

$$\frac{\|\mathbf{e}^{(k)}\|_A}{\|\mathbf{e}^{(0)}\|_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k, \qquad (3.27)$$

which is the same estimation of convergence as the Chebyshev semi-iteration (3.22) – although we measured that convergence in the 2-norm. Note again how this parallels the convergence of steepest descent compared with Richardson iteration.

As with steepest descent, we need no spectral information for CG to work, however, which was the case with the Chebyshev semi-iteration. Also, CG is a non-linear method in the same sense as steepest descent was, whereas the Chebyshev semi-iteration is a linear iteration.

Note also that the bound (3.27) almost always underestimates the speed of convergence; only in the case where the eigenvalues of A are uniformly spaced in an interval will we see this convergence in practice. For example, if we have a system of size n with only two eigenvalues, λ and μ, say (hence having arbitrary condition number), then we can construct a polynomial of degree two, $p_2(z)$, such that $p_2(0) = 1$ and $p_2(\lambda) = p_2(\mu) = 0$. Thus, by (3.26), the error in the conjugate gradient algorithm will be zero after just two iterations, regardless of the condition number of the matrix – we will revisit this idea in our preconditioning strategies in Chapter 5.

The power of CG lies in the fact that the algorithm automatically picks the optimal polynomial without any prior information about the spectrum, as opposed to polynomial iterative methods like the Chebyshev semi-iteration, where you have to manually choose an appropriate polynomial. Because of this it is characteristic to see superlinear convergence with CG. In the cases where CG does converge linearly, the convergence of the Chebyshev semi-iteration, when measured in the appropriate norm for the method, will be exactly the same.

Of course, as was the case with the methods above, the concept of preconditioning can be applied to CG to give the preconditioned conjugate gradient algorithm, which we shall abbreviate as PCG. Here, as was the case with the Richardson iteration in Section 3.1.2, we want to preserve symmetry, so we must use a symmetric positive definite preconditioner and, formally, solve

$$H^{-1}AH^{-T}\mathbf{y} = H^{-1}\mathbf{b}, \mathbf{y} = H^T\mathbf{x}$$

where the preconditioner M has been decomposed as $M = HH^T$. As was the case previously, however, all that is needed in practice is to solve a system with M. The preconditioned conjugate gradient algorithm is given in Algorithm 8.

Note that at each iteration of Algorithm 8 we have to solve $M\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$, so the ability to efficiently solve systems with M is a prerequisite for M to be an effective preconditioner. Note that this is the only additional work needed in the preconditioned version compared to Algorithm 7. As discussed in the section on

Algorithm 8 Preconditioned Conjugate Gradients to solve $A\mathbf{x} = \mathbf{b}$ with preconditioner M

Choose $\mathbf{x}^{(0)}$
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
$M\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$
Set $\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$
for $k = 0, 1, 2...$**until** convergence **do**
  $\alpha_k = \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle / \left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}$
  TEST FOR CONVERGENCE
  Solve $M\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$
  $\beta_k = \left\langle \mathbf{z}^{(k+1)}, \mathbf{r}^{(k+1)} \right\rangle / \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle$
  $\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} + \beta_k \mathbf{p}^{(k)}$
end for

convergence above, the aim for a good preconditioner is that M should be a good approximation to A in the sense that the eigenvalues λ of the generalized eigenvalue problem $A\mathbf{v} = \lambda M\mathbf{v}$ are well clustered. In this case the minimum polynomial, which CG implicitly finds, will be small when evaluated at the eigenvalues after a small number of iterations, and so Algorithm 8 will converge in far fewer iterations than Algorithm 7. The art of preconditioning is trying to balance these two competing requirements. There are a number of strategies that have been developed that meet these criteria – the most effective usually depend on exploiting the structure of the matrix A, and so are problem-specific. We shall discuss some of these methods in Section 3.2 and Chapters 5 to 7. For more preconditioning techniques see, for example, the books by Meurant [84, Chapter 8] or Greenbaum [53, Section II].

3.1.6 CG with a non-standard inner product

Our description of the conjugate gradient algorithm above relied heavily on the matrix A being symmetric positive definite. In fact, we can weaken this criterion slightly; all that is required is that the matrix A is self-adjoint in the inner product that is used in the algorithm. In the discussion above we have only considered the Euclidean inner product, in which case self-adjointness and symmetry are equivalent. However, there's no reason why we shouldn't use any other inner product. Let us consider inner products of the form

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}} = \langle \mathcal{H}\mathbf{x}, \mathbf{y} \rangle ,$$

for some symmetric positive definite matrix $\mathcal{H}$. Then the condition for self-adjointness is

$$\langle A\mathbf{x}, \mathbf{y}\rangle_{\mathcal{H}} = \langle \mathbf{x}, A\mathbf{y}\rangle_{\mathcal{H}} \Leftrightarrow \mathcal{H}A = A^T\mathcal{H}.$$

Also, if we are to use CG we need $\mathcal{H}A$ to be positive definite. If these conditions on $\mathcal{H}$ are satisfied, then conjugate gradients will work with a non-standard inner product of this form. The method is given as Algorithm 9.

Algorithm 9 Conjugate Gradients with a $\mathcal{H}$ inner product to solve $A\mathbf{x} = \mathbf{b}$

Choose $\mathbf{x}^{(0)}$
Compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
Set $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$
for $k = 0, 1, 2...$**until** convergence **do**
  $\alpha_k = \left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{\mathcal{H}} / \left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle_{\mathcal{H}}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}$
  TEST FOR CONVERGENCE
  $\beta_k = \left\langle \mathbf{r}^{(k+1)}, \mathbf{r}^{(k+1)} \right\rangle_{\mathcal{H}} / \left\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{\mathcal{H}}$
  $\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta_k \mathbf{p}^{(k)}$
end for

It can be shown [48, Corollary 5.2], [27] that for any square matrix A with real entries, there exists a real symmetric matrix $\mathcal{H}$ such that $\mathcal{H}A = A^T\mathcal{H}$. Of course, constructing a matrix $\mathcal{H}$ which satisfies the required properties is not trivial. There are only a handful of examples in the literature – the most famous being the Bramble-Pasciack method for solving the Stokes equations [20], which appears to be where this concept was introduced. We will return to this application in Chapter 7.

Of course, we can again use this method with a preconditioner. It is no longer necessary to maintain symmetry, but rather self-adjointness, so consider left preconditioning, i.e. solving the system

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}.$$

Then if we label $\widehat{A} = M^{-1}A$, then the condition for $\widehat{A}$ to be $\mathcal{H}$-symmetric is that $\mathcal{H}\widehat{A} = \widehat{A}^T\mathcal{H}$. The preconditioned CG method with a non-standard inner product is given in Algorithm 10.

3.1.7 MINRES

We saw in the previous section that the conjugate gradient algorithm chooses the vector $\mathbf{x}^{(k)}$ from the Krylov subspace $\mathbf{x}^{(0)} + \mathcal{K}_k(A, \mathbf{r}^{(0)})$ that minimizes the error in

Algorithm 10 Preconditioned Conjugate Gradients with a $\mathcal{H}$ inner product to solve $A\mathbf{x} = \mathbf{b}$ with preconditioner M

Choose $\mathbf{x}^{(0)}$
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
$M\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$
Set $\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$
for $k = 0, 1, 2...$**until** convergence **do**
  $\alpha_k = \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{\mathcal{H}} / \left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle_{\mathcal{H}}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}$
  TEST FOR CONVERGENCE
  Solve $M\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$
  $\beta_k = \left\langle \mathbf{z}^{(k+1)}, \mathbf{r}^{(k+1)} \right\rangle_{\mathcal{H}} / \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle_{\mathcal{H}}$
  $\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} + \beta_k \mathbf{p}^{(k)}$
end for

the A-norm over this space. It can be shown (e.g. [84, Section 6.5]) that CG is just a re-scaling of the Lanczos algorithm. This is a method, which was developed by Cornelius Lanczos in 1950 [77], for finding an orthogonal basis for $\mathcal{K}_k(A, \mathbf{r}^{(0)})$ based on the recurrence

$$\gamma_{k+1}\mathbf{v}^{(k+1)} = A\mathbf{v}^{(k)} - \delta_k \mathbf{v}^{(k)} - \gamma_{k-1}\mathbf{v}^{(k-1)}, \tag{3.28}$$

where $\delta_j = \left\langle A\mathbf{v}^{(j)}, \mathbf{v}^{(j)} \right\rangle$ and γ_{j+1} is chosen to normalize $\mathbf{v}^{(j+1)}$.

If we define $V := [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(k)}]$ and $T_k := \texttt{tridiag}(\gamma_k, \delta_k, \gamma_{k+1})$, then an alternative statement of (3.28) is

$$AV_k = V_k T_k + \gamma_{k+1}[\mathbf{0}, \ldots, \mathbf{0}, \mathbf{v}^{(k+1)}] = V_{k+1}\widehat{T}_k. \tag{3.29}$$

Here $\widehat{T}_k = \begin{bmatrix} T_k \\ \gamma_{k+1}\mathbf{e}_k \end{bmatrix} \in \mathbb{R}^{k+1\times k}$, where $\mathbf{e}_i$ denotes the unit vector where the i^{th} entry is the only non-zero.

We saw in the previous section that CG can only be applied to symmetric positive definite matrices; can we use the Lanczos process above to derive a method that allows us to relax the positive definite requirement?

Suppose that instead of minimizing $\|\mathbf{e}^{(k)}\|_A$ over the Krylov subspace we pick $\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathcal{K}_k(A, \mathbf{r}^{(0)})$ such that $\|\mathbf{r}^{(k)}\|_2$ is a minimum. Note that we can write $\mathbf{x}^{(k)}$ as

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + V_k \mathbf{z}^{(k)},$$

for some vector of coefficients $\mathbf{z}^{(k)}$. Now,

$$\begin{aligned}\|\mathbf{r}^{(k)}\|_2 &= \|\mathbf{b} - A\mathbf{x}^{(k)}\|_2 \\ &= \|\mathbf{b} - A(\mathbf{x}^{(0)} + V_k\mathbf{z}^{(k)})\|_2 \\ &= \|\mathbf{r}^{(0)} - AV_k\mathbf{z}^{(k)}\|_2 \\ &= \|\mathbf{r}^{(0)} - V_{k+1}\widehat{T}_k\mathbf{z}^{(k)}\|_2.\end{aligned}$$

Note that $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|_2$, so we have

$$\begin{aligned}\|\mathbf{r}^{(k)}\|_2 &= \|\mathbf{v}^{(1)}\|\mathbf{r}^{(0)}\|_2 - V_{k+1}\widehat{T}_k\mathbf{z}^{(k)}\|_2 \\ &= \|V_{k+1}(\|\mathbf{r}^{(0)}\|_2\mathbf{e}_1 - \widehat{T}_k\mathbf{z}^{(k)})\|_2 \\ &= \|\|\mathbf{r}^{(0)}\|_2\mathbf{e}_1 - \widehat{T}_k\mathbf{z}^{(k)}\|_2,\end{aligned}$$

where the last equality comes from the fact that V_{k+1} has orthogonal columns. Therefore we have that

$$\min \|\mathbf{r}^{(k)}\|_2 = \min \|\|\mathbf{r}^{(0)}\|_2\mathbf{e}_1 - \widehat{T}_k\mathbf{z}^{(k)}\|_2.$$

Finding $\mathbf{z}^{(k)}$, and hence $\mathbf{r}^{(k)}$ is therefore just solving a linear least squares problem; this can be done with a QR factorization, achieved with just one Givens rotation at each iteration [43, p. 179]. The resulting algorithm is the Minimal Residual (*MINRES*) method of Paige and Saunders [91], and is a robust Krylov subspace method for solving systems of the form (3.1) where the matrix A is symmetric. Algorithm 11 is an implementation of MINRES as described above – the version presented here can be found in the book by Elman, Silvester and Wathen [39, Algorithm 2.4]. Notice by comparing Algorithm 11 with Algorithm 7 that MINRES is only slightly more work than conjugate gradients.

The property that characterizes this Krylov subspace method is that

$$\|\mathbf{r}_k\|_2 \leq \|\mathbf{s}\|_2 \;\forall\; \mathbf{s} \in \mathbf{r}_0 + \text{span}\{A\mathbf{r}_0, ..., A^k\mathbf{r}_0\}$$

and, using this, if we follow a convergence analysis such as in Section 3.1.5 we get the following convergence bound in terms of the eigenvalue interval $\lambda_j \in [a, b]$:

$$\|\mathbf{r}_k\|_2 \leq \min_{p_k \in \Pi_k, p_k(0)=1} \max_{z \in [a,b]} |p_k(z)| \; \|\mathbf{r}_0\|_2. \tag{3.30}$$

Thus, the comments about the possible preconditioning strategies made in Section 3.1.5 are equally relevant here. In fact, the bound (3.30) generalizes the CG theory, as this remains valid in the case where $0 \in [a, b]$, and in the positive definite case the MINRES and the CG bounds differ only in the definition of the norm. With MINRES,

Algorithm 11 MINRES

$\mathbf{v}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(1)} = \mathbf{0}$
Choose $\mathbf{x}^{(0)}$, compute $\mathbf{v}^{(1)} = \mathbf{b} - A\mathbf{x}^{(0)}$, set $\gamma_1 = ||\mathbf{v}^{(1)}||$
Set $\eta = \gamma_1$, $,s_0 = s_1 = 0$, $c_0 = c_1 = 1$
for $j = 1, 2...$**until** convergence **do**
  $\mathbf{v}^{(j)} = \mathbf{v}^{(j)}/\gamma_j$
  $\delta_j = \left\langle A\mathbf{v}^{(j)}, \mathbf{v}^{(j)} \right\rangle$
  $\mathbf{v}^{(j+1)} = A\mathbf{v}^{(j)} - \delta_j \mathbf{v}^{(j)} - \gamma_j \mathbf{v}^{(j-1)}$
  $\gamma_{j+1} = ||\mathbf{v}^{(j+1)}||$
  $\alpha_0 = c_j \delta_j - c_{j-1} s_j \gamma_j$
  $\alpha_1 = \sqrt{\alpha_0^2 + \gamma_{j+1}^2}$
  $\alpha_2 = s_j \delta_j + c_{j-1} c_j \gamma_j$
  $\alpha_3 = s_{j-1} \gamma_j$
  $c_{j+1} = \alpha_0/\alpha_1$; $s_{j+1} = \gamma_{j+1}/\alpha_1$
  $\mathbf{w}^{(j+1)} = (\mathbf{v}^{(j)} - \alpha_3 \mathbf{w}^{(j-1)} - \alpha_2 \mathbf{w}^{(j)})/\alpha_1$
  $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + c_{j+1} \eta \mathbf{w}^{(j+1)}$
  $\eta = -s_{j+1} \eta$
  TEST FOR CONVERGENCE
end for

however, because the matrix may have both positive and negative eigenvalues, the polynomial p_k must be equal to unity at zero and be small at points on either side. Thus, having a set of eigenvalues $[a, b]$ is not, in general, useful until it can be split into an inclusion interval $[-\alpha, -\beta] \cup [\gamma, \delta]$, for $\alpha, \beta, \gamma, \delta > 0$.

Just as we did in the conjugate gradients case, we can write a preconditioned version of MINRES, Algorithm 12. Here, again, at each iteration we need to solve for the preconditioner $\mathcal{P}$, so we must be able to do this cheaply.

$\mathcal{P}$ must be symmetric and positive definite in this case in order to preserve symmetry in the preconditioned system. With the preconditioned method it can be shown (see [39, Section 6.1]) that it is $||\mathbf{r}_k||_{\mathcal{P}^{-1}}$ that is minimized, and the corresponding convergence estimate becomes

$$||\mathbf{r}_k||_{\mathcal{P}^{-1}} \leq \min_{p_k \in \Pi_k, p_k(0)=1} \max_{\lambda} |p_k(z)| \; ||\mathbf{r}_0||_{\mathcal{P}^{-1}}.$$

where the maximum is over the eigenvalues λ of $\mathcal{P}^{-1}A$. Note that a positive definite preconditioner is required in order for $\|\cdot\|_{\mathcal{P}^{-1}}$ to define a norm. As the reduction of the residual in this case is in a norm dependent on the preconditioner, we must be careful not to chose a preconditioner that simply distorts this norm. The aim of preconditioning is again to cluster the eigenvalues, although in this case they can lie

Algorithm 12 Preconditioned MINRES

$\mathbf{v}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(1)} = \mathbf{0}$
Choose $\mathbf{x}^{(0)}$, compute $\mathbf{v}^{(1)} = \mathbf{b} - A\mathbf{x}^{(0)}$
Solve $P\mathbf{z}^{(1)} = \mathbf{v}^{(1)}$, set $\gamma_1 = \sqrt{\langle \mathbf{z}^{(1)}, \mathbf{v}^{(1)} \rangle}$
Set $\eta = \gamma_1$, ,$s_0 = s_1 = 0$, $c_0 = c_1 = 1$
for $j = 1, 2...$**until** convergence **do**
  $\mathbf{z}^{(j)} = \mathbf{z}^{(j)}/\gamma_j$
  $\delta_j = \left\langle A\mathbf{z}^{(j)}, \mathbf{z}^{(j)} \right\rangle$
  $\mathbf{v}^{(j+1)} = A\mathbf{v}^{(j)} - (\delta_j/\gamma_j)\mathbf{v}^{(j)} - (\gamma_j/\gamma_{j-1})\mathbf{v}^{(j-1)}$
  Solve $P\mathbf{z}^{(j+1)} = \mathbf{v}^{(j+1)}$
  $\gamma_{j+1} = \sqrt{\langle \mathbf{z}^{(j+1)}, \mathbf{v}^{(j+1)} \rangle}$
  $\alpha_0 = c_j\delta_j - c_{j-1}s_j\gamma_j$
  $\alpha_1 = \sqrt{\alpha_0^2 + \gamma_{j+1}^2}$
  $\alpha_2 = s_j\delta_j + c_{j-1}c_j\gamma_j$
  $\alpha_3 = s_{j-1}\gamma_j$
  $c_{j+1} = \alpha_0/\alpha_1$; $s_{j+1} = \gamma_{j+1}/\alpha_1$
  $\mathbf{w}^{(j+1)} = (\mathbf{z}^{(j)} - \alpha_3\mathbf{w}^{(j-1)} - \alpha_2\mathbf{w}^{(j)})/\alpha_1$
  $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + c_{j+1}\eta\mathbf{w}^{(j+1)}$
  $\eta = -s_{j+1}\eta$
  TEST FOR CONVERGENCE
end for

either side of the origin. Ideally we would like to get the solution in a number of iterations that is independent of the size of the system.

For a more comprehensive discussion on the properties of MINRES see, for example, Saad [100] or van der Vorst [121].

3.1.8 GMRES

The two Krylov subspace methods we have considered so far are only applicable if we have a symmetric matrix. If we would like to solve a non-symmetric matrix, one alternative could be to consider the normal equations,

$$A^T A\mathbf{x} = A^T\mathbf{b}.$$

This could be solved using, say, conjugate gradients. However, the squaring of the condition number – and consequent slower convergence – that will result here means that this is generally inadvisable unless you use a good preconditioner.

The analogue of the Lanczos process for non-symmetric matrices is the Arnoldi method, which is obtained by carrying out a Gram-Schmidt orthogonalization on the

Krylov subspace $\mathcal{K}_k(A, \mathbf{r}^{(0)})$. It can be written in matrix form [49, Section 10.4.4] as

$$AV_k = V_k H_k + h_{k+1,k}[\mathbf{0}, \ldots, \mathbf{0}, \mathbf{v}^{(k+1)}],$$

where here H_k is a upper Hessenberg matrix and V_k is as defined in the previous section. Note that the only difference between this and (3.29) is the analogue of H_k in the latter, symmetric, case is a tridiagonal matrix.

If we want to minimize $\|\mathbf{r}^{(k)}\|_2$ then, as in the previous section, this is equivalent to finding a vector that is the solution of the linear least squares problem

$$\min \|\|\mathbf{r}^{(0)}\|_2 \mathbf{e}_1 - \widehat{H}_k \mathbf{z}^{(k)}\|_2.$$

As was the case with MINRES, we can solve this system with a QR factorization which requires just one Givens rotation at each iteration. The resulting method is known as the *Generalized Minimal Residual* method (GMRES), and was first described by Saad and Schultz in 1986 [101]. Algorithm 13 gives the algorithm as described above. As usual, this can be combined with a preconditioner in the standard way.

Algorithm 13 GMRES

Choose $\mathbf{x}^{(0)}$, compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$, set $\beta_0 = \|\mathbf{r}^{(0)}\|$, $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/\beta_0$
for $j = 1, 2...$**until** convergence **do**
  $\mathbf{w}_0^{(k+1)} = A\mathbf{v}^{(k)}$
  for $l = 1$ **to** l **do**
    $h_{l,k} = \left\langle \mathbf{w}^{(k+1)}, \mathbf{v}^{(l)} \right\rangle$
    $\mathbf{w}_{l+1}^{(k+1)} = \mathbf{w}_l^{(k+1)} - h_{l,k}\mathbf{v}^{(l)}$
  end for
  $h_{k+1,k} = \|\mathbf{w}_{k+1}^{(k+1)}\|$
  $\mathbf{v}^{(k+1)} = \|\mathbf{w}_{k+1}^{(k+1)}\|/h_{k+1,k}$
  Compute $\mathbf{y}^{(k)}$ such that $\beta_k = \|\beta_0 \mathbf{e}_1 - \widehat{H}_k \mathbf{y}^{(k)}\| \to \min!$
  TEST FOR CONVERGENCE
end for
$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + V_k \mathbf{y}^{(k)}$

The drawback with GMRES compared to MINRES or CG is that it relies on a Hessenberg matrix. To generate $\mathbf{v}^{(k+1)}$ we must use – and hence store – every subsequent vector $\mathbf{v}^{(j)}$, $j = 1, \ldots, k$. This can be prohibitively expensive if many iterations are required[3]. For this reason the method is often restarted after m iterations in practice, where m is a fixed number. The last iteration $\mathbf{x}^{(m)}$ then becomes $\mathbf{x}^{(0)}$ for the

[3]Recall CG and MINRES are both based on three-term recurrences, so only the last two vectors need to be stored to generate the next one.

next cycle, and the process continues until convergence. Such a scheme is known as GMRES(m), or restarted GMRES.

A convergence analysis for GMRES is more difficult than in the case of CG and MINRES – mainly due to the presence of complex eigenvalues. It can be shown [39, Theorem 4.1] that if A is a diagonalizable matrix, the residual satisfies the following 'minimax' bound

$$\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{r}^{(0)}\|} \leq \kappa(V) \min_{p_k \in \Pi_k, p_k(0)=1} \max_{\lambda_j} |p_k(\lambda)|, \tag{3.31}$$

where $A = V\Lambda V^{-1}$ is the eigendecomposition of A, λ_j are the eigenvalues and $\kappa(V)$ is the condition number of the eigenvector matrix, V. The presence of the term $\kappa(V)$ in (3.31) tells us that convergence of GMRES is not just dependent on the eigenvalues of the matrix, as was the case with CG and MINRES. In fact, Greenbaum, Pták and Strakoš [54] showed that for any non-increasing convergence curve, and any desired set of eigenvalues, there exists a matrix with these eigenvalues and an associated right hand side vector such that the convergence of GMRES when applied to this system is described by the given curve.

For more discussion on the convergence of GMRES see, for example, Greenbaum [53, Section 3.2] or Embree [40] and the references therein. It is worth noting that for many matrices which arise in applications, and which are not constructed to be pathological examples, good convergence is achieved if the matrix has eigenvalues clustered away from the origin.

3.2 Multigrid

In the preceding sections we have described general methods for solving linear systems of equations. We now want to consider a case where the matrix has some structure we can exploit – namely the situation where the system we have to solve comes from a partial differential equation. In particular, we will consider solving Laplace's equation with Dirichlet boundary conditions, namely

$$-\nabla^2 u = f \ \text{ in } \Omega, \ u = g \text{ on } \partial\Omega,$$

where $\Omega = [0,1]^d$, $d = 2, 3$, and for appropriate functions f and g. As described in Section 2.1, the finite element method gives us a way to obtain an approximate solution to this equation. Suppose that we discretize the system using $\mathbf{Q}_1$ finite elements with a uniform mesh size h. Then the finite element approximation is obtained by solving

$$K\mathbf{u} = \mathbf{f}, \tag{3.32}$$

where $K \in \mathbb{R}^{m \times m}$ is the stiffness matrix and $\mathbf{f} \in \mathbb{R}^m$ is an appropriate vector. Here, assuming we apply the boundary conditions after solving the system, $m = (N-1)^d \times (N-1)^d$, where $N = 1/h$.

The main difficulty with solving a system of the form (3.32) is the size of the system. In Section 2.1 we saw that the smaller we choose the mesh size, h, the more accurate the solution is. If you half the mesh size, the dimension of the matrix K essentially increases by a factor of 2^d, and so the system can get very large.

We will briefly discuss how direct methods cope with such systems. MATLAB's `backslash` is such a command. In this case, since we have a banded sparse matrix, `backslash` will call a Gaussian elimination routine with partial pivoting – namely LAPACK commands `DGBTRF`, which performs an LU factorization, followed by `DGBTRS`, which solves it. See `doc mldivide` in MATLAB for more details.

Table 3.1 shows the time taken to solve (3.32) using `backslash` for different values of h in both two and three dimensions. Table 3.1(a) shows that in two dimensions the direct method performs rather well – although it doesn't quite scale linearly. However, it is clear from Table 3.1(b) that in three dimensions the direct solver starts to struggle.

Table 3.1: CPU times to solve (3.32) using `backslash` for different mesh sizes, along with the ratio between current and previous time

(a) Two dimensions

h	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}
time (s)	0.0001	0.0002	0.0006	0.0031	0.016	0.083	0.48	2.8	17.6
t_n/t_{n-1}	—	1.43	3.76	4.76	5.32	5.13	5.82	5.75	6.28

(b) Three dimensions

h	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
time (s)	0.0005	0.0026	0.1124	11.6929	—[a]
t_n/t_{n-1}	—	5.20	42.5	104.0	—[a]

[a] Ran out of memory

This behaviour is seen because of the significant *fill-in* of zero entries with non-zeros during the Gaussian Elimination. This leads to a large growth in storage requirements, and hence the computer runs out of memory. Iterative methods do not suffer from this problem, so we would therefore like to find iterative methods with which to solve equation (3.32). Problems of this form clearly have lots of structure, and it is this structure that is exploited to give us a fast solution method.

First let us consider what happens if we apply a simple iteration. Consider the Jacobi iteration:

$$D\mathbf{x}^{(k+1)} = (L+U)\mathbf{x}^{(k)} + \mathbf{b},$$

where $D = \texttt{diag}(K)$, and L, U are the strictly lower and upper triangular parts of K respectively. We get the rate of convergence by looking at the spectral radius of $I - D^{-1}K$. If K is discretized using $\mathbf{Q}_1$ elements, then it can be shown ([39, Chapter 2]) that the eigenvalues satisfy

$$\lambda^{(r,s)} = \frac{1}{4}\left(\cos\frac{r\pi}{N} + \cos\frac{s\pi}{N}\right) + \frac{1}{2}\cos\frac{r\pi}{N}\cos\frac{s\pi}{N}, \quad r, s = 1, \ldots, N-1.$$

It is clear that $|\lambda^{(r,s)}| < 1$, so this converges, but how quickly? Recall from (3.5) that the error at the k^{th} iteration satisfies

$$\mathbf{e}^{(k)} = (I - M^{-1}K)^k \mathbf{e}^{(0)}.$$

If we relabel the $\lambda^{(r,s)}$ as λ_i, $i = 1, \ldots, m$ and write the corresponding eigenvector as $\mathbf{v}_i$, then we can expand the initial error in terms of the eigenvectors

$$\mathbf{e}^{(0)} = \sum_i c_i \mathbf{v}_i,$$

and hence

$$\mathbf{e}^{(k)} = \sum_i c_i \lambda_i^k \mathbf{v}_i.$$

This tells us that the eigenvector components of the initial error corresponding to eigenvalues near to zero are reduced very rapidly by Jacobi iteration. On the other hand, the components of the eigenvectors corresponding to the larger eigenvalues (i.e. r and s both close to zero) will take many iterations to be reduced.

Another way to look at this is that the high frequency components of the error will get small very quickly, whereas the low frequency, or smooth, components will take a lot longer to converge. What we will be left with after a few steps is an error that is not necessarily close to zero in norm, but which will look much smoother. This observation leads us to the idea of *multigrid*.

Since a small number of steps of a suitable simple iteration have the effect of 'smoothing' the vector they are applied to, we might consider this smoothed problem on a coarser grid. Thus if we take every other mesh point, say, and define some operator to interpolate between the two grids, then solving on the new, coarser, grid will be easier, since we have significantly reduced the problem size. If we apply this

method recursively and stop when we get to a system small enough that it can be easily dealt with by a direct method, then we have the multigrid algorithm.

Multigrid has been developed from its beginnings – some theoretical results in the mid 60s – to the first practical results by Brandt [22] and the independent discovery and subsequent development of the mathematical foundations by Hackbusch in the mid seventies (see [59] and the references therein). Today it is widely used as a fast solution technique for many different problems.

We would like to make the description above more formal – first, we consider a 'two-grid' algorithm. Our description follows [39, Section 2.5]. Suppose K is discretized with respect to a basis S^h, which we call the *fine grid space*. For $\mathbf{Q}_1$ approximation this is the set of bi- or tri-linear functions defined on some grid of squares or bricks, each point being a distance h from its nearest neighbour. Then consider the equivalent *coarse grid space* S^{2h} of a grid of width $2h$ (see Figure 3.3). We want to write the grid function, which lives in S^h, in terms of this coarser basis S^{2h} and the *fine grid correction space*, B^h, which is the span of the nodes that are not nodes of the coarse grid (so $S^h = S^{2h} + B^h$). Figure 3.4 illustrates this in one dimension.

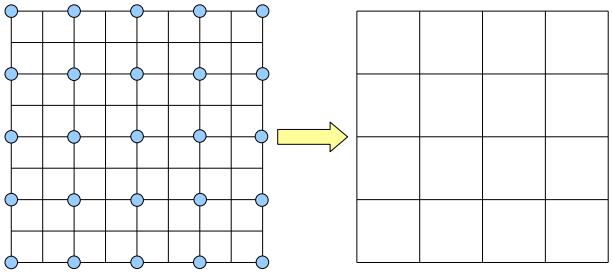


Figure 3.3: Sequence of grids used by the two-grid algorithm.

The first things we need for our algorithm are a mapping from the coarse grid space to the fine grid space, which we call a *prolongation* operator, and a *restriction* operator, which maps from S^{2h} to S^h. If we call these operators I_h^{2h} and I_{2h}^h respectively, then we have

$$I_h^{2h} : S^h \to S^{2h}, \quad I_{2h}^h : S^{2h} \to S^h.$$

Let us first consider the prolongation operator. Let ϕ_i^{2h} be an element in the coarse grid space, $S^{2h} := \text{span}\left\{\phi_1^{2h}, \ldots, \phi_{n_{2h}}^{2h}\right\}$, and let $S^h := \text{span}\left\{\phi_1^h, \ldots, \phi_{n_h}^h\right\}$. As $S^{2h} \subset$

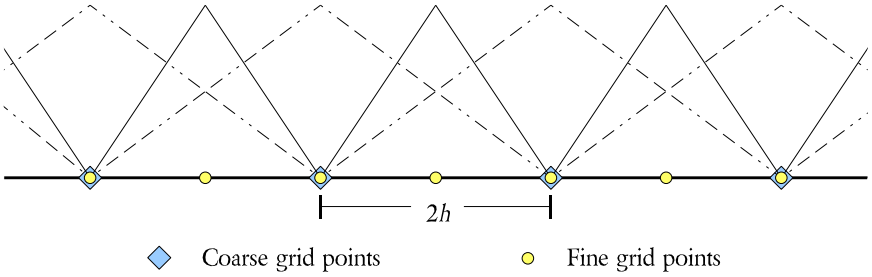


Figure 3.4: 1D representation of the coarse grid space, S^{2h}, basis (dashed line) and the fine grid correction basis, B^h

S^h,

$$\phi_i^{2h} = \sum_{j=1}^{n_h} p_{i,j}\phi_j^h,$$

for some coefficients $p_{i,j}$. Now if $x_{2h} \in S^{2h}$, we can write

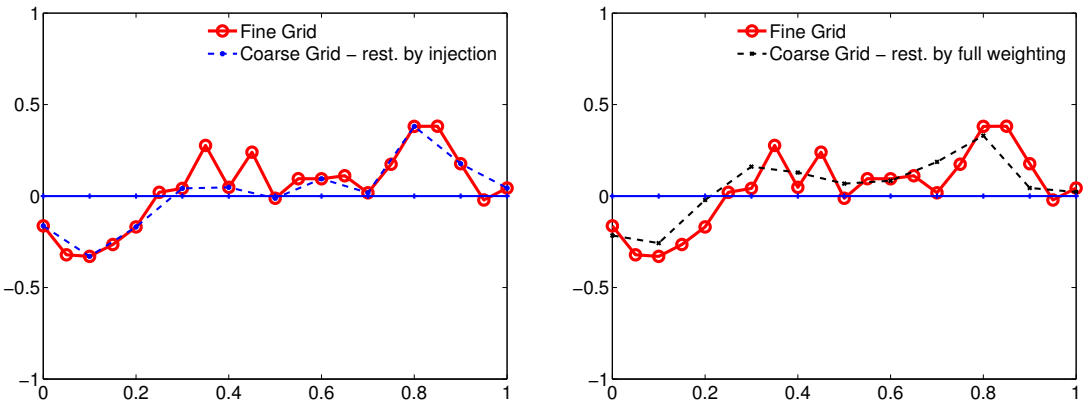
$$x_{2h} = \sum_{i=1}^{n_{2h}} \mathbf{x}_i^{2h}\phi_i^{2h} = \sum_{i=1}^{n_{2h}} \mathbf{x}_i^{2h} \sum_{j=1}^{n_h} p_{i,j}\phi_j^h = \sum_{j=1}^{n_h} [P\mathbf{x}^{2h}]_j \phi_j^h.$$

We can therefore write the prolongation operator in terms of an $n_h \times n_{2h}$ matrix P, which is defined by the coefficients of this linear combination. That is, if the vector $\mathbf{x}^{2h}$ represents the function x_{2h} in S^{2h}, then $\mathbf{x}^h = P\mathbf{x}^{2h}$ is the vector that represents x_{2h} in S^h. This is called projection by interpolation.

Now we have a mapping that takes us from the coarse grid to the fine grid, but we still need the reverse map. Whereas there was an obvious choice of projection operator, there are a number of possible options for a restriction operator. Two of the most intuitive are injection, where you ignore the components of the basis functions at the points you discard, or full weighting, where you take an average of the neighbouring points. These two options are demonstrated in one dimension in Figure 3.5. As we see from the diagrams, if the vector is at all oscillatory, information is lost by taking injection as the restriction method (Figure 3.5(a)), whereas restriction by full weighting (Figure 3.5(b)) includes data from all nodes.

Let $x_h = \sum_{i=1}^{n_h} \mathbf{x}_i^h \phi_i^h \in S^h$. Then, as above, we want to find the matrix R such that $\mathbf{x}^{2h} = R\mathbf{x}^h$, where $\sum_{j=1}^{n_{2h}} \mathbf{x}_j^{2h}\phi_j^{2h} = x_{2h} = I_h^{2h}(x_h)$. In this case, if we use restriction by full weighting this matrix R is given in terms of the prolongation matrix P described above by the simple relation

$$R = P^T.$$



(a) Restriction by injection (b) Restriction by full weighting

Figure 3.5: Schematic diagram of two methods of restriction in 1D

The central idea of multigrid is that we want to use our simple iteration – which is usually called a *smoother* in the multigrid literature – to reduce the fine grid component of the error. For example, if we take the initial error $\mathbf{e}^{(0)} = \mathbf{u} - \mathbf{u}^{(0)}$, then we can use the fact that $S^h = S^{2h} + B^h$ to write this as

$$\mathbf{e}^{(0)} = P\mathbf{e}^{(0)}_{S^{2h}} + \mathbf{e}^{(0)}_{B^h},$$

where $\mathbf{e}_{S^{2h}}$ and $\mathbf{e}_{B^h}$ are the vectors of coefficients with respect to the bases of S^{2h} and B^h respectively. If we then apply, say, k steps of relaxed Jacobi as our smoother, then the error will satisfy

$$\mathbf{e}^{(k)} = (I - M^{-1}K)^k\mathbf{e}^{(0)} = P\mathbf{e}^{(k)}_{S^{2h}} + \mathbf{e}^{(k)}_{B^h}.$$

Since the application of the simple iteration rapidly reduces the high frequency component of a vector, we should have that

$$\mathbf{e}^{(k)}_{B^h} \approx \mathbf{0},$$

and so the fine grid error vector can be restricted to the coarse grid with minimal loss of information.

The smoothing, although explained above in terms of the error, is actually done on the residual. Note that

$$\mathbf{r}^{(k)} = K\mathbf{e}^{(k)} = K(I - M^{-1}K)^k\mathbf{e}^{(0)} = (I - KM^{-1})^k\mathbf{r}^{(0)},$$

and so we can restrict the smoothed residual, giving

$$\bar{\mathbf{r}} := R\mathbf{r}^{(k)} = P^T(I - KM^{-1})^k\mathbf{r}^{(0)}.$$

Now,

$$\bar{\mathbf{r}} = P^T \mathbf{r}^{(k)} = P^T K \mathbf{e}^{(k)}.$$

If the smoothing has been effective, we have that $\mathbf{e}^{(k)} \approx P\bar{\mathbf{e}}$, where $\bar{\mathbf{e}}$ is the error on the coarse grid. Therefore, if we solve the system

$$P^T K P \bar{\mathbf{e}} = \bar{\mathbf{r}},$$

we will have an estimate for the error in the basis of the coarse grid. Note that the matrix $P^T K P$ is much smaller – about 2^d times smaller in d dimensions – than the original matrix, and so is correspondingly easier to solve. The vector $\bar{\mathbf{e}}$ is called the *coarse grid correction*. Prolonging $\bar{\mathbf{e}}$ back into the fine grid and updating gives a better approximation to the solution

$$\mathbf{u}^{(1)} = \mathbf{u}^{(0)} + P\bar{\mathbf{e}}.$$

We can do further iterations like this until convergence, using the previous $\mathbf{u}^{(k)}$ as the next starting vector. This gives us a simple version of the two-grid algorithm. Putting the components described above together, we get a basic two-grid algorithm, given as Algorithm 14.

We call $\bar{K} := P^T K P$ a *coarse grid operator*. Consider an entry of the stiffness matrix on the grid of size $2h$, $\bar{k}_{i,j}$. Then,

$$\begin{aligned}\bar{k}_{i,j} = \int_\Omega \nabla\phi_i^{2h} \cdot \nabla\phi_j^{2h} &= \int_\Omega \left(\sum_l p_{i,l} \nabla\phi_i^h\right) \cdot \left(\sum_m p_{j,m} \nabla\phi_j^h\right) \\ &= \sum_l p_{i,l} \sum_m p_{j,m} \int_\Omega \nabla\phi_i^h \cdot \nabla\phi_j^h,\end{aligned}$$

which shows us that for the restriction and projection operators as defined above, the stiffness matrix on the coarse grid is precisely our coarse grid operator, $P^T K P$. In other words, $P^T K^h P = K^{2h}$, where the superscript denotes the size of the grid on which the matrix is discretized. This is not always the case – we call the operator $\bar{K}$ the *Galerkin* coarse grid operator. In other situations[4] it may be better to use the coarse grid discretization, K^{2h}, as our coarse grid operator.

To analyze convergence, we look at the error in this method. It is straightforward to show that the error in Algorithm 14 satisfies

$$\mathbf{e}^{(i+1)} = (K^{-1} - P\bar{K}^{-1}P^T)K(I - M^{-1}K)^k \mathbf{e}^{(i)}. \tag{3.33}$$

[4]for example, when solving the advection-diffusion equation – see Chapter 6.

It can be shown (see Hackbusch [58] for details) that the two-grid algorithm applied to Laplace's equation in 2D reduces the error by a factor independent of h if it satisfies two conditions; the *smoothing* property,

$$\|K(I - M^{-1}K)^k \mathbf{y}\| \leq \eta(k)\|\mathbf{y}\|_K \text{ with } \eta(k) \to 0 \text{ as } k \to \infty,$$

and the *approximation* property,

$$\|(K^{-1} - P\bar{K}^{-1}P^T)\mathbf{y}\|_K \leq C\|\mathbf{y}\|,$$

for all vectors $\mathbf{y}$, where η and C are independent of the grid size, h. Similar conditions, which split the convergence into the quality of the smoother and the quality of our coarse grid operator and grid transfer operations, can be shown for other problems.

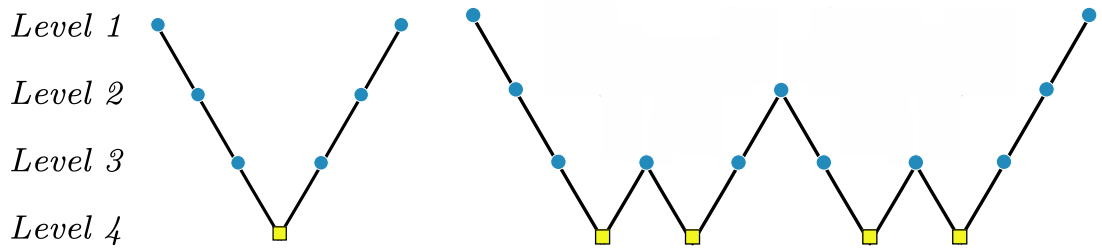
Algorithm 14 2-Grid algorithm to solve $K\mathbf{u} = \mathbf{f}$, for some splitting $K = M - N$, with m smoothing steps

Choose $\mathbf{u}^{(0)}$
for $i = 1$ **until** convergence **do**
  for $k = 1...m$ **do**
    $\mathbf{u}^{(i)} = (I - M^{-1}K)\mathbf{u}^{(i)} + M^{-1}\mathbf{f}$
  end for
  $\bar{\mathbf{r}} = P^T(\mathbf{f} - K\mathbf{u}^{(i)})$
  $\bar{K}\bar{\mathbf{e}} = \bar{\mathbf{r}}$
  $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + P\bar{\mathbf{e}}$
end for

Of course, as we have alluded to above, we don't have to stop at a two-grid method. When Algorithm 14 calls for a solve on the coarse grid ($\bar{K}\bar{\mathbf{e}} = \bar{\mathbf{r}}$), we can simply recursively call the algorithm again, stopping when we get to a coarse grid matrix small enough that a direct method, say, can safely solve the system.

We still have to pick the way in which we go from grid to grid. Two of the most common methods are the V-cycle and the W-cycle – schematic diagrams illustrating these two cycling methods are given in Figure 3.6. In the diagram a square represents an exact solve and a circle represents grid transfer (and smoothing). A W-cycle is clearly more expensive than a V-cycle, but it is subsequently easier to prove convergence properties. Algorithm 15 gives the pseudocode for a single V-cycle.

To solve the system using multigrid we just repeat the V- or W-cycle until the required tolerance is achieved. As long as the smoothing and approximation properties hold the number of cycles required for convergence to a fixed tolerance is independent of h. Therefore, in the sense described in the start of the chapter, multigrid is an optimal solver.



(a) V-cycle (b) W-cycle

Figure 3.6: Multigrid V-cycle and W-cycle

Algorithm 15 Multigrid V-Cycle to solve $K\mathbf{u} = \mathbf{f}$ with a smoothing splitting $K = M - N$

function $\mathbf{u} = \texttt{VCycle}(K, \mathbf{f}, \mathbf{u}, \text{level})$
for $k = 1...m$ **do**
  $\mathbf{u} = (I - M^{-1}K)\mathbf{u} + M^{-1}\mathbf{f}$
end for
if coarsest level **then**
  solve $K\mathbf{u} = \mathbf{f}$
else
  $\bar{\mathbf{r}} = P^T(\mathbf{f} - K\mathbf{u})$
  $\bar{\mathbf{e}} = \texttt{VCycle}(\bar{K}, \bar{\mathbf{r}}, \bar{\mathbf{e}}, \text{level} + 1)$
  $\mathbf{u} = \mathbf{u} + P\bar{\mathbf{e}}$
end if

Table 3.2 shows the time taken to solve the system (3.32) using Algorithm 15 to a relative residual of 10^{-6}, where we've used two steps of relaxed Jacobi as a smoother. Comparing the results here with those in Table 3.1 we see that `backslash` is more efficient for the smaller problems, but as h decreases, multigrid gets to be the faster method. In three dimensions the situtation even is more clear – multigrid performs a lot better than `backslash` here.

Table 3.2: CPU times to solve (3.32) using Multigrid for different mesh sizes, along with the ratio between current and previous time

(a) Two dimensions

h	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}
time (s)	0.0019	0.0019	0.0035	0.010	0.037	0.14	0.65	2.84	11.5
iterations	4	5	5	6	6	6	6	6	6
t_n/t_{n-1}	—	1.01	1.84	3.07	3.45	3.95	4.51	4.32	4.03

(b) Three dimensions

h	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
time (s)	0.0053	0.0054	0.0402	0.5545	5.1720
iterations	3	4	4	5	5
t_n/t_{n-1}	—	1.02	7.45	13.7	9.32

For a more detailed discussion about multigrid techniques, see for example Briggs, Henson and McCormick [26], Wesseling [128], or Trottenberg, Oosterlee and Schüller [119].

The description in this section has been of 'geometric' multigrid, but there is also the possibility of 'algebraic' multigrid (AMG). This uses algebraic connections in the matrix to define coarse grid operators, and so requires no knowledge of the geometric information of the problem. AMG can therefore be treated as more of a 'black box' algorithm. The latest HSL release includes an AMG code – MI20, for which an interface with MATLAB has been written by Dollar [15]. For a discussion of algebraic multigrid, see e.g. Brandt, McCormick and Ruge [23].

Notice that the equation for the error in the two-grid algorithm, (3.33), has the same form as the error in a simple iteration (3.5). This is because multigrid as described above is actually a simple iteration, with splitting matrix M_{MG} given by

$$M_{\text{MG}}^{-1} = K^{-1} - (K^{-1} + P\bar{K}^{-1}P^T)K(I - M^{-1}K)^k K^{-1}.$$

It can be shown [84, Chapter 9] that this is indeed positive definite. All the results relating to the simple iteration in Section 3.1.1 are therefore true here; in particular, a fixed number of multigrid cycles is a linear operator. This leads us to ask

whether we can improve the convergence of CG or MINRES by using multigrid as a preconditioner?

If we use the algorithm as given in Algorithm 15, then the answer is no – for a linear operator to be used as a preconditioner for these two methods, it must be symmetric, and it is easy to see that this is not the case here. However, this problem can be easily circumvented with the addition of post-smoothing – i.e. after the correction step we run some further iterations with another smoother. It is easy to see that if the (pre-)smoothing step consisted of k steps with the iteration matrix $I - M^{-1}K$, then if we apply a further k steps of post-smoothing with the iteration matrix $I - M^{-T}K$ – where we take the transpose of the original splitting matrix – will lead to a symmetric method with the splitting matrix

$$M_{\text{MG}}^{-1} = K^{-1} - (I - M^{-T}K)^k(K^{-1} + P\bar{K}^{-1}P^T)K(I - M^{-1}K)^k K^{-1}.$$

This version, which is suitable as a preconditioner, is given as Algorithm 16.

Algorithm 16 Multigrid V-Cycle suitable as a preconditioner for $K\mathbf{u} = \mathbf{f}$

function $\mathbf{u} = \texttt{VCycle}(K, \mathbf{f}, \mathbf{u}, \text{level})$
for $k = 1...m$ **do**
  $\mathbf{u} = (I - M^{-1}K)\mathbf{u} + M^{-1}\mathbf{f}$
end for
if coarsest level **then**
  solve $K\mathbf{u} = \mathbf{f}$
else
  $\bar{\mathbf{r}} = P^T(\mathbf{f} - K\mathbf{u})$
  $\bar{\mathbf{e}} = \texttt{VCycle}(\bar{K}, \bar{\mathbf{r}}, \bar{\mathbf{e}}, \text{level} + 1)$
  $\mathbf{u} = \mathbf{u} + P\bar{\mathbf{e}}$
end if
for $k = 1...m$ **do**
  $\mathbf{u} = (I - M^{-T}K)\mathbf{u} + M^{-T}\mathbf{f}$
end for

Applying just one V-cycle of multigrid is in many instances a very good preconditioner, and allows us to tap into the non-linear qualities of CG to get very rapid convergence.

We can make this statement more precise. From the approximation and smoothing properties above, it was shown by Braess and Hackbusch [17] that there exists a constant ρ independent of h such that

$$\|\mathbf{u} - \mathbf{u}^{(i+1)}\|_K \leq \rho\|\mathbf{u} - \mathbf{u}^{(i)}\|_K. \tag{3.34}$$

Since multigrid is a simple iteration, we can write

$$\mathbf{u}^{(i+1)} = (I - \widetilde{K}^{-1}K)\mathbf{u}^{(i)} + \widetilde{K}^{-1}\mathbf{f},$$

where $\widetilde{K}$ is the same as M_{MG} in the notation above. We can therefore write (3.34) as

$$\left\langle K(I - \widetilde{K}^{-1}K)\mathbf{e}^{(i)}, (I - \widetilde{K}^{-1}K)\mathbf{e}^{(i)} \right\rangle \leq \rho^2 \left\langle K\mathbf{e}^{(i)}, \mathbf{e}^{(i)} \right\rangle,$$

where, as usual, $\mathbf{e}^{(i)} = \mathbf{u} - \mathbf{u}^{(i)}$. This must hold regardless of the vector $\mathbf{u}^{(i)}$, so we can write $\mathbf{v} = K^{\frac{1}{2}}\mathbf{e}^{(i)}$ (since K is symmetric positive definite) to get

$$\left\langle (I - K^{\frac{1}{2}}\widetilde{K}^{-1}K^{\frac{1}{2}})\mathbf{v}, (I - K^{\frac{1}{2}}\widetilde{K}^{-1}K^{\frac{1}{2}})\mathbf{v} \right\rangle \leq \rho^2 \left\langle K\mathbf{e}^{(i)}, \mathbf{e}^{(i)} \right\rangle \quad \forall\ \mathbf{v}.$$

Therefore

$$-\rho \left\langle \mathbf{v}, \mathbf{v} \right\rangle \leq \left\langle (I - K^{\frac{1}{2}}\widetilde{K}^{-1}K^{\frac{1}{2}})\mathbf{v}, \mathbf{v} \right\rangle \leq \rho \left\langle \mathbf{v}, \mathbf{v} \right\rangle \quad \forall\ \mathbf{v},$$

and hence we have

$$1 - \rho \leq \frac{\mathbf{v}^T K \mathbf{v}}{\mathbf{v}^T \widetilde{K} \mathbf{v}} \leq 1 + \rho \quad \forall\ \mathbf{v} \neq 0, \tag{3.35}$$

which is Lemma 6.2 in [39]. This results tells us that the eigenvalues of $\widetilde{K}^{-1}K$ satisfy

$$\lambda(\widetilde{K}^{-1}K) \in [1 - \rho, 1 + \rho],$$

and so $\kappa(\widetilde{K}^{-1}K) = \frac{1+\rho}{1-\rho}$. A typical contraction rate per V-cycle applied to the Dirichlet problem would be $\rho \approx 0.15$. This would give $\kappa \approx 1.35$, and so the error bound (3.27) to solve the system with CG and 1 V-cycle as a preconditioner would be

$$\frac{\|\mathbf{e}_{\text{CG}}^{(k)}\|_K}{\|\mathbf{e}_{\text{CG}}^{(0)}\|_K} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k = 2(0.075)^k,$$

so a conjugate gradient acceleration significantly speeds up convergence.

Table 3.3 gives results in two and three dimensions for the solution of equation (3.32) with preconditioned conjugate gradients, with one V-cycle with two pre- and two post-smoothing steps of relaxed Jacobi as a preconditioner. We also, for ease of comparison, reproduce the results given in Tables 3.1 and 3.2 here.

3.3 Comments

In this chapter we have presented a variety of methods for solving linear systems of equations. The methods we have presented can be split into two groups – linear and non-linear methods.

Table 3.3: CPU times to solve (3.32) using PCG preconditioned using one V-cycle for different mesh sizes, along with the ratio between current and previous time. Also included are the equivalent results for solution with `backslash` and Multigrid

(a) Two dimensions

	PCG			`backslash`		Multigrid		
h	time (s)	iter.	t_n/t_{n-1}	time	t_n/t_{n-1}	time	iter.	t_n/t_{n-1}
2^{-2}	0.002	5	—	0.0001	—	0.002	4	—
2^{-3}	0.002	6	1.04	0.0002	1.43	0.002	5	1.01
2^{-4}	0.003	5	1.46	0.0006	3.76	0.003	5	1.84
2^{-5}	0.008	5	2.65	0.003	4.76	0.010	6	3.07
2^{-6}	0.030	5	3.52	0.016	5.32	0.037	6	3.45
2^{-7}	0.12	5	4.06	0.083	5.13	0.14	6	3.95
2^{-8}	0.55	5	4.4	0.48	5.82	0.65	6	4.51
2^{-9}	2.32	5	4.20	2.81	5.75	2.84	6	4.32
2^{-10}	9.57	5	4.12	17.6	6.28	11.5	6	4.03

(b) Three dimensions

	PCG			`backslash`		Multigrid		
h	time (s)	iter.	t_n/t_{n-1}	time	t_n/t_{n-1}	time	iter.	t_n/t_{n-1}
2^{-2}	0.0095	2	—	0.0005	—	0.0053	3	—
2^{-3}	0.0136	3	1.43	0.0026	5.20	0.0054	4	1.02
2^{-4}	0.0405	4	2.96	0.1124	42.5	0.0402	4	7.45
2^{-5}	0.4460	4	11.02	11.6929	104.0	0.5545	5	13.7
2^{-6}	4.1829	4	9.37	—[a]	—[a]	5.1720	5	9.32

[a] Ran out of memory

The basic linear method is the simple iteration. This method encompasses both fairly naïve approaches – Jacobi and Gauss-Seidel iterations, for example – and also very sophisticated approaches, such as multigrid. We have discussed two ways to accelerate convergence of the simple iteration – by relaxation and by the Chebyshev semi-iteration. Both of these methods (to varying degrees) require knowledge of the extremal eigenvalues.

The non-linear methods, on the other hand, require no spectral information to be effective. We have seen that these are not much more expensive than the corresponding simple iteration, yet the algorithms automatically pick the optimal parameters at each iteration. For this reason these methods often display superlinear convergence.

The most effective non-linear methods we considered were the Krylov subspace methods conjugate gradients, MINRES and GMRES. These need to be coupled with a preconditioner – which is equivalent to the splitting matrix in the simple iterations – to be effective. A preconditioner is a matrix – or more generally, a linear opera-

tion that implicitly defines a matrix – that accelerates convergence by clustering the eigenvalues. As we need to solve with the preconditioner at each iteration this should be chosen such that this is computationally inexpensive. Preconditioned Krylov subspace methods are generally considered to be the iterative methods of choice.

The non-linear nature of the Krylov subspace methods means that you cannot use CG as a preconditioner for MINRES, for example. However, it is often advantageous to use one of the linear methods as a preconditioner. An example of this was when we preconditioned CG with multigrid, and we'll use this technique in a different situation in Section 5.2.

If we did want to apply a non-linear method as a preconditioner, then we would essentially be applying a different preconditioner at each step. 'Flexible' methods have been developed that can deal with this – for example, FGMRES [102]. For more information on such methods, see, for example, Simoncini and Szyld [107] and the references therein. The theoretical understanding of the convergence of such methods is less complete than that of CG or MINRES, so we will not consider them here.

We have focused on Krylov subspace methods that satisfy a minimality property, although there are many other variants available. For example, an alternative to MINRES for symmetric linear systems is SYMMLQ [91], and for non-symmetric systems one could use QMR [45], BI-CGSTAB [120], or CGS [108], to name but a few. For an overview of these methods, see e.g. Barret et al. [7, Chapter 2] or Meurant [84].

Chapter 4

The Numerical Solution of Saddle Point Systems

As we saw in Chapter 2, the system we are interested in solving has the form

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{4.1}$$

where is $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{m \times n}$, $m < n$. We will sometimes write this equation in the shortened form

$$\mathcal{A}\mathbf{x} = \mathbf{b}. \tag{4.2}$$

Matrices of this type are called *saddle point* systems, since the problem of finding first-order optimality conditions of an equality constrained quadratic programming problem is of this form. Note that – provided A is non-singular – we can decompose $\mathcal{A}$ into

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & -BA^{-1}B^T \end{bmatrix} \begin{bmatrix} I & A^{-1}B^T \\ 0 & I \end{bmatrix}.$$

If we assume that A is positive definite, then $BA^{-1}B^T$ is also positive definite, so the matrix $\mathcal{A}$ will be indefinite. In this case it is clear that $\mathcal{A}$ is invertible if and only if B^T has full column rank. In the case of A being indefinite, the saddle point system will be invertible if A is positive definite on $\ker(B)$ – see, for example, [9, Theorem 3.2] for a proof.

In this chapter we look at some of the methods introduced in Chapter 3, and see how we can apply them to matrices with this specific structure. For a more comprehensive overview, see the survey article by Benzi, Golub and Liesen [9].

4.1 Simple iterations and Inexact Uzawa

Suppose we try to solve (4.1) using a simple iteration, as in Section 3.1.1. Then we have to find a splitting matrix $\mathcal{M}$ such that $\rho(I - \mathcal{M}^{-1}\mathcal{A}) < 1$, and such that M is easy to invert. We assume in the following that the (1,1) block A is symmetric positive definite. Since the matrix $\mathcal{A}$ has a block structure, it makes sense for the matrix $\mathcal{M}$ to also have a block structure. The easiest block matrix to invert is a block diagonal matrix. Suppose we look for a splitting matrix of the form

$$\mathcal{M} = \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix},$$

i.e. we leave the (1,1) block unchanged, and have a matrix to be determined in the (2,2) block. It is well known (see, for example, [87]) that if we take the (2,2) block to be the *Schur complement*, i.e. $S = BA^{-1}B^T$, then the eigenvalues of the generalized eigenvalue problem

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} = \lambda \begin{bmatrix} A & 0 \\ 0 & BA^{-1}B^T \end{bmatrix}$$

are $\lambda = 1$ and $\lambda = \frac{1 \pm \sqrt{5}}{2}$ [87]. Therefore, this choice of $\mathcal{M}$ will not give us $\rho(I - \mathcal{M}^{-1}\mathcal{K}) < 1$, but it does give us something to work with. Suppose, in a similar manner to Section 3.1.2, we introduce a parameter σ, so

$$\mathcal{M} = \begin{bmatrix} A & 0 \\ 0 & \sigma S \end{bmatrix},$$

where $S = BA^{-1}B^T$. Can we find the value of σ that makes the spectral radius as small as possible?

Consider the generalized eigenvalue problem

$$A\mathbf{u} + B^T\mathbf{p} = \lambda A\mathbf{u} \tag{4.3}$$

$$B\mathbf{u} = \sigma\lambda S\mathbf{p}. \tag{4.4}$$

If $B\mathbf{u} = 0$, then $\mathbf{p} = 0$ (since $\sigma \neq 0$) and hence we have repeated eigenvalues at $\lambda = 1$. If this is not the case, then $\lambda \neq 1$, so from (4.3) we can write

$$(1 - \lambda)A\mathbf{u} + B^T\mathbf{p} = \mathbf{0},$$

and rearranging this and multiplying on the left by B we get

$$B\mathbf{u} = \frac{1}{\lambda - 1} S\mathbf{p}.$$

Subsituting this into (4.4),

$$\frac{1}{\lambda - 1} S\mathbf{p} = \sigma\lambda S\mathbf{p},$$

so we must have

$$\left(\frac{1}{\lambda - 1} - \sigma\lambda \right) S\mathbf{p} = \mathbf{0}.$$

As $S\mathbf{p} \neq \mathbf{0}$ (since S is invertible, and $\mathbf{p} \neq \mathbf{0}$), we have

$$\frac{1}{\lambda - 1} - \sigma\lambda = 0,$$

hence

$$\lambda^2 - \lambda - \frac{1}{\sigma} = 0.$$

Therefore the remaining two eigenvalues are given by

$$\lambda = \frac{1 \pm \sqrt{1 + 4/\sigma}}{2}.$$

The eigenvalues of the iteration matrix $I - M^{-1}\mathcal{K}$ are

$$0, \frac{1 + \sqrt{1 + 4/\sigma}}{2}, \text{ and } \frac{1 - \sqrt{1 + 4/\sigma}}{2}.$$

The choice of σ that minimizes the spectral radius is therefore clearly given by $\sigma = -4$, in which case $\rho(I - \mathcal{M}^{-1}\mathcal{A}) = \frac{1}{2}$, and hence the iteration will be convergent.

Suppose we want to do better than the block-diagonal iteration matrix. We could try a block-triangular matrix, for example

$$\mathcal{M} = \begin{bmatrix} A & 0 \\ B & \tau S \end{bmatrix},$$

where again τ is a constant to be determined. Note that this only requires one matrix-vector multiply (with B) per iteration more to invert than the block diagonal iteration matrix, so it is not much more expensive to apply.

This time we have to consider the generalized eigenvalue problem

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} = \lambda \begin{bmatrix} A & 0 \\ B & \tau S \end{bmatrix},$$

which we can write as

$$A\mathbf{u} + B^T\mathbf{p} = \lambda A\mathbf{u} \tag{4.5}$$

$$(1 - \lambda)B\mathbf{u} = \lambda\tau S\mathbf{p}. \tag{4.6}$$

As in the block diagonal case, if $B\mathbf{u} = \mathbf{0}$, then we must have that $\lambda = 1$. Suppose $B\mathbf{u} \neq \mathbf{0}$. Then, as above, from (4.5) we get

$$B\mathbf{u} = \frac{1}{\lambda - 1} S\mathbf{p},$$

and substituting this into (4.6) we get

$$\lambda = -1/\tau.$$

The eigenvalues of $I - \mathcal{M}^{-1}\mathcal{A}$ therefore satisfy $\lambda = 0,\ \lambda = 1 + 1/\tau$, and so the optimal value of τ is therefore $\tau = -1$, with which we will get the exact solution after two iterations. This is not surprising, as if we have an exact solve with S, we can easily recover the solution to the whole system (since S is one block of a block-Gaussian elimination of the matrix (4.1)).

We therefore need to approximate S in some way to have a usable iterative method. The simplest approximation would be to take the identity, so that

$$\mathcal{M} = \begin{bmatrix} A & 0 \\ B & \tau I \end{bmatrix}.$$

In this case, the eigenvalue analysis works as above, and gives us that $\lambda = 1$ or

$$\lambda \mathbf{p} = -\frac{1}{\tau} S\mathbf{p} \Rightarrow \lambda = -\frac{1}{\tau} \cdot \frac{\mathbf{p}^T S \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Now suppose that the eigenvalues of S satisfy

$$\lambda(S) \in [\phi, \Phi],$$

then the non-zero eigenvalues μ of $I - \mathcal{M}^{-1}\mathcal{A}$ satisfy

$$1 + \frac{\phi}{\tau} \leq \mu \leq 1 + \frac{\Phi}{\tau}.$$

As always, the optimal value of τ is when $1 + \frac{\phi}{\tau} = -(1 + \frac{\Phi}{\tau})$, so the optimal value of τ is $-(\phi + \Phi)/2$.

This method is called the Uzawa algorithm [3, 44]. It is usually given in the form of Algorithm 17.

In the Uzawa algorithm, we took the identity as an approximation of the Schur complement, S. Suppose we have some positive definite matrix S_0 which is inexpensive to solve and approximates S in the sense that there exist constants ϕ, Φ such that

$$\phi \leq \frac{\mathbf{x}^T S \mathbf{x}}{\mathbf{x}^T S_0 \mathbf{x}} \leq \Phi \quad \forall \mathbf{x}. \tag{4.7}$$

Algorithm 17 Uzawa algorithm for solving (4.1)

for $k = 0$ **until** convergence, **do**
  Solve $A\mathbf{u}^{(k+1)} = \mathbf{f} - B^T\mathbf{p}^{(k)}$
  $\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} + \frac{1}{\tau}(B\mathbf{u}^{(k+1)} - \mathbf{g})$
end for

Then we can generalize the method above by taking the splitting matrix as

$$\mathcal{M} = \begin{bmatrix} A & 0 \\ B & \tau S_0 \end{bmatrix}.$$

We again have the optimal value $\tau = -(\phi + \Phi)/2$, although here ϕ and Φ are the smallest and largest eigenvalues of $S_0^{-1}S$. This is called the preconditioned Uzawa method, with S_0 being the preconditioner. Algorithm 18 gives the method as it is usually implemented.

Algorithm 18 Preconditioned Uzawa algorithm for solving (4.1) with preconditioner S_0

for $k = 0$ **until** convergence, **do**
  $A\mathbf{u}^{(k+1)} = \mathbf{f} - B^T\mathbf{p}^{(k)}$
  $\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} + (\tau S_0)^{-1}(B\mathbf{u}^{(k+1)} - \mathbf{g})$
end for

All of the methods so far require an exact solve with the (1,1) block, A, at each step. In many practical cases such a solve would be expensive. We would therefore ideally look for some positive definite approximation A_0 to A, such that

$$\delta \leq \frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T A_0\mathbf{x}} \leq \Delta \quad \forall \mathbf{x}. \tag{4.8}$$

Our splitting matrix would then be

$$\mathcal{M} = \begin{bmatrix} A_0 & 0 \\ B & \tau S_0 \end{bmatrix},$$

and this defines a practical method, known as the inexact Uzawa method [38, 21, 130].

Given our experience with the two cases above it is reasonable to assume that the (2,2) block should be negative definite. We will therefore absorb the constant $-\tau$ into our approximation S_0, and look for bounds on the eigenvalues λ given by the generalized eigenvalue problem

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}. \tag{4.9}$$

Note that since the matrix $\widehat{\mathcal{A}} := \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix}^{-1} \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}$ is not generally symmetric, these eigenvalues could be complex. However, we also know that $\widehat{\mathcal{A}}$ is self-adjoint in the inner product defined by $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{H}} := \mathbf{u}^T \mathcal{H} \mathbf{v}$, where

$$\mathcal{H} = \begin{bmatrix} A - A_0 & 0 \\ 0 & S_0 \end{bmatrix},$$

provided that this defines an inner product – i.e. provided $A - A_0$ and S_0 are positive definite [20]. We will consider this case first.

If $A - A_0$ is positive definite, then

$$\mathbf{x} A \mathbf{x}^T > \mathbf{x} A_0 \mathbf{x}^T,$$

and so our assumption (4.8) becomes

$$1 < \delta \leq \frac{\mathbf{x} A \mathbf{x}^T}{\mathbf{x} A_0 \mathbf{x}^T} \leq \Delta. \tag{4.10}$$

Equation (4.7) tells us that

$$\phi \leq \frac{\mathbf{x}^T B A^{-1} B^T \mathbf{x}}{\mathbf{x}^T S_0 \mathbf{x}} \leq \Phi \quad \forall \mathbf{x}. \tag{4.11}$$

Note that if S_0 and A are invertible,

$$B A^{-1} B^T \mathbf{x} = \lambda S_0 \mathbf{x} \Rightarrow A^{-1} B^T S_0^{-1} B A^{-1} B^T \mathbf{x} = \lambda A^{-1} B^T \mathbf{x} \quad \forall \mathbf{x} \tag{4.12}$$

$$\Rightarrow A^{-1} B^T S_0^{-1} B \mathbf{y} = \lambda \mathbf{y} \quad \forall \mathbf{y} = A^{-1} B^T \mathbf{x}, \tag{4.13}$$

and hence (4.11) can also be written as

$$\phi \leq \frac{\mathbf{y}^T B^T S_0^{-1} B \mathbf{y}}{\mathbf{y}^T A \mathbf{y}} \leq \Phi, \tag{4.14}$$

where $\mathbf{y} \in \mathbb{R}^n$ is such that $\mathbf{y} = A^{-1} B^T \mathbf{x}$ for some $\mathbf{x} \in \mathbb{R}^m$. This can be characterized by taking $\mathbf{y} \in \{\mathbf{v} \in \mathbb{R}^n \ : \ \mathbf{v}^T A \mathbf{u} = 0 \text{ if} \mathbf{u} \in \text{null}(B)\}$.

Equation (4.9) can be written as

$$A\mathbf{x} + B^T \mathbf{y} = \lambda A_0 \mathbf{x} \tag{4.15}$$

$$(1 - \lambda) B \mathbf{x} = -\lambda S_0 \mathbf{y}. \tag{4.16}$$

First, note that if we multiply (4.15) on the left by $(1 - \lambda)\mathbf{x}^T$ and subtract (4.16) multiplied on the right by $\mathbf{y}^T$, we get

$$(1 - \lambda) \mathbf{x}^T A \mathbf{x} = (1 - \lambda) \lambda \mathbf{x}^T A_0 \mathbf{x} + \lambda \mathbf{y}^T S_0 \mathbf{y} \tag{4.17}$$

$$< (1 - \lambda) \lambda \mathbf{x}^T A \mathbf{x} + \lambda \mathbf{y}^T S_0 \mathbf{y}, \tag{4.18}$$

the second line being a consequence of (4.10). Rearranging, we get that

$$(1-\lambda)^2\mathbf{x}^T A\mathbf{x} < \lambda \mathbf{y}^T S_0 \mathbf{y}.$$

Clearly $(1-\lambda)^2 \geq 0$, and as A and S_0 are positive definite matrices we must have that $\lambda \geq 0$.

Now, suppose $B\mathbf{x} = 0$. Then, from (4.16), $-\lambda S_0\mathbf{y} = 0 \Rightarrow \mathbf{y} = 0$, since S_0 is positive definite. Therefore, substituting this into (4.15) we get

$$A\mathbf{x} = \lambda A_0 \mathbf{x}.$$

This tells us that

$$\delta \leq \lambda = \frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T A_0 \mathbf{x}} \leq \Delta. \tag{4.19}$$

Now suppose that $B\mathbf{x} \neq 0$. Note that we can rearrange (4.16) to give

$$\mathbf{y} = \frac{\lambda - 1}{\lambda} S_0^{-1} B\mathbf{x},$$

and, substituting this into (4.15) and rearranging, we get

$$\lambda \mathbf{x}^T A\mathbf{x} + (\lambda - 1)\mathbf{x}^T B^T S_0^{-1} B\mathbf{x} = \lambda^2 \mathbf{x}^T A_0 \mathbf{x}.$$

We know that A is positive definite, so $\mathbf{x}^T A\mathbf{x} > 0$, and hence we can write

$$\lambda = \lambda^2 \frac{\mathbf{x}^T A_0 \mathbf{x}}{\mathbf{x}^T A\mathbf{x}} + (1-\lambda)\frac{\mathbf{x}^T B^T S_0^{-1} B\mathbf{x}}{\mathbf{x}^T A\mathbf{x}}. \tag{4.20}$$

Suppose that $\lambda < 1$, so $1 - \lambda > 0$. Then, using (4.10) and (4.14) we get that

$$\lambda \geq \frac{\lambda^2}{\Delta} + (1-\lambda)\phi.$$

We can rearrange this to give

$$\lambda^2 - (1+\phi)\Delta\lambda + \phi\Delta \leq 0.$$

This quadratic in λ is zero at

$$\lambda = \frac{(1+\phi)\Delta \pm \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2},$$

and hence we must have that

$$\frac{(1+\phi)\Delta - \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2} \leq \lambda \leq \frac{(1+\phi)\Delta + \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2}$$

Note, however, that we assumed $\lambda < 1$ to get these bounds, so we want to check if the upper bound is useful. First note that $\Delta > 1$, so $1 - \Delta < 0$. Therefore

$$\begin{aligned}\frac{(1+\phi)\Delta + \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2} &\geq \frac{(1+\phi)\Delta + \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta + 4(1-\Delta)}}{2} \\ &= \frac{(1+\phi)\Delta + \sqrt{((1+\phi)\Delta - 2)^2}}{2} \\ &= \frac{(1+\phi)\Delta + |(1+\phi)\Delta - 2|}{2}.\end{aligned}$$

Therefore the upper bound satisfies

$$\begin{aligned}\frac{(1+\phi)\Delta + \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2} &\geq \frac{(1+\phi)\Delta + |(1+\phi)\Delta - 2|}{2} \\ &= \begin{cases} 1 & \text{if } 2 \geq (1+\phi)\Delta \\ (1+\phi)\Delta - 1 & \text{if } 2 < (1+\phi)\delta \end{cases} \\ &\geq 1,\end{aligned}$$

and hence it is of no use.

Now let's go back to (4.20). Instead of taking an upper bound, we can use the same argument to find a lower bound (again, while $\lambda < 1$.) This gives us

$$\lambda \leq \frac{\lambda^2}{\delta} + (1-\lambda)\Phi,$$

which can be rearranged to give

$$\lambda^2 - (1-\Phi)\delta + \Phi\delta \geq 0.$$

Consideration of the roots of this polynomial gives

$$\begin{aligned}\lambda &\leq \frac{(1+\Phi)\delta - \sqrt{(1+\Phi)^2\delta^2 - 4\Phi\delta}}{2} \\ \lambda &\geq \frac{(1+\Phi)\delta + \sqrt{(1+\Phi)^2\delta^2 - 4\Phi\delta}}{2}.\end{aligned}$$

As above, we can complete the square under the root to give

$$\begin{aligned}\frac{(1+\Phi)\delta + \sqrt{(1+\Phi)^2\delta^2 - 4\Phi\delta}}{2} &\geq \frac{(1+\Phi)\delta - |(1+\Phi)\delta - 2|}{2} \\ &\geq \frac{(1+\Phi)\delta + |(1+\Phi)\delta - 2|}{2} \\ &= (1+\Phi)\delta - 1 \\ &\geq 1,\end{aligned}$$

since, in this case, $\delta \geq 1$ and $\Phi \geq 1$. Therefore the second bound here is of no use.

To summarize our findings so far, we have shown that if $\lambda < 1$, then

$$\frac{(1+\phi)\Delta - \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2} \leq \lambda \leq \frac{(1+\Phi)\delta - \sqrt{(1+\Phi)^2\delta^2 - 4\Phi\delta}}{2}. \quad (4.21)$$

Now let us return again to (4.20):

$$\lambda = \lambda^2 \frac{\mathbf{x}^T A_0 \mathbf{x}}{\mathbf{x}^T A \mathbf{x}} + (1-\lambda)\frac{\mathbf{x}^T B^T S_0^{-1} B\mathbf{x}}{\mathbf{x}^T A\mathbf{x}}.$$

So far we have just considered $\lambda < 1$ – now, what if $\lambda > 1$ and so $1 - \lambda < 0$? We can do the same analysis as above. First, we get the inequality

$$\lambda \leq \frac{\lambda^2}{\delta} + (1-\lambda)\phi,$$

which tells us that

$$\lambda \leq \frac{(1+\phi)\delta - \sqrt{(1+\phi)^2\delta^2 - 4\phi\delta}}{2}$$
$$\lambda \geq \frac{(1+\phi)\delta + \sqrt{(1+\phi)^2\delta^2 - 4\phi\delta}}{2}.$$

The lower bound satisfies

$$\begin{aligned}\frac{(1+\phi)\delta - \sqrt{(1+\phi)^2\delta^2 - 4\phi\delta}}{2} &\leq \frac{(1+\phi)\delta - |(1+\phi)\delta - 2|}{2} \\ &= \begin{cases} 1 & \text{if } 2 \leq (1+\phi)\delta \\ (1+\phi)\delta - 1 & \text{if } 2 > (1+\phi)\delta \end{cases} \\ &\leq 1.\end{aligned}$$

Therefore, this bound is also of no use, due to our assumption that $\lambda > 1$.

Going back to (4.20), we also obtain

$$\lambda \geq \frac{\lambda^2}{\Delta} + (1-\lambda)\Phi,$$

which gives us that

$$\frac{(1+\Phi)\Delta - \sqrt{(1+\Phi)^2\Delta^2 - 4\Phi\Delta}}{2} \leq \lambda \leq \frac{(1+\Phi)\Delta + \sqrt{(1+\Phi)^2\Delta^2 - 4\Phi\Delta}}{2}.$$

Note that, in this case,

$$\begin{aligned}\frac{(1+\Phi)\Delta - \sqrt{(1+\Phi)^2\Delta^2 - 4\Phi\Delta}}{2} &\leq \frac{(1+\Phi)\Delta - |(1+\Phi)\Delta - 2|}{2} \\ &= 1,\end{aligned}$$

since $\Phi, \Delta > 0$. Hence the lower bound is of no use, and we have shown that

$$\frac{(1+\phi)\delta + \sqrt{(1+\phi)^2\delta^2 - 4\phi\delta}}{2} \leq \lambda \leq \frac{(1+\Phi)\Delta + \sqrt{(1+\Phi)^2\Delta^2 - 4\Phi\Delta}}{2}. \quad (4.22)$$

We have just proved Theorem 4.1.1:

Theorem 4.1.1. *Let λ be an eigenvalue of the generalized eigenvalue problem*

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix},$$

where A, S_0, A_0 and $A - A_0$ are positive definite. Then λ is real and positive, and moreover satisfies

$$\begin{aligned} \frac{(1+\phi)\Delta - \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2} \leq &\lambda \leq \frac{(1+\Phi)\delta - \sqrt{(1+\Phi)^2\delta^2 - 4\Phi\delta}}{2} \\ \delta \leq &\lambda \leq \Delta \qquad \text{or} \\ \frac{(1+\phi)\delta + \sqrt{(1+\phi)^2\delta^2 - 4\phi\delta}}{2} \leq &\lambda \leq \frac{(1+\Phi)\Delta + \sqrt{(1+\Phi)^2\Delta^2 - 4\Phi\Delta}}{2}, \end{aligned}$$

where ϕ, Φ, δ and Δ are measures of the effectiveness of the approximation of A_0 to A and S_0 to $BA^{-1}B^T$, as defined by (4.10) and (4.7).

From this it is trivial to find the spectral radius of $I - \mathcal{M}^{-1}\mathcal{A}$:

Corollary 4.1.1. *Suppose that the eigenvalues of (4.9) are as described in Theorem 4.1.1, and $A - A_0 > 0$. Define*

$$\begin{aligned} \xi := \max \Bigg\{ & 1 - \delta, 1 - \frac{(1+\phi)\Delta - \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2}, \\ & \Delta - 1, \frac{(1+\Phi)\Delta + \sqrt{(1+\Phi)^2\Delta^2 - 4\Phi\Delta}}{2} - 1 \Bigg\}. \end{aligned}$$

Then a simple iteration based on the splitting matrix

$$\mathcal{M} = \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix}$$

will converge if $\xi < 1$, with the asymptotic convergence rate being ξ.

An alternative derivation of this result has been given by Zulehner [130]. Zulehner's method is based on factorizing the iteration matrix $I - \mathcal{M}^{-1}\mathcal{A} = \mathcal{P}\mathcal{N}\mathcal{Q}$, where $\mathcal{P}$ and $\mathcal{Q}$ are block diagonal, and $\mathcal{N}$ is symmetric. Since $A - A_0 > 0$ there exists a matrix $\mathcal{E}$ such that the eigenvalues of the iteration matrix are the same as those of the generalized eigenvalue problem

$$\mathcal{N}\mathbf{x} = \lambda\mathcal{E}\mathbf{x}. \tag{4.23}$$

Corollary 4.1.1 agrees with Theorem 4.1 in Zulehner [130].

If we explicitly include our parameter τ again here, note that finding the optimal value for a give approximation S_0 is no longer trivial. However, it is reasonable to assume that the value obtained for the exact case $A = A_0$ is a good value here also – this is the approach used in, for example, [38].

In the case where $A - A_0$ is indefinite the situation is more complicated, as now the eigenvalues will in general be complex. Recall the generalized eigenvalue problem:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}.$$

We still assume that A, A_0 and S_0 are positive definite. Again, if $B\mathbf{x} = 0$, we still have that $\lambda = \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T A_0 \mathbf{x}}$, and hence these eigenvalues must be real, with

$$\delta \leq \lambda \leq \Delta.$$

All that changes here is now $\delta < 1$.

Suppose $B\mathbf{x} \neq 0$. Equation (4.20) still holds, namely:

$$\lambda = \lambda^2 \frac{\mathbf{x}^T A_0 \mathbf{x}}{\mathbf{x}^T A \mathbf{x}} + (1 - \lambda) \frac{\mathbf{x}^T B^T S_0^{-1} B \mathbf{x}}{\mathbf{x}^T A \mathbf{x}}.$$

If we define

$$\kappa := \kappa(\mathbf{x}) = \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T A_0 \mathbf{x}}, \qquad \sigma := \sigma(\mathbf{x}) = \frac{\mathbf{x}^T B^T S_0^{-1} B \mathbf{x}}{\mathbf{x}^T A \mathbf{x}},$$

then we can write this as

$$\lambda^2/\kappa + (1 - \lambda)\sigma - \lambda = 0,$$

or, alternatively,

$$\lambda^2 - (\sigma + 1)\kappa\lambda + \sigma\kappa = 0.$$

Therefore the eigenvalues satisfy

$$\lambda = \frac{(\sigma + 1)\kappa \pm \sqrt{(\sigma + 1)^2\kappa^2 - 4\sigma\kappa}}{2}.$$

We know from above that if $\kappa = \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T A_0 \mathbf{x}} \geq 1$, then all the eigenvalues are real. Note that

$$(\sigma + 1)^2\kappa^2 - 4\sigma\kappa = 0 \Rightarrow \kappa = 0 \text{ or } \kappa = \frac{4\sigma}{(1 + \sigma)^2} \leq 1,$$

the last inequality being since $\frac{4\sigma}{(1+\sigma)^2}$ has a maximum value of 1 which occurs when $\sigma = 1$. This tells us that for $\kappa \in [0, \frac{4\sigma}{(1+\sigma)^2}]$, $\lambda \in \mathbb{C}$.

In this case,

$$\begin{aligned}\lambda &= \frac{(\sigma+1)\kappa \pm i\sqrt{4\sigma\kappa - (\sigma+1)^2\kappa^2}}{2} \\ \Rightarrow |\lambda|^2 &= \frac{(\sigma+1)^2\kappa^2 + 4\sigma\kappa - (\sigma+1)^2\kappa^2}{4} \\ &= \sigma\kappa.\end{aligned}$$

Therefore the complex eigenvalues satisfy

$$\sqrt{\delta\phi} \leq |\lambda| \leq \sqrt{\Phi}. \tag{4.24}$$

Moreover, $\mathrm{Re}(\lambda) = \frac{(\sigma+1)\kappa}{2} > 0$, so all the complex eigenvalues live in the right-hand plane. Also,

$$\begin{aligned}\frac{|\mathrm{Im}(\lambda)|}{\mathrm{Re}(\lambda)} &= \frac{\sqrt{4\sigma\kappa - (\sigma+1)^2\kappa^2}}{2} \cdot \frac{2}{(\sigma+1)\kappa} \\ &= \frac{\sqrt{4\sigma\kappa - (\sigma+1)^2\kappa^2}}{(\sigma+1)\kappa}.\end{aligned}$$

If we define

$$F(\sigma,\kappa) := \frac{\sqrt{4\sigma\kappa - (\sigma+1)^2\kappa^2}}{(\sigma+1)\kappa},$$

then

$$\frac{\partial F}{\partial \sigma} = \frac{2(\sigma-1)}{(\sigma+1)^2\sqrt{(4-2\kappa)\kappa\sigma - \kappa^2(\sigma^2+1)}},$$

so

$$\frac{\partial F}{\partial \sigma} = 0 \Rightarrow \sigma = 1.$$

This critical point is clearly a maximum. This means that, for any fixed κ, $F(\sigma,\kappa)$ has it's maximum at $\sigma = 1$. Therefore

$$\frac{|\mathrm{Im}(\lambda)|}{\mathrm{Re}(\lambda)} = F(\sigma,\kappa) \leq \frac{\sqrt{\kappa - \kappa^2}}{\kappa} = \sqrt{\frac{1}{\kappa} - 1} \leq \sqrt{\frac{1}{\delta} - 1}.$$

Therefore, putting this together with (4.24) above, the complex eigenvalues satisfy

$$\lambda \in \left\{ z = re^{i\theta} \in \mathbb{C} \ : \ \sqrt{\delta\phi} \leq r \leq \sqrt{\Phi}, \ -\tan^{-1}(\sqrt{\delta^{-1}-1}) \leq \theta \leq \tan^{-1}(\sqrt{\delta^{-1}-1}) \right\}. \tag{4.25}$$

For $\kappa > 1$, the results proved for $A - A_0$ positive definite still hold, and we have $\lambda \in \mathbb{R}$ which satisfy

$$\frac{(\phi+1)\Delta - \sqrt{(\phi+1)^2\Delta^2 - 4\phi\Delta}}{2} \leq \lambda \leq \frac{(\Phi+1)\Delta + \sqrt{(\Phi+1)^2\Delta^2 - 4\Phi\Delta}}{2}.$$

What about $\kappa \in \left[\frac{4\sigma}{(1+\sigma)^2}, 1\right]$? In this case, too, the bounds in Theorem 4.1.1 hold, since in the derivation of these bounds we required no information about δ – all we assumed was that $\lambda \in \mathbb{R}$. Verifying the inner bounds required that $\delta > 1$, so these do not carry over, but there is no such problem with the outer bounds.

We have proved the following theorem:

Theorem 4.1.2. *Let λ be an eigenvalue associated with the generalized eigenvalue problem*

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix},$$

where A, A_0 and S_0 are positive definite. If $\lambda \in \mathbb{R}$, then it satisfies

$$\begin{aligned} \frac{(1+\phi)\Delta - \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2} \leq & \lambda \leq \frac{(1+\Phi)\Delta + \sqrt{(1+\Phi)^2\Delta^2 - 4\Phi\Delta}}{2} \\ \text{or} \qquad \delta \leq & \lambda \leq \Delta, \end{aligned}$$

and if $\lambda \in \mathbb{C}$, then $\lambda = re^{i\theta}$, where r and θ satisfy

$$\sqrt{\delta\phi} \leq r \leq \sqrt{\Phi}, \ -\tan^{-1}(\sqrt{\delta^{-1}-1}) \leq \theta \leq \tan^{-1}(\sqrt{\delta^{-1}-1}).$$

Here ϕ, Φ, δ and Δ are measures of the effectiveness of the approximation of A_0 to A and S_0 to $BA^{-1}B^T$, as defined by (4.8) and (4.7).

To get bounds for $\rho(I - P_{BLT}^{-1}\mathcal{A})$ we have to be more careful because of the presence of the complex eigenvalues. Figure 4.1 shows a diagram of the situation here. All the complex eigenvalues will be contained in the unit circle if $d < 1$. By the cosine rule:

$$d^2 = 1 + \Phi - 2\sqrt{\Phi}\cos\theta,$$

where $\tan\theta = \sqrt{\delta^{-1} - 1}$. Therefore all the complex eigenvalues are in the unit circle if

$$\frac{\sqrt{\Phi}}{2} < \cos\theta.$$

Note that, using the same argument as above, the distance from the origin to the point where the circle of radius $\sqrt{\phi\delta}$, centre -1, touches the ray that makes an angle θ with the x-axis is

$$\sqrt{1 + \phi\delta - 2\sqrt{\phi\delta}\cos\theta}.$$

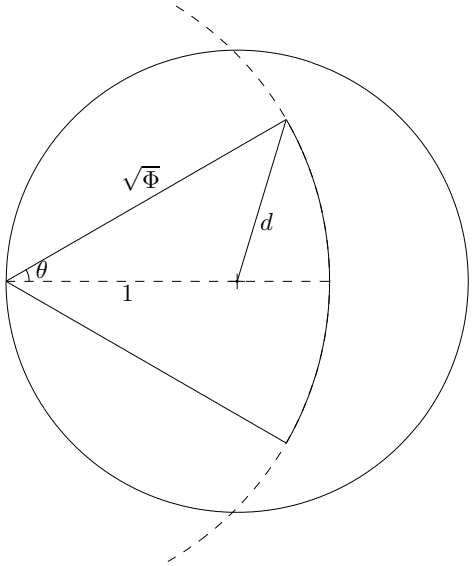


Figure 4.1: Diagram of the geometry containing the complex eigenvalues. $\theta = \sqrt{\delta^{-1} - 1}$ and d is the unknown length.

.

We have just proved the following:

Corollary 4.1.2. *Suppose that the eigenvalues of (4.9) are as described in Theorem 4.1.2. Define*

$$\xi := \max\left\{1-\delta, \Delta - 1, 1 - \frac{(1+\phi)\Delta - \sqrt{(1+\phi)^2\Delta^2 - 4\phi\Delta}}{2}, \right.$$
$$\frac{(1+\Phi)\Delta + \sqrt{(1+\Phi)^2\Delta^2 - 4\Phi\Delta}}{2} - 1, \sqrt{1+\Phi-2\sqrt{\Phi}\cos\theta},$$
$$\left.\sqrt{1+\phi\delta-2\sqrt{\phi\delta}\cos\theta}\right\}.$$

Then a simple iteration with splitting matrix

$$\mathcal{M} = \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix}$$

will converge if $\xi < 1$, with the asymptotic convergence rate being ξ.

Zulehner also derived an approximation to the convergence factor [130, Theorem 4.3]. Note that Corollary 4.1.2 differs slightly from the result in Zulehner – this is because neither the result given here nor in [130] are sharp with regards to the

complex eigenvalues. The two results are obtained in very different ways – in this case, Zulehner defines the matrix $\mathcal{E}$ in (4.23) via an arbitrary function of a matrix, then picks a specific instance to derive bounds – so neither can be said to be a better approximation than the other one. We compare the bounds in the specific case of a discretization of the Stokes equations in Chapter 7.

As pointed out by Jiránek and Rozložník [72], there are at least three different ways of writing the inexact Uzawa algorithm described above in the literature. While all are equivalent in exact arithmetic, the version they show to give the maximum attainable accuracy is given in Algorithm 19. We observed that if you treat the Uzawa algorithm as a splitting of a simple iteration, as described above, then the accuracy of that method is the same as Algorithm 19.

Algorithm 19 Preconditioned inexact Uzawa algorithm for solving (4.1)

for $k = 0$ **until** convergence, **do**
  $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + A_0^{-1}(\mathbf{f} - A\mathbf{u}^{(k)} - B^T\mathbf{p}^{(k)})$
  $\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} + S_0^{-1}(B\mathbf{u}^{(k+1)} - \mathbf{g})$
end for

4.2 Block diagonal preconditioners for MINRES

Suppose, instead of a simple iteration, we want to take advantage of the superlinear convergence properties of a Krylov-subspace method to solve the system (4.1). Since the matrix is typically symmetric, but indefinite, we cannot use the standard Conjugate Gradient method, Algorithm 8, so instead the method of choice is MINRES, Algorithm 11.

In order for MINRES to be effective we need a good preconditioner, $\mathcal{P}$. Recall from Section 3.1.7 that a preconditioner for MINRES must be symmetric positive definite. Drawing from our experience in Section 4.1, a good choice would seem to be

$$\mathcal{P} = \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix},$$

where $S = BA^{-1}B^T$, the Schur complement. Murphy, Golub and Wathen [87] showed that, with this choice of preconditioner, as long as $\mathcal{A}$ is nonsingular, then the preconditioned system is diagonalizable, with the eigenvalues of the generalized eigenvalue problem $\mathcal{A}\mathbf{x} = \lambda\mathcal{P}\mathbf{x}$ satisfying

$$\lambda = 1, \text{ or } \lambda = \frac{1 \pm \sqrt{5}}{2}.$$

Therefore if $\mathcal{P}$ was used as our preconditioner the theory in Section 3.1.7 tells us that we should get convergence in at most three iterations.

Recall that we have to solve a system with our preconditioner at each MINRES iteration. We run into the same difficulty that we had in Section 4.1 – an exact solve with S is basically solving the system, and in many cases a solve with A is computationally expensive. We therefore would like to replace S and A in our 'ideal' preconditioner above by some positive definite approximations S_0 and A_0 such that again they satisfy (4.7) and (4.8), i.e.

$$\phi \leq \frac{\mathbf{x}^T S \mathbf{x}}{\mathbf{x}^T S_0 \mathbf{x}} \leq \Phi \quad \text{and} \quad \delta \leq \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T A_0 \mathbf{x}} \leq \Delta \quad \forall \mathbf{x}.$$

Our practical preconditioner would therefore become

$$\mathcal{P} = \begin{bmatrix} A_0 & 0 \\ 0 & S_0 \end{bmatrix}.$$

Of course, the choice of approximation S_0 and A_0 is, as in the inexact Uzawa case, highly problem dependent. We will discuss good choices of approximation for the problems we would like to solve in the following chapters.

We now show how these approximations change the clustering of the eigenvalues. We are interested in the eigenvalues of the generalized eigenvalue problem

$$A\mathbf{u} + B^T \mathbf{p} = \lambda A_0 \mathbf{u} \tag{4.26}$$

$$B\mathbf{u} = \lambda S_0 \mathbf{p}. \tag{4.27}$$

First note that all the eigenvalues are real, as we have a symmetric matrix $\mathcal{A}$ and a symmetric positive definite matrix $\mathcal{P}$. As we saw in Section 4.1, if $B\mathbf{u} = 0$, then $\mathbf{p} = 0$ and so

$$\delta \leq \lambda \leq \Delta.$$

Suppose $B\mathbf{u} \neq 0$.

Using (4.27) to eliminate $\mathbf{p}$ from (4.26) we get

$$\lambda \mathbf{u}^T A_0 \mathbf{u} - \mathbf{u}^T A \mathbf{u} = \frac{1}{\lambda} \mathbf{u}^T B^T S_0^{-1} B \mathbf{u},$$

and since A_0 is positive definite we get

$$\lambda = \frac{\mathbf{u}^T A \mathbf{u}}{\mathbf{u}^T A_0 \mathbf{u}} + \frac{1}{\lambda} \cdot \frac{\mathbf{u}^T B^T S_0^{-1} B \mathbf{u}}{\mathbf{u}^T A_0 \mathbf{u}} = \frac{\mathbf{u}^T A \mathbf{u}}{\mathbf{u}^T A_0 \mathbf{u}} + \frac{1}{\lambda} \cdot \frac{\mathbf{u}^T B^T S_0^{-1} B \mathbf{u}}{\mathbf{u}^T A \mathbf{u}} \cdot \frac{\mathbf{u}^T A \mathbf{u}}{\mathbf{u}^T A_0 \mathbf{u}}. \tag{4.28}$$

Now, suppose that $\lambda > 0$, then applying (4.8) and (4.14) we get the relation

$$\delta + \frac{\phi\delta}{\lambda} \leq \lambda \leq \Delta + \frac{\Phi\Delta}{\lambda}. \tag{4.29}$$

Consider the right hand inequality in (4.29) first; rearranging, we get

$$\lambda^2 - \lambda\Delta - \Phi\Delta \leq 0.$$

This tells us that λ must lie between the roots of the quadratic equation, i.e.

$$\frac{\Delta - \sqrt{\Delta^2 + 4\Delta\Phi}}{2} \leq \lambda \leq \frac{\Delta + \sqrt{\Delta^2 + 4\Delta\Phi}}{2}.$$

Since we've assumed that $\lambda > 0$, only the upper bound is meaningful here. Now consider the left hand side of (4.29). This gives us

$$\lambda^2 - \lambda\delta - \phi\delta \geq 0,$$

and hence λ satisfies

$$\lambda \leq \frac{\delta - \sqrt{\delta^2 + 4\phi\delta}}{2} \text{ or } \lambda \geq \frac{\delta + \sqrt{\delta^2 + 4\phi\delta}}{2}.$$

Again, $\lambda > 0$ tells us only to consider the lower bound.

Suppose that $\lambda < 0$. Then the application of (4.8) and (4.14) to (4.28) gives us

$$\delta + \frac{\Phi\Delta}{\lambda} \leq \lambda \leq \Delta + \frac{\phi\delta}{\lambda}, \tag{4.30}$$

which in turn, noting that the negative λ changes the sign of the inequality, leads to

$$\frac{\delta - \sqrt{\delta^2 + 4\Delta\Phi}}{2} \leq \lambda \leq \frac{\delta + \sqrt{\delta^2 + 4\Delta\Phi}}{2},$$

and

$$\lambda \leq \frac{\Delta - \sqrt{\Delta^2 + 4\phi\delta}}{2} \text{ or } \lambda \geq \frac{\Delta + \sqrt{\Delta^2 + 4\phi\delta}}{2}.$$

Again, we have assumed that $\lambda > 0$, so only the negative values of the square roots are useful. Therefore we have that

$$\frac{\delta - \sqrt{\delta^2 + 4\Delta\Phi}}{2} \leq \lambda \leq \frac{\Delta - \sqrt{\Delta^2 + 4\phi\delta}}{2},$$

and we have proved the following:

Theorem 4.2.1. *Let λ be an eigenvalue associated with the generalized eigenvalue problem*

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \lambda \begin{bmatrix} A_0 & 0 \\ 0 & S_0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix},$$

where A, A_0 and S_0 are positive definite. Then $\lambda \in \mathbb{R}$, and satisfies

$$\frac{\delta - \sqrt{\delta^2 + 4\Delta\Phi}}{2} \leq \lambda \leq \frac{\Delta - \sqrt{\Delta^2 + 4\phi\delta}}{2},$$
$$\delta \leq \lambda \leq \Delta,$$
$$\text{or} \qquad \frac{\delta + \sqrt{\delta^2 + 4\delta\phi}}{2} \leq \lambda \leq \frac{\Delta + \sqrt{\Delta^2 + 4\Phi\Delta}}{2}.$$

Here ϕ, Φ, δ and Δ are measures of the effectiveness of the approximation of A_0 to A and S_0 to $BA^{-1}B^T$, as defined by (4.8) and (4.7).

Therefore, as long as we choose approximations A_0 and S_0 that are close enough (in the sense defined above) to A and S, then we can also expect good clustering of the eigenvalues in the generalized eigenvalue problem above, and hence good convergence of MINRES.

4.3 Bramble-Pasciak CG

Suppose we did want to use a conjugate gradient method, instead of MINRES, to solve a saddle point problem (4.1). Since (4.1) is not positive definite, the standard conjugate gradient algorithm cannot be used. However, we noted in Section 4.1 that

$$\begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix}^{-1} \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}$$

is self-adjoint with respect to the inner product defined by $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{H}} := \mathbf{u}^T \mathcal{H} \mathbf{v}$, where

$$\mathcal{H} = \begin{bmatrix} A - A_0 & 0 \\ 0 & S_0 \end{bmatrix},$$

provided that this defines an inner product – i.e. when $A - A_0$ and S_0 are positive definite. Therefore can we apply the conjugate gradient algorithm with this inner product, along with preconditioner

$$\mathcal{P} = \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix},$$

as described in Section 3.1.6. This method was first described by Bramble and Pasciak in [20], and has since generated a lot of interest – see, for example, [37, 75, 85, 103, 78, 110, 32].

As described in Section 3.1.5, the convergence of this method depends on the eigenvalue distribution of the preconditioned system. We therefore need to find the eigenvalues of the generalized eigenvalue problem

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}. \tag{4.31}$$

This is the same as (4.9), and so these eigenvalues are again given by Theorem 4.1.1. Note that it is a requirement for the method to work that $A - A_0$ is positive definite, so Theorem 4.1.1 will describe all the eigenvalues – i.e. all the eigenvalues are real.

Generally our approximation to A or S will not be in the form of a matrix, but a linear process that gives the action of the inverse of the matrix. This is the case, for example, when we use multigrid, as described in Section 3.2. It is not obvious that we can apply this method in such cases, since we do not explicitly have the matrix approximation A_0 or S_0. However, it was shown by Bramble and Pasciak [20] that even in this case the method is applicable, since only the action of the inverse of A_0 is needed when evaluating the inner products with $\mathcal{H}$. Elman [37] showed that the same is true for the Schur-complement approximation S_0.

We will demonstrate this for one of the quantities involved – full details can be found in [20, 37, 109]. We need to compute $\left\langle \mathcal{P}^{-1}\mathcal{A}\mathbf{r}^{(k)}, \mathbf{p}^{(k)} \right\rangle_{\mathcal{H}}$ (see Algorithm 10). Now,

$$\begin{aligned}\mathcal{H}\mathcal{P}^{-1} &= \begin{bmatrix} A - A_0 & 0 \\ 0 & S_0 \end{bmatrix} \begin{bmatrix} A_0^{-1} & 0 \\ S_0^{-1}BA_0^{-1} & -S_0^{-1} \end{bmatrix} \\ &= \begin{bmatrix} AA_0^{-1} - I & 0 \\ BA_0^{-1} & -I \end{bmatrix},\end{aligned}$$

so we can compute this inner product without having to multiply with the approximation matrices A_0 or S_0. The same is true throughout the algorithm – see, e.g., the thesis of M. Stoll [109] for more details.

The preconditioner $\mathcal{P}$ has to be applied only once per iteration. In order to evaluate the inner product with $\mathcal{H}$ no extra multiplications with blocks of the saddle point matrix are required (see [37] for details). The extra cost of the Bramble-Pasciak method compared to MINRES with block-diagonal preconditioning is therefore just one additional multiplication with the B block of the matrix [37]. Hence, we can have the superior convergence properties of a conjugate gradient method with an indefinite system.

As described in Section 3.1.6, the drawback with this method is that you need $A - A_0$ positive definite; this means that not just any approximation to A will do. This property usually results in having to find the eigenvalues of $A_0^{-1}A$ for a candidate A_0, and then adding an appropriate scaling parameter γ so that $A > \gamma A_0$. We will describe a specific situation to which this applies in Section 5.2.

4.4 Null space methods and Projected Conjugate Gradients

4.4.1 Null space methods

Consider again the saddle point system equation (4.1), which we write in the form

$$A\mathbf{x} + B^T\mathbf{y} = \mathbf{f}$$
$$B\mathbf{x} = \mathbf{g}.$$

Recall that $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times n}$, $m < n$. In the following we will assume that $\ker(A) \cap \ker(B) = \{0\}$. We will now discuss a method of simplifying the solution of this system, called a null space, or reduced Hessian, method. The discussion follows the presentation in the paper by Benzi, Golub and Liesen [9, Section 6].

Suppose that we have a vector $\widehat{\mathbf{x}}$ that satisfies $B\widehat{\mathbf{x}} = \mathbf{g}$. If the saddle point matrix came from an optimization problem then this would mean that $\widehat{\mathbf{x}}$ satisfies the constraints. Suppose also that we have a matrix $Z \in \mathbb{R}^{n\times(n-m)}$, the columns of which span the null space of B. As B has rank m there must exist a permutation matrix P such that $BP = [B_1\ B_2]$, where $B_1 \in \mathbb{R}^{m\times m}$ is non-singular. Then such a null basis Z would be given by

$$Z = P \begin{bmatrix} -B_1^{-1}B_2 \\ I \end{bmatrix}.$$

Such a null basis is called a *fundamental basis*.

If $\mathbf{x}$ is a solution of $B\mathbf{x} = \mathbf{g}$, then $\mathbf{x} = Z\mathbf{w} + \widehat{\mathbf{x}}$, where $\mathbf{w} \in \mathbb{R}^{(n-m)}$. Substituting this into the first equation of the saddle point problem, we get

$$AZ\mathbf{w} + A\widehat{\mathbf{x}} + B^T\mathbf{y} = \mathbf{f},$$

and hence premultiplying by Z^T and rearranging gives

$$Z^TAZ\mathbf{w} = Z^T(\mathbf{f} - A\widehat{\mathbf{x}}),$$

where we have used the fact that $Z^TB^T = 0$. This is a much smaller problem than (4.1), and so is correspondingly easier to solve. Once we have found $\mathbf{w}$, we set $\mathbf{x} = Z\mathbf{w} + \widehat{\mathbf{x}}$ and we can obtain $\mathbf{y}$ by solving

$$BB^T\mathbf{y} = B(\mathbf{f} - A\mathbf{x}),$$

or equivalently by solving the overdetermined least squares problem

$$\min_{\mathbf{y}} \|(\mathbf{f} - A\mathbf{x}) - B^T\mathbf{y}\|_2.$$

Note that in method the matrix A need not be invertible, which is contrary to the assumptions we needed in the previous sections of this chapter. If $(n-m)$ is small, then we can apply a direct method, say, to solve the small matrix Z^TAZ, and this is an attractive method. However, the drawback with this method as presented here is that we need to find the null space matrix Z, and this is not always trivial. For more details see Benzi, Golub and Leisen [9, Section 6] and the references therein.

4.4.2 Projected Conjugate Gradients

It is natural to try and apply a Krylov subspace method to the reduced matrix Z^TAZ. Assume that this matrix is symmetric positive definite, which would be the case if A is symmetric and positive definite on the null space of B. Then we can apply CG to the reduced system. Because of the form of the reduced matrix natural choice of preconditioner would be a matrix M of the form $M = Z^TGZ$, for some symmetric matrix G such that P is positive definite.

One way of doing this, which has the advantage that it avoids the explicit use of the null space matrix Z, is the *projected conjugate gradient* algorithm, developed by Gould, Hribar and Nocedal in [52]. They modify the standard CG algorithm – which would give the vector $\mathbf{w}$ in the notation above – so that it explicitly calculates the vector $\mathbf{x}$. If we define the scaled projection matrix $P := Z(Z^TGZ)^{-1}Z^T$, which is independent of the choice of null space basis Z, then we give this expanded form of CG as Algorithm 20.

Algorithm 20 Projected Preconditioned Conjugate Gradients in expanded form (Algorithm II in [52])

Choose an initial point $\mathbf{x}^{(0)}$ satisfying $B\mathbf{x}^{(0)} = \mathbf{f}$
Compute $\mathbf{r}^{(0)} = A\mathbf{x}^{(0)} - \mathbf{f}$, $\mathbf{z}^{(0)} = P\mathbf{r}^{(0)}$, $\mathbf{p}^{(0)} = -\mathbf{z}^{(0)}$.
for $k = 0, 1, 2...$**until** convergence **do**
  $\alpha_k = \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle / \left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \alpha_k A\mathbf{p}^{(k)}$
  $\mathbf{z}^{(k+1)} = P\mathbf{r}_{k+1}$
  $\beta_k = \left\langle \mathbf{z}^{(k+1)}, \mathbf{r}^{(k+1)} \right\rangle / \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle$
  $\mathbf{p}^{(k+1)} = -\mathbf{z}^{(k+1)} + \beta_k \mathbf{p}^{(k)}$
  $\mathbf{z}^{(k+1)} = -\mathbf{z}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k+1)} - B^T\mathbf{x}^{(k+1)}$
  TEST FOR CONVERGENCE
end for

In Algorithm 20, $\mathbf{z}^{(k)}$ is defined by $\mathbf{z}^{(k)} = P\mathbf{r}^{(k)}$, so this is the preconditioned residual. This lies in the null space of B, which implies all of the iterates $\mathbf{x}_k$ will satisfy the constraint, $B\mathbf{x}^{(k)} = \mathbf{f}$, at least in exact arithmetic.

Now we would like to perform a matrix-vector multiply with P without needing the null-space Z. First, note [47, Section 5.4.1] that P can be written as

$$P = G^{-1}(I - B^T(BG^{-1}B^T)^{-1}BG^{-1}),$$

and so it can be shown that $\mathbf{z}^{(k)} = P\mathbf{r}^{(k)}$ can be found by solving the saddle point problem

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}^{(k)} \\ \mathbf{v}^{(k)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{(k)} \\ 0 \end{bmatrix} \tag{4.32}$$

where G is positive definite on the null space of A. Algorithm 21 gives a version of preconditioned projected conjugate gradients which replaces the step $\mathbf{z}^{(k)} = P\mathbf{r}^{(k)}$ by solving (4.32) to define $\mathbf{z}^{(k)}$. The form of the system that we have to solve in (4.32), the preconditioner, is that of a constraint preconditioner – that is, only the (1,1) block is changed, and the (2,1) and (1,2) blocks are applied exactly. Algorithm 21 is a variant of the algorithm which incorporates this.

Algorithm 21 Preconditioned Projected Conjugate Gradients with residual update

Choose an initial point $\mathbf{x}^{(0)}$ satisfying $B\mathbf{x}^{(0)} = \mathbf{g}$
Solve $\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}^{(0)} \\ \mathbf{v}^{(0)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{(0)} \\ \mathbf{0} \end{bmatrix}$
$\mathbf{p}^{(0)} = -\mathbf{z}^{(0)}$, $\mathbf{r}^{(0)} = \mathbf{r}^{(0)} - B^T\mathbf{v}^{(0)}$.
for $k = 0, 1, 2...$**until** convergence **do**
  $\alpha_k = \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle / \left\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \alpha_k A\mathbf{p}^{(k)}$
  Solve $\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}^{(k+1)} \\ \mathbf{v}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{(k+1)} \\ \mathbf{0} \end{bmatrix}$
  $\beta_k = \left\langle \mathbf{z}^{(k+1)}, \mathbf{r}^{(k+1)} \right\rangle / \left\langle \mathbf{z}^{(k)}, \mathbf{r}^{(k)} \right\rangle$
  $\mathbf{p}^{(k+1)} = -\mathbf{z}^{(k+1)} + \beta_k\mathbf{p}^{(k)}$
  $\mathbf{z}^{(k+1)} = -\mathbf{z}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k+1)} - B^T\mathbf{x}^{(k+1)}$
  TEST FOR CONVERGENCE
end for

This method does have the drawback that it relies on the constraint $B\mathbf{x}^{(k)} = \mathbf{f}$ being satisfied at each step and, while this is true in exact arithmetic, roundoff errors can cause the method to fail. Algorithm 21 incorporates a residual update strategy,

suggested by Gould, Hribar and Nocedal, which reduces the roundoff errors in the projection operation.

Of course, Algorithm 21 itself requires a solution of a saddle point system at each iteration, which is not, in general, any cheaper than solving the system (4.1). Therefore when using this method we have to be careful to choose constraint preconditioners that are themselves easy to invert – for example, because of the structure of the matrix.

One tool that could be used to invert our constraint preconditioner is the Schilders factorization [34, 33]. Suppose we split our preconditioner, or a permutation of the rows and columns if necessary, up into a three-by-three block structure,

$$\mathcal{P} = \begin{bmatrix} G_{ii} & G_{iv} & B_i^T \\ G_{vi} & G_{vv} & B_v^T \\ B_i & B_v & 0 \end{bmatrix},$$

where B_i is a square non-singular matrix. Then we can factorize the matrix $\mathcal{P}$ as

$$\mathcal{P} = \begin{bmatrix} B_i^T & 0 & L_i \\ B_v^T & L_v & E \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} D_i & 0 & I \\ 0 & D_v & 0 \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} B_i & B_v & 0 \\ 0 & L_v^T & 0 \\ L_i^T & E^T & I \end{bmatrix},$$

where

$$\begin{aligned} D_i &= B_i^{-T} G_{i,i} B_i^{-1} - L_i^T B_i^{-1} - B_i^{-T} L_i \\ D_v &= L_v^{-1}(G_{v,v} - B_v^T D_i B_v - E B_v - B_v^T E^T) L_v^{-T} \\ E &= G_{v,i} B_i^{-1} - B_v^T D_i - B_v^T L_i^T B_i^{-1}. \end{aligned}$$

Then each of the matrices in this factorization is a permutation of a block triangular matrix, and therefore can be solved by inverting the pivot matrices. Note that the matrices L_v and L_i are not fixed by the factorization, so typically values of these would be chosen that make the application of the preconditioner easier. See the thesis of Dollar [31] for a comprehensive analysis of the properties of the factorization.

As with the standard Krylov subspace methods, the convergence of PPCG also depends on the clustering of the eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$. The following result was proved by Keller, Gould and Wathen [74].

Theorem 4.4.1. *Let $\mathcal{A}$ and $\mathcal{P}$ be as above, where A, $G \in \mathbb{R}^{n\times n}$ are symmetric ($A \neq G$) and $B \in \mathbb{R}^{m\times n}$ is of full rank. Assume Z is an $n \times (n-m)$ basis for the nullspace of B. Then the matrix $\mathcal{P}^{-1}\mathcal{A}$ has*

- *an eigenvalue at 1 with multiplicity $2m$, and*

- *the remaining $n - m$ eigenvalues defined by the generalized eigenvalue problem $Z^T AZ\mathbf{x} = \lambda Z^T GZ\mathbf{x}$.*

Furthermore, if G is nonsingular then these eigenvalues interlace those of $G^{-1}A$.

Theorem 4.4.1 tells us that as we should aim for a (1,1) block, G, in our constraint preconditioner that is a good approximation of A, while bearing in mind that $\mathcal{P}$ needs to be easily inverted. We will discuss a choice of G for our application in Section 5.6.

For a more detailed discussion above constraint preconditioners and the projected conjugate gradient algorithm, see the thesis of H.S. Dollar [31].

Chapter 5

Preconditioners for the optimal control of Poisson's equation

We return to the PDE-constrained optimization problem

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||_2^2 + \frac{\beta}{2}||u||_2^2$$

$$\begin{aligned} \text{s.t.} \qquad -\nabla^2 y &= u \text{ in } \Omega \\ y &= f \text{ on } \partial\Omega. \end{aligned}$$

In Chapter 2 we saw that this requires solving a saddle point system of the form

$$\begin{bmatrix} \beta Q_u & 0 & -\widehat{Q}^T \\ 0 & Q_y & K \\ -\widehat{Q} & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{5.1}$$

We now want to apply the theory developed in Chapters 3 and 4 to develop fast iterative solvers for systems of this type. The matrix here is a saddle point matrix of the form $\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}$, where

$$A = \begin{bmatrix} \beta Q_u & 0 \\ 0 & Q_y \end{bmatrix}, \quad B = \begin{bmatrix} -\widehat{Q} & K \end{bmatrix}.$$

The theory in Chapter 4 tells us that good preconditioners for saddle point systems such as this generally rely upon good approximations for A and $S = BA^{-1}B^T$, the Schur complement. We introduce some model problems in the next section. Sections 5.2 and 5.3 look at some approaches to making the approximations to A and S respectively in these particular cases. In the remainder of the chapter we apply these approximations to some of the techniques described in Chapter 4 for the model problems.

5.1 Model Problems

Below we introduce some example problems which we will use in the later sections of this chapter to test the effectiveness of our methods. The first two examples are distributed control problems with Dirichlet boundary conditions, but different desired states, $\hat{y}$. The next two problems are also distributed control problems with the same objective function as Example 5.1.1, but with Neumann and mixed boundary conditions respectively. The final problem is a boundary control problem.

Example 5.1.1. *Let $\Omega = [0,1]^m$, where $m = 2, 3$, and consider the problem*

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||^2_{L_2(\Omega)} + \frac{\beta}{2}||u||^2_{L_2(\Omega)}$$

$$\begin{aligned} \text{s.t.} \qquad -\nabla^2 y &= u \qquad \text{in } \Omega \qquad (5.2) \\ y &= \hat{y}|_{\partial\Omega} \qquad \text{on } \partial\Omega \qquad (5.3) \end{aligned}$$

where, in 2D,

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

and, in 3D,

$$\hat{y} = \begin{cases} (2x_1 - 1)^2(2x_2 - 1)^2(2x_3 - 1)^2 & \text{if } \mathbf{x} \in [0, \frac{1}{2}]^3 \\ 0 & \text{otherwise} \end{cases}$$

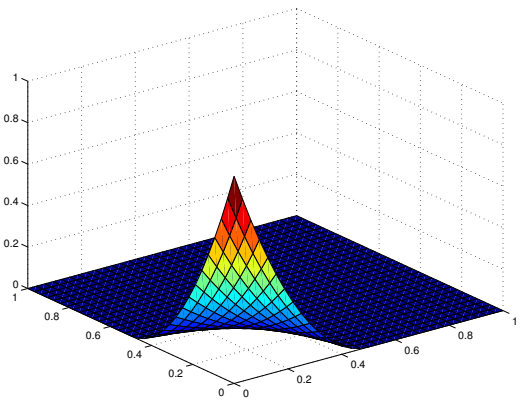
i.e. $\hat{y}$ is bi- or tri-quadratic (depending on whether $m = 2$ or 3) with a peak of unit height at the origin and is zero outside $[0, \frac{1}{2}]^m$.

A plot of the desired state for Example 5.1.1, together with the computed state and control for $\beta = 10^{-2}$ and $\beta = 10^{-5}$ are given in Figure 5.1. Note that with a mesh size of $h = 2^{-5}$, the cost functional is 7.87×10^{-4} when $\beta = 10^{-2}$ and 2.67×10^{-5} when $\beta = 10^{-5}$.

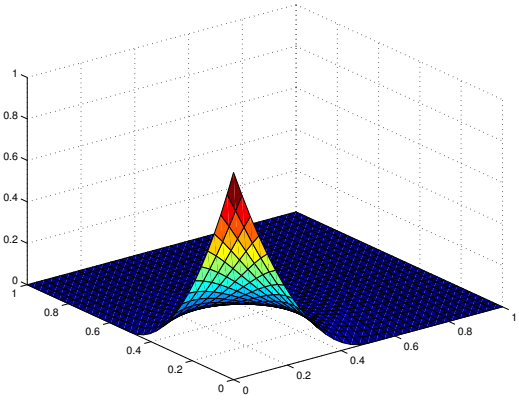
Example 5.1.2. *Let $\Omega = [0,1]^m$, where $m = 2, 3$, and consider the problem*

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||^2_{L_2(\Omega)} + \frac{\beta}{2}||u||^2_{L_2(\Omega)}$$

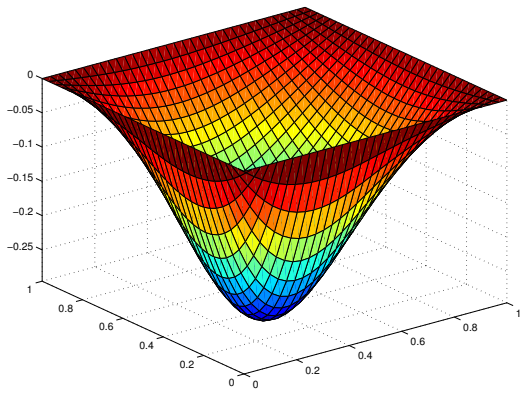
$$\begin{aligned} \text{s.t.} \qquad -\nabla^2 y &= u \qquad \text{in } \Omega \qquad (5.4) \\ y &= \hat{y}|_{\partial\Omega} \qquad \text{on } \partial\Omega \qquad (5.5) \end{aligned}$$



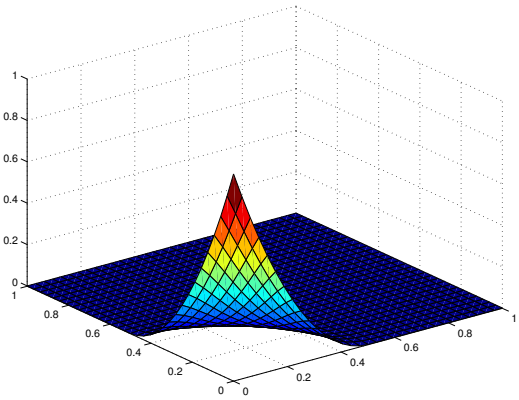
(a) Desired state, $\hat{y}$



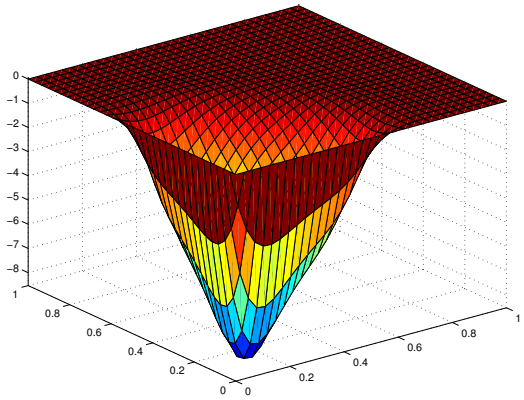
(b) Computed state, y, for $\beta = 10^{-2}$



(c) Computed control, u, for $\beta = 10^{-2}$



(d) Computed state, y, for $\beta = 10^{-5}$



(e) Computed control, u, for $\beta = 10^{-5}$

Figure 5.1: Desired state, state and control for Example 5.1.1 in two dimensions, $\beta = 10^{-2}$ and $\beta = 10^{-5}$.

where, in 2D,

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{x} \in [0, \frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases}$$

and, in 3D,

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{x} \in [0, \frac{1}{2}]^3 \\ 0 & \text{otherwise} \end{cases} .$$

Again, we plot the desired state for Example 5.1.2 along with the computed state and control for $\beta = 10^{-2}$ and $\beta = 10^{-5}$, in Figure 5.2. Here the cost functional is 2.51×10^{-2} when $\beta = 10^{-2}$ and 7.47×10^{-3} when $\beta = 10^{-5}$, both values given using a discretization of $h = 2^{-5}$. Note that the theory in section 2.2 tells us that an optimal control doesn't exist in the case $\beta = 0$ for this problem. This can be seen here by the fact that as β gets closer to zero a singularity develops in the control along the lines where the desired state is discontinuous; we start to see this effect in Figure 5.2(e).

Examples 5.1.3 and 5.1.4 take the same objective function as 5.1.1, but have Neumann and mixed boundary conditions respectively.

Example 5.1.3. *Let $\Omega = [0,1]^2$ and consider the Neumann problem*

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||^2_{L_2(\Omega)} + \frac{\beta}{2}||u||^2_{L_2(\Omega)}$$

$$\text{s.t.} \qquad -\nabla^2 y = u \qquad \text{in } \Omega \tag{5.6}$$

$$\frac{\partial y}{\partial n} = 0 \qquad \text{on } \partial\Omega \tag{5.7}$$

where

$$\hat{y} = \begin{cases} (2x_1 - 1)^2(2x_2 - 1)^2 & \text{if } \mathbf{x} \in [0, \frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases} .$$

Example 5.1.4. *Let $\Omega = [0,1]^2$ and consider the problem*

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||^2_{L_2(\Omega)} + \frac{\beta}{2}||u||^2_{L_2(\Omega)}$$

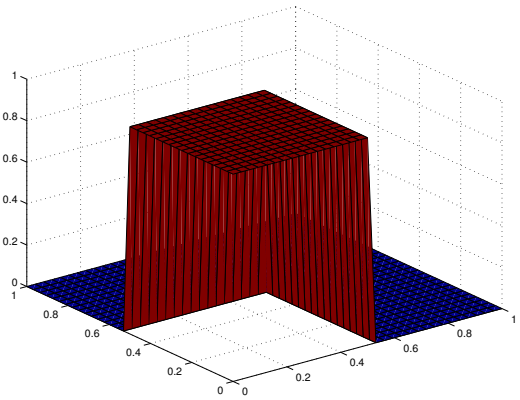
$$\text{s.t.} \qquad -\nabla^2 y = u \qquad \text{in } \Omega \tag{5.8}$$

$$y = \hat{y}|_{\partial\Omega} \text{ on } \partial\Omega_1 \qquad \text{and} \qquad \frac{\partial y}{\partial n} = 0 \text{ on } \partial\Omega_2 \tag{5.9}$$

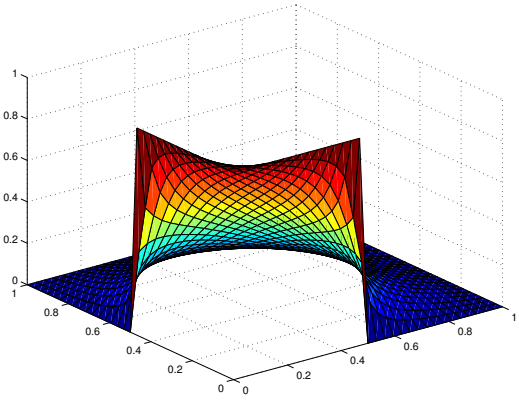
where

$$\hat{y} = \begin{cases} (2x_1 - 1)^2(2x_1 - 1)^2 & \text{if } \mathbf{x} \in [0, \frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases}$$

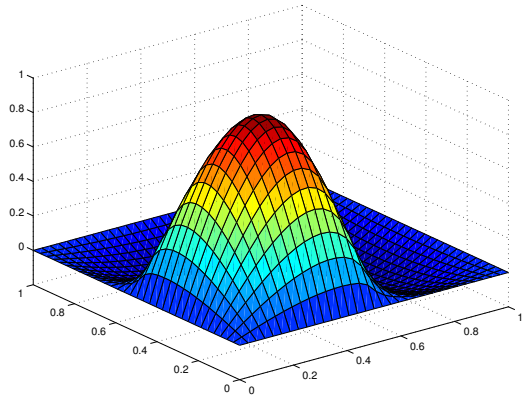
and $\partial\Omega_1 = (0 \times [0,1)) \cup ((0,1] \times 0)$ and $\partial\Omega_2 = (1 \times (0,1]) \cup ([0,1) \times 1)$.



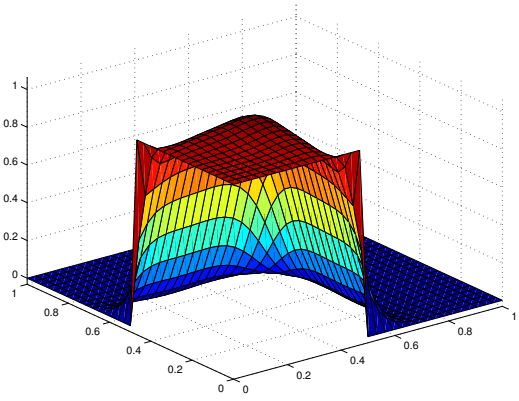
(a) Desired state, $\hat{y}$



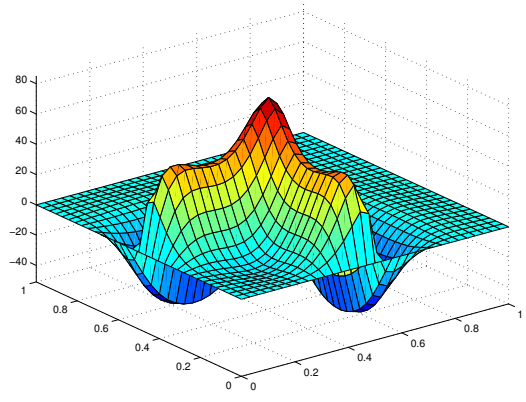
(b) Computed state, y, for $\beta = 10^{-2}$



(c) Computed control, u, for $\beta = 10^{-2}$



(d) Computed state, y, for $\beta = 10^{-5}$



(e) Computed control, u, for $\beta = 10^{-5}$

Figure 5.2: Desired state, state and control for Example 5.1.2 in two dimensions, $\beta = 10^{-2}$ and $\beta = 10^{-5}$.

Finally, Example 5.1.5 is a boundary control problem, with a Neumann boundary condition.

Example 5.1.5. *Let* $\Omega = [0,1]^2$ *and consider the boundary control problem*

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||^2_{L_2(\Omega)} + \frac{\beta}{2}||u||^2_{L_2(\partial\Omega)}$$

$$\begin{aligned} \text{s.t.} \qquad -\nabla^2 y &= 0 \text{ in } \Omega & (5.10)\\ \frac{\partial y}{\partial n} &= u \text{ on } \partial\Omega & (5.11) \end{aligned}$$

where

$$\hat{y} = \begin{cases} (2x_1 - 1)^2(2x_2 - 1)^2 & \text{if } \mathbf{x} \in [0, \frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases}.$$

5.2 Approximating the mass matrix

First, consider $A = \begin{bmatrix} \beta Q_u & 0 \\ 0 & Q_y \end{bmatrix}$. Solving exactly with A is expensive, so we need to obtain good, yet inexpensive, approximations to the action of the inverse of this matrix. Since A is block diagonal, and composed of mass matrices, it is sufficient to look for an approximation to the action of the inverse of a mass matrix, Q.

Consider $D = \text{diag}(Q)$. Wathen [125] gave a method to calculate constants θ, Θ (which depend on the type of elements used, but not the mesh size) such that

$$\theta \leq \lambda(D^{-1}Q) \leq \Theta. \tag{5.12}$$

Recall from (2.6) that we can write Q in terms of the element mass matrices as

$$Q = L^T \text{blkdiag}(Q_e) L.$$

We can do the same thing with the diagonal of the mass matrix, D; here

$$D = L^T \text{blkdiag}(D_e) L,$$

where $D_e = \text{diag}(Q_e)$. Now, since Q and D are symmetric positive definite, consider the generalized Rayleigh quotient

$$\frac{\mathbf{x}^T Q \mathbf{x}}{\mathbf{x}^T D \mathbf{x}} = \frac{\mathbf{x}^T L^T \text{blkdiag}(Q_e) L\mathbf{x}}{\mathbf{x}^T L^T \text{blkdiag}(D_e) L\mathbf{x}} = \frac{\mathbf{y}^T \text{blkdiag}(Q_e)\mathbf{y}}{\mathbf{y}^T \text{blkdiag}(D_e)\mathbf{y}},$$

where $\mathbf{y} = L\mathbf{x}$. The maximum (resp. minimum) value of the generalized Rayleigh quotient is the same as the maximum (resp. minimum) eigenvalue of $D_e^{-1}Q_e$ over

all the elements, therefore we can obtain the bound Θ (resp. θ) by this method. If the mesh is regular then all the element matrices are the same, so we only need to perform the calculation once to get eigenvalue bounds. These values are tabulated in [125] for some common elements, and we reproduce the bounds here in Table 5.1. Figure 5.3 shows the eigenvalues of $D^{-1}Q$ for a $\mathbf{Q}_1$ discretization in two dimensions, along with the predicted bounds. Note that if we have an irregular mesh then it is just an $\mathcal{O}(N)$ calculation to obtain the maximum and minimum eigenvalues for each element.

Elements	θ	Θ
Arbitrary linear (1D)	1/2	3/2
Arbitrary linear triangles ($\mathbf{P}_1$)	1/2	2
Bilinear rectangles ($\mathbf{Q}_1$)	1/4	9/4
Arbitrary quadratic triangles ($\mathbf{Q}_2$)	0.3924	2.0598
Biquadratic rectangles ($\mathbf{Q}_2$)	1/4	25/16
Rectangular trilinear brick ($\mathbf{Q}_1$)	1/8	27/8

Table 5.1: Eigenvalue bounds for $D^{-1}Q$, from Table 1 in [125]

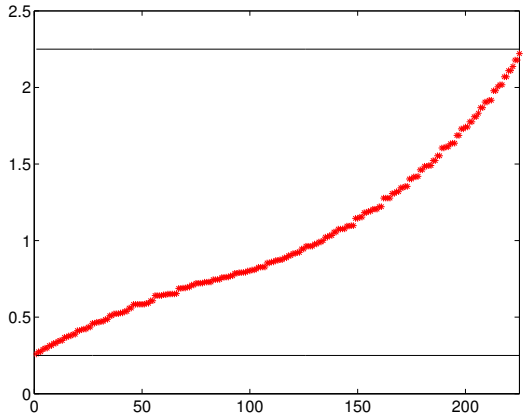


Figure 5.3: Distribution of eigenvalues in two-dimensions for a $\mathbf{Q}_1$ discretization with $h = 2^{-5}$ ('*'), and the bounds predicted by Table 5.1 ('-').

The results above show us that simply replacing Q by $\text{diag}(Q)$ in A would be a reasonable approximation, at least for standard finite elements. Is it possible to do any better? The original paper derives these bounds to show that the diagonal of the mass matrix is a good preconditioner for conjugate gradients, so a number of steps of the conjugate gradient method would be a good approximation to action of the inverse of Q. However, as discussed in Chapter 3, conjugate gradients is a non-linear method, so cannot be used as part of a preconditioner for a standard – i.e. not flexible

– Krylov subspace method. Recall from Section 3.1.5 that a non-linear method like conjugate gradients gives superlinear convergence since it automatically picks the best polynomial approximation to the eigenvalues, and hence can take advantage of any clustering. Figure 5.3 shows that, in the case of $D^{-1}Q$, the eigenvalues are quite uniformly distributed between the two extrema, so we should expect close to linear convergence – i.e. the convergence will be close to that described by the usually pessimistic bound (3.27).

Recall that the bound (3.27) has the same constant as the bound in the Chebyshev semi-iteration (3.22), which was the (linear) method from which we derived conjugate gradients in Chapter 3. This suggests that the Chebyshev semi-iteration shouldn't be too much worse than conjugate gradients in this case. Since it is a linear method, it is also suitable for use as a preconditioner for a Krylov subspace method.

As described in Section 3.1.4, the drawback with this method is that we require accurate bounds for the eigenvalues of the iteration matrix for the underlying simple iteration. Suppose the underlying method is relaxed Jacobi, as defined in Section 3.1.2. The iteration matrix is then $I - \omega D^{-1}Q$, where ω is the relaxation parameter. Equation (3.7) tells us that, if the bounds in Table 5.1 are tight, the optimal value of ω is

$$\omega = \frac{2}{\theta + \Theta}.$$

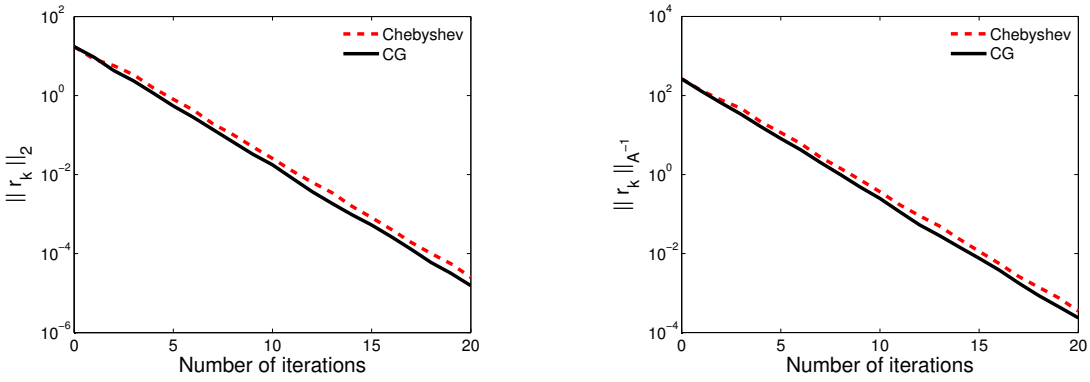
Therefore, again using (5.12), we see that if we use this optimal value of ω the eigenvalues of the iteration matrix satisfy

$$\frac{\theta - \Theta}{\theta + \Theta} \leq \lambda(I - \omega D^{-1}Q) \leq \frac{\Theta - \theta}{\theta + \Theta}.$$

Hence we have good bounds for the eigenvalues of the iteration matrix, and so the Chebyshev semi-iteration is a viable method when solving a system with a mass matrix. Figure 5.4 shows convergence curves for the Chebyshev semi-iterative method and the preconditioned conjugate gradient method, plotting convergence in two different norms. This supports our prediction that not much is lost by choosing the Chebyshev semi-iteration over the conjugate gradient method.

We now want to calculate eigenvalue bounds for the system preconditioned by the Chebyshev semi-iteration. Suppose we want to solve the system $Q\mathbf{x} = \mathbf{b}$. If we take an initial guess $\mathbf{x}^{(0)} = \mathbf{0}$ in the underlying relaxed Jacobi method, equation (3.17) tells us that the Chebyshev semi-iteration satisfies

$$\mathbf{x} - \mathbf{y}^{(k)} = p_k(S)\mathbf{x},$$



(a) The residuals in the 2-norm

(b) The residuals in the $\mathcal{A}^{-1}$-norm

Figure 5.4: Comparison of convergence of PCG and Chebyshev semi-iteration.

where $S := I - \omega D^{-1}Q$ and $p_k(x)$ is the appropriate scaled and shifted Chebyshev polynomial, as described in Section 3.1.4. On rearranging, this becomes

$$\mathbf{y}^{(k)} = (I - p_k(S))\mathbf{x} = (I - p_k(S))M^{-1}\mathbf{b}.$$

Therefore we can write the k^{th} iterate of the Chebyshev semi-iteration as a matrix. Let us define $C_k^{-1} := (I - p_k(S))M^{-1}$; then C_k is our approximation to the mass matrix.

In order to see how effective this approximation is we can look at the eigenvalues of $C_k^{-1}M = I - p_k(S)$. Since we know the polynomial $p_k(x)$ explicitly (3.19) these eigenvalues are easy to calculate. For $\mathbf{Q}_1$ elements we give the upper and lower bounds on the eigenvalues for the first twenty iterations in two and three dimensions in Table 5.2. The table shows that this method quickly gives a very accurate approximation to the solution, and – as we can see from Algorithm 4 – the Chebyshev semi-iteration is very cheap to implement. This is therefore an effective preconditioner, and we have shown, for a given k, there exist constants δ_k and Δ_k independent of h such that

$$\delta_k \leq \frac{\mathbf{x}^T Q \mathbf{x}}{\mathbf{x}^T C_k \mathbf{x}} \leq \Delta_k. \tag{5.13}$$

Hence, returning to the control problem, if we define $A_0 = \begin{bmatrix} \beta C_k^u & 0 \\ 0 & C_k^y \end{bmatrix}$, where the superscripts distinguish between the triangulations used to discretize the control and the state, then we have

$$\min\left(\delta_k^u, \delta_k^y\right) \leq \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T A_0 \mathbf{x}} \leq \max\left(\Delta_k^u, \Delta_k^y\right). \tag{5.14}$$

	Two Dimensions			Three Dimensions		
k	Lower bound	Upper bound	ϵ_k	Lower bound	Upper bound	ϵ_k
1	0.200000000000000	1.800000000000000	0.800000000000	0.071428571428571	1.928571428571429	0.928571428571
2	0.529411764705882	1.470588235294118	0.470588235294	0.242152466367712	1.757847533632288	0.757847533632
3	0.753846153846154	1.246153846153846	0.246153846153	0.433470861268694	1.566529138731306	0.566529138731
4	0.875486381322957	1.124513618677043	0.124513618677	0.597147975231673	1.402852024768327	0.402852024768
5	0.937560975609756	1.062439024390244	0.062439024390	0.720776486124292	1.279223513875708	0.279223513875
6	0.968757627532341	1.031242372467659	0.031242372467	0.808846507572246	1.191153492427754	0.191153492427
7	0.984375953616112	1.015624046383888	0.015624046383	0.869897818711463	1.130102181288537	0.130102181288
8	0.992187619207471	1.007812380792529	0.007812380792	0.911689150016541	1.088310849983459	0.088310849983
9	0.996093764901104	1.003906235098896	0.003906235098	0.940130893927575	1.059869106072427	0.059869106072
10	0.998046876862643	1.001953123137357	0.001953123137	0.959435805298037	1.040564194701956	0.040564194701
11	0.999023437732830	1.000976562267170	0.000976562267	0.972523033141943	1.027476966858060	0.027476966858
12	0.999511718779104	1.000488281220896	0.000488281220	0.981390172808929	1.018609827191073	0.018609827191
13	0.999755859378638	1.000244140621363	0.000244140621	0.987396479797611	1.012603520202398	0.012603520202
14	0.999877929687954	1.000122070312045	0.000122070312	0.991464472578415	1.008535527421550	0.008535527421
15	0.999938964843807	1.000061035156194	0.000061035156	0.994219521276648	1.005780478723391	0.005780478723
16	0.999969482421881	1.000030517578119	0.000030517578	0.996085332018970	1.003914667981061	0.003914667981
17	0.999984741210937	1.000015258789063	0.000015258789	0.997348906791884	1.002651093208068	0.002651093208
18	0.999992370605466	1.000007629394531	0.000007629394	0.998204627482791	1.001795372516598	0.001795372517
19	0.999996185302733	1.000003814697268	0.000003814697	0.998784139008908	1.001215860991333	0.001215860991
20	0.999998092651366	1.000001907348635	0.000001907348	0.999176595616630	1.000823404383702	0.000823404383

Table 5.2: Upper and lower bounds for $\lambda(C_k^{-1}M)$, and ϵ_k s.t. $|1-\lambda_k| \leq \epsilon_k$, where C_k^{-1} is given by k steps of the Chebyshev semi-iteration (without scaling) for square Q1 elements

Using Chebyshev semi-iteration as a preconditioner is an old idea – see, for example, [5, 50, 73, 90, 99]. The first person to suggest using it to precondition a solve with a mass matrix, as described above, appears to be van Gijzen in 1995 [122]. The idea was discovered independently, and given a more theoretical understanding, by Wathen and Rees in 2008 [126]. This idea is useful in many contexts, not just problems in the optimal control of PDEs. For example, when using standard preconditioners for the Stokes equation (see Chapter 7) one of the blocks in an ideal preconditioner is a mass matrix. This method was described in this context by Wathen and Rees [126] and has since been used by Grinevich and Olshanskii [55].

5.3 Approximating the Schur Complement

Now consider the Schur complement. For the saddle point system we're considering here, (5.1),

$$BA^{-1}B^T = \frac{1}{\beta}\widehat{Q}Q_u^{-1}\widehat{Q}^T + KQ_y^{-1}K.$$

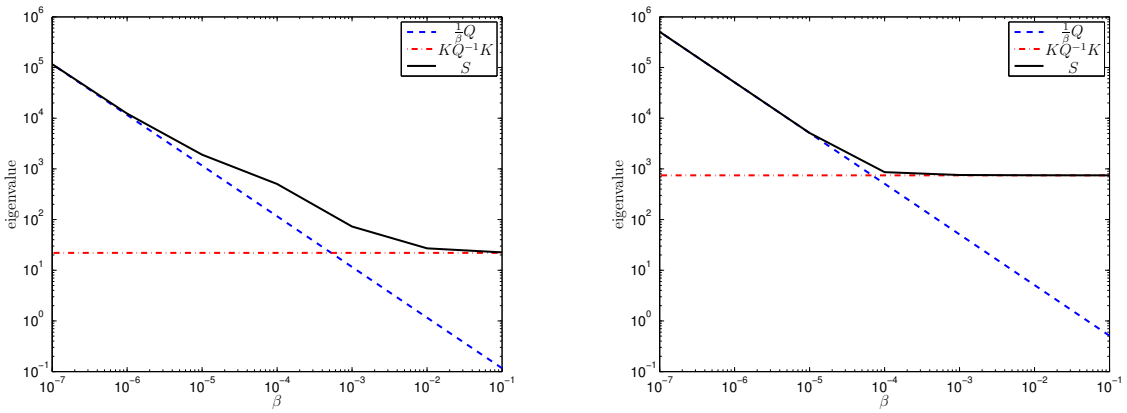
The first difficulty in applying the inverse of this is the addition. Figure 5.5 shows the largest and smallest eigenvalues for different values of β for Example 5.1.1, in the case where u and y are both discretized with uniform $\mathbf{Q}_1$ elements. This suggests that for all but the smallest values of β the dominant term in the sum is the $KQ_y^{-1}K$ term. A thorough analysis of how the condition number of the matrix changes with β has been given by Thorne [114].

Recall equation (2.27) tells us that, for this problem, we have that the error satisfies

$$\|u - u_h\|_2 \leq Ch^2/\beta,$$

so if we take a value of β smaller than h^2 we cannot guarantee the accuracy of the solution. This suggests that the 'larger' values of β are the ones that are important. The value of $\beta = 10^{-2}$ has often been used in the literature – see, e.g. [29, 56, 71]. S. Ulbrich [68, p. 6] suggests that $\beta \in [10^{-5}, 10^{-3}]$. We therefore develop preconditioners that are effective for these 'larger' β.

Let us define $\widehat{S} := KQ_y^{-1}K$. We want to make the intuition above more precise by considering the eigenvalues of $\widehat{S}^{-1}S$. Consider the case where both y and u are discretized using the same elements, so $Q_u = Q_y = \widehat{Q} := Q$. Then $S = \frac{1}{\beta}Q + KQ^{-1}K$,



(a) Minimum eigenvalue

(b) Maximum eigenvalue

Figure 5.5: Extremal eigenvalues vs β where the control and the state are both discretized using $\mathbf{Q}_1$ elements – $h = 2^{-2}$.

and so

$$\begin{aligned} S\mathbf{x} &= \lambda \widehat{S}\mathbf{x} \\ (\frac{1}{\beta}Q + KQ^{-1}K)\mathbf{x} &= \lambda KQ^{-1}K\mathbf{x} \\ (K^{-1}QK^{-1}Q + \beta I)\mathbf{x} &= \beta\lambda\mathbf{x} \\ (K^{-1}Q)^2\mathbf{x} &= \beta(\lambda - 1)\mathbf{x}. \end{aligned}$$

Since $K^{-1}Q$ – and hence $(K^{-1}Q)^2$ – is positive definite, let $\beta(\lambda - 1) = \nu^2$. Then

$$Q\mathbf{x} = \nu K\mathbf{x},$$

and so

$$\nu = \frac{\mathbf{x}^T Q\mathbf{x}}{\mathbf{x}^T K\mathbf{x}} = \frac{\mathbf{x}^T Q\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \cdot \frac{\mathbf{x}^T\mathbf{x}}{\mathbf{x}^T K\mathbf{x}}.$$

Assuming our triangulation satisfies the conditions for Theorem 2.1.1 and Theorem 2.1.2, appealing to these immediately gives us that there exists constants c, C independent of the mesh size such that

$$\sqrt{c}h^2 \leq \frac{\mathbf{x}^T Q\mathbf{x}}{\mathbf{x}^T K\mathbf{x}} \leq \sqrt{C}.$$

Therefore we have

$$ch^4 \leq \beta(\lambda - 1) \leq C \Rightarrow \frac{1}{\beta}ch^4 + 1 \leq \lambda \leq \frac{1}{\beta}C + 1.$$

We have shown that there exists constants c and C independent of the mesh size such that

$$\frac{1}{\beta}ch^4 + 1 \leq \frac{\mathbf{x}^T S\mathbf{x}}{\mathbf{x}^T \widehat{S}\mathbf{x}} \leq \frac{1}{\beta}C + 1. \tag{5.15}$$

Therefore, as long as β isn't too small, these eigenvalues will be clustered, and $\widehat{S}$ is a good approximation to S.

We have shown that, at least for large values of β, $\widehat{S}$ is a good approximation to S. However, a solve with $\widehat{S}$ requires a matrix-vector multiply with Q_y – which is relatively cheap – and two solves with K – which is expensive. Therefore $\widehat{S}$ is not itself a practical preconditioner, and so we need to approximate this again to get an efficient approximation to the Schur complement.

The difficulty is that we have to solve two systems of the form

$$K\mathbf{x} = \mathbf{b}. \tag{5.16}$$

We know from Section 3.2 that multigrid, for example, is a good preconditioner for systems of this type, so what if we replace K by $\tilde{K}$, where $\tilde{K}$ approximates K in the sense that there exist positive constants $\gamma \leq 1$ and $\Gamma \geq 1$ such that

$$\gamma \leq \frac{\mathbf{x}^T K\mathbf{x}}{\mathbf{x}^T \tilde{K}\mathbf{x}} \leq \Gamma \tag{5.17}$$

for all $\mathbf{x}$. Then is $\tilde{K}Q_y^{-1}\tilde{K}$ a good approximation for $KQ_y^{-1}K$? We have that

$$\frac{\mathbf{x}^T KQ_y^{-1}K\mathbf{x}}{\mathbf{x}^T \tilde{K}Q_y^{-1}\tilde{K}\mathbf{x}} = \frac{\mathbf{y}^T Q_y^{-1}\mathbf{y}}{\mathbf{y}^T\mathbf{y}} \cdot \frac{\mathbf{x}^T K^2\mathbf{x}}{\mathbf{x}^T \tilde{K}^2\mathbf{x}} \cdot \frac{\mathbf{z}^T\mathbf{z}}{\mathbf{z}^T Q_y^{-1}\mathbf{z}},$$

where $\mathbf{y} = K\mathbf{x}$ and $\mathbf{z} = K^T\mathbf{x}$. Applying Theorem 2.1.2 we have that there exist constants $c^* \leq 1$ and $C^* \geq 1$ such that

$$c^* \inf_{\mathbf{x}} \frac{\mathbf{x}^T K^2\mathbf{x}}{\mathbf{x}^T \tilde{K}^2\mathbf{x}} \leq \frac{\mathbf{x}^T KQ_y^{-1}K\mathbf{x}}{\mathbf{x}^T \tilde{K}Q_y^{-1}\tilde{K}\mathbf{x}} \leq C^* \sup_{\mathbf{x}} \frac{\mathbf{x}^T K^2\mathbf{x}}{\mathbf{x}^T \tilde{K}^2\mathbf{x}}. \tag{5.18}$$

Note that we can write the first inequality of (5.17) as $\gamma\mathbf{x}^T\tilde{K}\mathbf{x} \leq \mathbf{x}^T K\mathbf{x}$, or equivalently:

$$\mathbf{x}^T P_\gamma \mathbf{x} \leq \mathbf{x}^T K\mathbf{x},$$

where $P_\gamma := \gamma\tilde{K}$. We can similarly write the second inequality as

$$\mathbf{x}^T K\mathbf{x} \leq \mathbf{x}^T P_\Gamma \mathbf{x},$$

where $P_\Gamma := \Gamma K$. Therefore, if we know that $P_\gamma \leq K \Rightarrow P_\gamma^2 \leq K^2$ (and similarly $K \leq P_\Gamma \Rightarrow K^2 \leq P_\Gamma^2$), then we would necessarily have

$$\gamma^2 \leq \frac{\mathbf{x}^T K^2\mathbf{x}}{\mathbf{x}^T \tilde{K}^2\mathbf{x}} \leq \Gamma^2,$$

and hence, from (5.18), we see that $\tilde{K}Q_y^{-1}\tilde{K}$ is an effective preconditioner for $KQ_y^{-1}K$. Note that this is a sufficient, but not a necessary, condition.

We now have to address a further question: under what conditions does $B \leq C \Rightarrow B^2 \leq C^2$? If B and C are positive matrices which commute, then $C - B$ is positive, and $C(C - B)$ is a product of two positive matrices which commute, and is hence positive [69, p. 434]. The same is true for $(C - B)B$. Therefore

$$B^2 \leq CB \leq C^2,$$

and so the relationship holds in this case. As shown by the following example by Braess and Peisker [18], this is not true in general. Define matrices B and C by

$$B := \begin{bmatrix} 1 & a \\ a & 2a^2 \end{bmatrix}, \qquad C := \begin{bmatrix} 2 & 0 \\ 0 & 3a^2 \end{bmatrix},$$

where $a \gg 1$. Then the eigenvalues of $C - B$ are 0 and $1 + a^2$, so $B \leq C$. However,

$$B^2 = \begin{bmatrix} 1 + a^2 & a + 2a^3 \\ a + 2a^3 & a^2 + 4a^4 \end{bmatrix}, \qquad C^2 = \begin{bmatrix} 4 & 0 \\ 0 & 9a^4 \end{bmatrix},$$

and so the eigenvalues of $C^2 - B^2$ are

$$\lambda = \frac{1}{2}\left(5a^4 - 2a^2 + 3 \pm \sqrt{25a^8 + 16a^6 - 14a^4 + 4a^2 + 9}\right),$$

of which the smallest is plotted in Figure 5.6. From the figure we see that $C^2 - B^2$ is indefinite for $a \geq 1$.

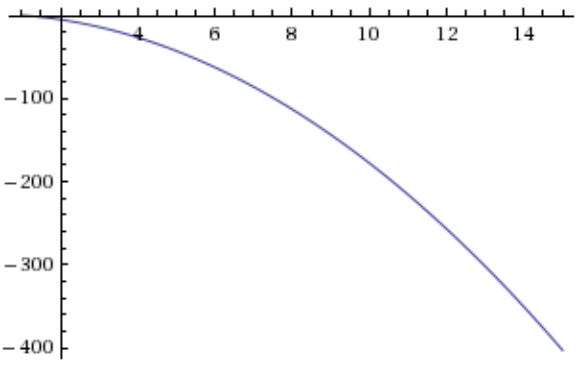


Figure 5.6: Plot of smallest eigenvalue of $C^2 - B^2$, $a = [1, 15]$

This shows that the relation $B \leq C \Rightarrow B^2 \leq C^2$ does not hold for all matrices B, C. We know the relationship holds if B and C commute. This is not a necessary condition, as can be seen by considering the matrices

$$B = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}, \qquad C = \begin{bmatrix} 10 & 1 \\ 1 & 20 \end{bmatrix}.$$

Here the eigenvalues of $C - B$ are 7.89 and 17.10, so $B < C$. The eigenvalues of $C^2 - B^2$ are also positive, namely 93.98 and 395.01, so $B^2 < C^2$. Here B and C clearly do not commute. Therefore commutivity of B and C is not the only property that determines whether $B^2 \leq C^2$.

Let $\widetilde{K}_m^{-1}$ denote the operation of applying to each $\mathbf{b}$ an approximation $\mathbf{x}_m = \widetilde{K}_m^{-1}\mathbf{b}$ to the solution of (5.16) by applying m steps of a convergent (multigrid) iteration with starting value $\mathbf{0}$. We would like to show the existence positive constants ω and Ω such that

$$\omega \leq \frac{\mathbf{x}^T K^2 \mathbf{x}}{\mathbf{x}^T \tilde{K}_m^2 \mathbf{x}} \leq \Omega. \tag{5.19}$$

By definition of convergence, there exists a sequence $\{\eta_m\}$ with $\eta_m \to 0$ such that

$$\frac{\|\mathbf{x}_m - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \eta_m.$$

This can be written as

$$\|\widetilde{K}_m^{-1}\mathbf{b} - K^{-1}\mathbf{b}\| \leq \eta_m \|K^{-1}\mathbf{b}\|. \tag{5.20}$$

Note that

$$\begin{aligned}
\eta_m &= \sup_{\mathbf{b}\neq 0} \frac{\|\widetilde{K}_m^{-1}\mathbf{b} - K^{-1}\mathbf{b}\|}{\|K^{-1}\mathbf{b}\|} = \sup_{\mathbf{x}\neq 0} \frac{\|\widetilde{K}_m^{-1}K\mathbf{x} - \mathbf{x}\|}{\|\mathbf{x}\|} \\
&= \|\widetilde{K}_m^{-1}K - I\| = \|(\widetilde{K}_m^{-1}K - I)^T\| \\
&= \|K\widetilde{K}_m^{-T} - I\|,
\end{aligned}$$

which means we also have the following equation, which involves the adjoint of $\widetilde{K}_m^{-1}$:

$$\|K\widetilde{K}_m^{-T}\mathbf{b} - \mathbf{b}\| \leq \eta_m \|\mathbf{b}\|. \tag{5.21}$$

From (5.21) we have that

$$(1 - \eta_m)\|\mathbf{b}\| \leq \|K\widetilde{K}_m^{-T}\mathbf{b}\| \leq (1 + \eta_m)\|\mathbf{b}\|,$$

and squaring this gives

$$(1 - \eta_m)^2\|\mathbf{b}\|^2 \leq \|K\widetilde{K}_m^{-T}\mathbf{b}\|^2 \leq (1 + \eta_m)^2\|\mathbf{b}\|^2.$$

Therefore we have that

$$(1 - \eta_m)^2\mathbf{b}^T\mathbf{b} \leq \mathbf{b}^T\widetilde{K}_m^{-1}K^2\widetilde{K}_m^{-T}\mathbf{b} \leq (1 + \eta_m)^2\mathbf{b}^T\mathbf{b},$$

and, finally, taking $\mathbf{x} = \widetilde{K}_m^{-T}\mathbf{b}$ we get

$$(1-\eta_m)^2\mathbf{x}^T\widetilde{K}_m\widetilde{K}_m^T\mathbf{x} \leq \mathbf{x}^T K^2\mathbf{x} \leq (1+\eta_m)^2\mathbf{x}^T\widetilde{K}_m\widetilde{K}_m^T\mathbf{x}.$$

We can therefore write

$$(1-\eta_m)^2 \leq \frac{\mathbf{x}^T K^2\mathbf{x}}{\mathbf{x}^T\widetilde{K}_m\widetilde{K}_m^T\mathbf{x}} \leq (1+\eta_m)^2,$$

and since here $\widetilde{K}_m$ is symmetric, this is the same form as (5.19), with $\omega = (1-\eta_m)^2$ and $\Omega = (1+\eta_m)^2$. Hence, from (5.18),

$$c^*(1-\eta_m)^2 \leq \frac{\mathbf{x}^T K Q_y^{-1} K\mathbf{x}}{\mathbf{x}^T\tilde{K}Q_y^{-1}\tilde{K}\mathbf{x}} \leq C^*(1+\eta_m)^2. \tag{5.22}$$

The argument above was first given by Braess and Peisker [18], and shows that any (linear) convergent iterative procedure that converges to the solution $\mathbf{x}$ of $K\mathbf{x} = \mathbf{y}$ – along with the procedure that corresponds to its transpose – can be used as a preconditioner for K^2. We can take multigrid as the iteration, and as long as the constant η_m is small enough – i.e. by taking a sufficient number of smoothing steps or V-cycles – we will get a good preconditioner for the square of K complement.

We now return to the case of control the Poisson equation. We want to see how well $\tilde{K}Q_y^{-1}\tilde{K}$ approximates S. We know

$$\frac{\mathbf{x}^T S\mathbf{x}}{\mathbf{x}^T\tilde{K}Q_y^{-1}\tilde{K}\mathbf{x}} = \frac{\mathbf{x}^T S\mathbf{x}}{\mathbf{x}^T K Q_y^{-1} K\mathbf{x}}\frac{\mathbf{x}^T K Q_y^{-1} K\mathbf{x}}{\mathbf{x}^T\tilde{K}Q_y^{-1}\tilde{K}\mathbf{x}},$$

and so simply applying (5.15) and (5.22) gives us

$$c^*(\frac{1}{\beta}ch^4+1)(1-\eta_m)^2 \leq \frac{\mathbf{x}^T S\mathbf{x}}{\mathbf{x}^T\tilde{K}Q_y^{-1}\tilde{K}\mathbf{x}} \leq C^*(\frac{1}{\beta}C+1)(1+\eta_m)^2. \tag{5.23}$$

This shows us that – provided our multigrid approximation to K is good enough – replacing a solve with K in our idealized preconditioner by a fixed number of multigrid cycles should cluster the eigenvalues, and hence be a reasonable approximation to the Schur complement. We shall see how this performs in practice with some numerical examples in the next section.

5.4 Block diagonal preconditioners

Consider the system

$$\begin{bmatrix} \beta Q & 0 & -Q \\ 0 & Q & K \\ -Q & K & 0 \end{bmatrix}\begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}, \tag{5.24}$$

where, for clarity in the following discussion, we have discretized the control and the state using the same finite element basis. Recall from Section 4.2 that one method to solve this system is to use MINRES with a block-diagonal preconditioner of the form

$$\mathcal{P}_{BD} = \begin{bmatrix} \beta Q_0 & 0 & 0 \\ 0 & Q_0 & 0 \\ 0 & 0 & S_0 \end{bmatrix},$$

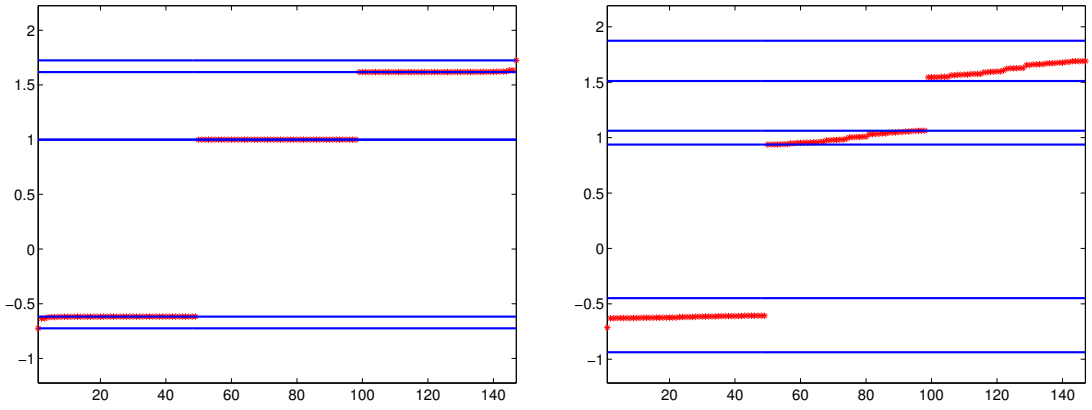
where S_0 approximates the Schur complement and Q_0 approximates Q. In the previous sections we established bounds for the case where $Q_0 = C_k$, where C_k denotes k steps of the Chebyshev semi-iteration, and $S_0 = \tilde{K}Q^{-1}\tilde{K}$, where $\tilde{K}^{-1}$ is a fixed number of multigrid cycles.

We saw in Section 3.1.7 that the speed of convergence of MINRES depends on the clustering of the eigenvalues of the preconditioned system. Appealing to Theorem 4.2.1, together with (5.13) and (5.23), we get that the eigenvalues of the preconditioned system satisfy

$$\begin{aligned} \frac{\delta_k - \sqrt{\delta_k^2 + 4\Delta_k\Phi}}{2} \leq \lambda \leq \frac{\Delta_k - \sqrt{\Delta_k^2 + 4\phi\delta_k}}{2}, \\ \delta_k \leq \lambda \leq \Delta_k, \\ \text{or} \qquad \frac{\delta_k + \sqrt{\delta_k^2 + 4\delta_k\phi}}{2} \leq \lambda \leq \frac{\Delta_k + \sqrt{\Delta_k^2 + 4\Phi\Delta_k}}{2}, \end{aligned}$$

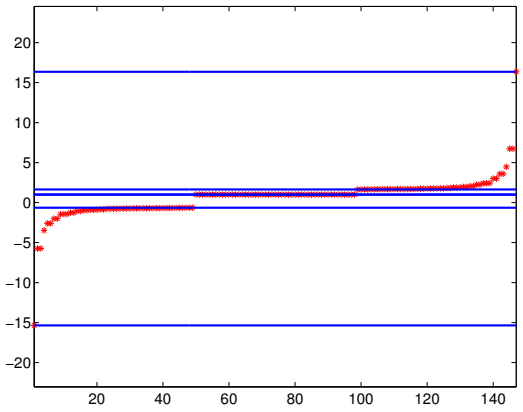
where $\phi = c^*(\frac{1}{\beta}ch^4 + 1)(1 - \eta_m)^2$ and $\Phi = C^*(\frac{1}{\beta}C + 1)(1 + \eta_m)^2$. Improving the accuracy of the approximation to Q or K will make the bounds δ_k and Δ_k closer to unity and η_k closer to zero. Numerical testing has shown that taking one GMG V-cycle with three pre-smoothing iterations for the stiffness matrix along with five steps of the Chebyshev semi-iteration for the mass matrix seems to give a good balance between having a preconditioner that is cheap to solve and yet give good convergence – see Appendix A for more details. We also consider approximating K using an AMG routine – HSL MI20 [15], applied via a MATLAB interface. We treat this as a 'black box', using the default parameters.

Figure 5.7 shows plots of the eigenvalues of the preconditioned system, together with the predicted bounds, for Example 5.1.1 with a mesh size of $h = 2^{-3}$ and for regularization parameters $\beta = 10^{-2}$ and $\beta = 10^{-5}$. We give the eigenvalues in both the cases where solves with the stiffness and mass matrices are exact, and also where they are approximated by one GMG V-cycle and five steps of the Chebyshev semi-iteration respectively. In calculating the bounds we take the contraction rate as $\eta_1 = 0.15$ and get δ, Δ from Table 5.2. As we see from the plots, the bounds above are quite descriptive.

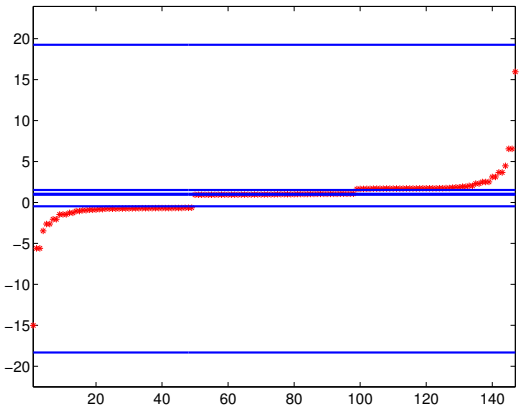


(a) $\beta = 10^{-2}$, exact solves

(b) $\beta = 10^{-2}$, approx solves



(c) $\beta = 10^{-5}$, exact solves



(d) $\beta = 10^{-5}$, approx solves

Figure 5.7: Eigenvalues ('*') and predicted bounds ('-') for the block diagonal preconditioner where $\beta = 10^{-2}, 10^{-5}$, with exact and approximated K and Q solves.

We now consider the examples introduced in Section 5.1, solved with the preconditioner described above. We stop MINRES using the preconditioned residual, $\|\mathbf{r}^{(k)}\|_{\mathcal{P}_{BD}^{-1}}$, which is the quantity that is minimized here – see Section 3.1.7. We take a fixed tolerance of 10^{-6}; it should be noted that, in view of the error estimates in Section 2.1, this is considerable overkill for the smaller values of h, but our main interest here is in the effectiveness of the solution method.

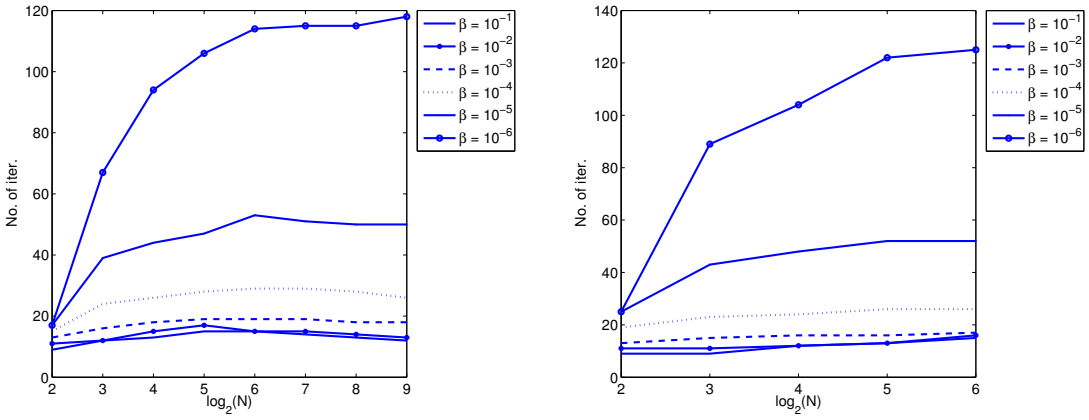
Example 5.1.1 is a distributed control problem with inhomogeneous Dirichlet boundary conditions. Table 5.3 shows results when solving this system with a regularization parameter $\beta = 10^{-2}$, using our block diagonal preconditioner with MINRES. We include the iteration numbers and time taken – in seconds – when using both a GMG and an AMG approximation of K, plus the time taken to solve the whole system using a direct solver. We also include the difference between the approximate solution and the objective function, $\|y_h - \hat{y}\|_2$, the value of the cost functional, $J(y_h, u_h)$ and the difference between the control computed using the iterative method with a GMG approximation to K, $\mathbf{u}^G$, and the control computed using `backslash`, $\mathbf{u}^B$.

Table 5.3: Comparison of solution methods for solving Example 5.1.1 using MINRES

h	GMG		AMG		`backslash`	$\|y_h^G - \hat{y}\|_2$	$J(y_h^G, u_h^G)$	$\|\mathbf{u}^G - \mathbf{u}^B\|_2$
	time	its	time	its	time			
2^{-2}	0.305	11	0.016	11	0.000	1.412e-03	8.103e-04	5.233e-07
2^{-3}	0.301	12	0.010	12	0.001	1.404e-03	7.962e-04	2.051e-06
2^{-4}	0.329	15	0.023	15	0.006	1.396e-03	7.891e-04	2.522e-06
2^{-5}	0.442	17	0.070	16	0.034	1.394e-03	7.871e-04	9.031e-06
2^{-6}	0.806	15	0.271	15	0.196	1.393e-03	7.866e-04	4.281e-05
2^{-7}	2.563	15	1.238	15	1.194	1.393e-03	7.865e-04	4.156e-05
2^{-8}	9.760	14	5.304	14	7.229	1.393e-03	7.864e-04	1.466e-04
2^{-9}	36.698	13	22.596	15	61.416	1.393e-03	7.864e-04	5.951e-04

We see from the results in Table 5.3 that MINRES gives mesh-size independent convergence, and the time taken to solve the system scales linearly with the problem size, which is not true for the direct method. Note that the AMG routine gives faster results than the GMG routine – this is probably because the MI20 AMG is a complied code, as opposed to the interpreted MATLAB code of the geometric multigrid we use; if both were compiled, then GMG should be more efficient. The results above are for $\beta = 10^{-2}$ – Figure 5.8(a) shows a plot of the iteration numbers – using a GMG approximation of K – for a range of β. This shows us that, as predicted by the theory above, the method is fairly robust – albeit with higher iteration numbers as the

parameter gets smaller – for β to around 10^{-5}, but for yet smaller β the convergence is poorer. The MATLAB interface for the MI20 AMG method failed for the largest problem size in three dimensions, so we are unable to report results in this case.



(a) Example 5.1.1 (b) Example 5.1.1 in 3D

Figure 5.8: Plot of problem size vs MINRES iterations needed for different β for a distributed control problem with Dirichlet boundary conditions, in two and three dimensions.

Table 5.4: Comparison of solution methods for solving Example 5.1.1 in 3D using MINRES

h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	0.133	10	0.022	10	0.013
2^{-3}	0.138	9	0.055	9	0.108
2^{-4}	0.597	8	0.777	8	6.572
2^{-5}	4.810	7	8.716	7	—[a]
2^{-6}	43.176	7	—[b]		—[a]

[a] Ran out of memory
[b] AMG failed

Table 5.5 shows results for the distributed control problem described in Example 5.1.2. This is an objective function that is harder be obtain, as we see in the value of the cost functional, which is two orders of magnitude larger here than in the previous case. The results here are analogous to those in Table 5.3 – this suggests that the effectiveness of the method is independent of the problem considered.

The results in Table 5.5 are again for $\beta = 10^{-2}$ – Figure 5.9(a) shows the iteration count for a range of β. The results here are again similar to those for Example 5.1.1.

We now consider Example 5.1.3, which is again a distributed control problem, but with Neumann boundary conditions. Since the discrete Neumann Laplacian matrix

Table 5.5: Comparison of solution methods for solving Example 5.1.2 using MINRES

h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	0.314	11	0.018	11	0.000
2^{-3}	0.306	12	0.010	12	0.001
2^{-4}	0.346	15	0.026	15	0.008
2^{-5}	0.419	15	0.066	15	0.034
2^{-6}	0.810	15	0.271	15	0.197
2^{-7}	2.524	15	1.237	15	1.163
2^{-8}	9.663	14	5.212	14	7.168
2^{-9}	36.660	13	20.118	13	58.599

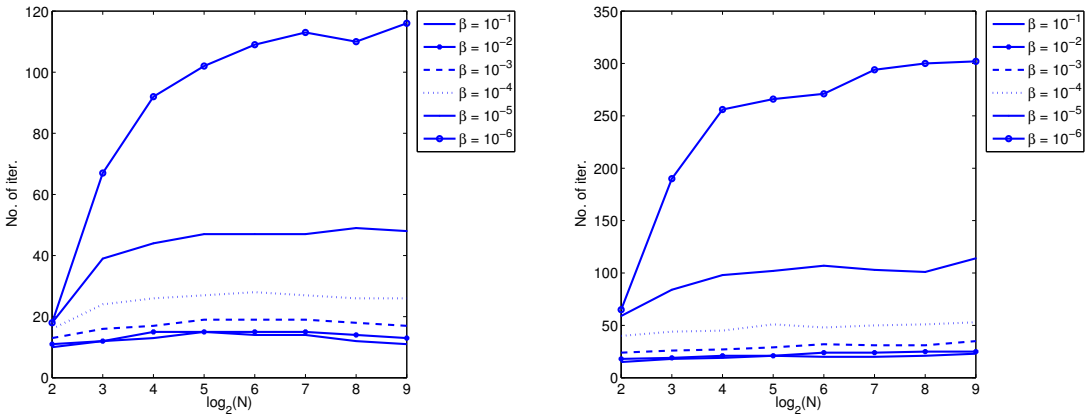
is singular – see Section 2.1 – the method as described above will not work[1]. We get around this problem by 'pinning' one of the nodes by enforcing a Dirichlet boundary condition at one point. This makes the matrix invertible K, and then the method as described above can be used as before.

Table 5.5 shows the results in this case, again taking $\beta = 10^{-2}$. Here the iteration count is higher than in the Dirichlet case, due to the worse spectral properties of the matrix K in this case. The algebraic multigrid method here seems to cope with the pinning of the node better than the geometric multigrid method. Here we see only a slight increase in iteration numbers as N decreases with a GMG approximation to K, and again we see mesh-independent convergence for the AMG approximation. The times still scale linearly with the problem size. Figure 5.9(b) again shows a plot of the number of iterations needed for different values of β.

Table 5.6: Comparison of solution methods for solving Example 5.1.3 using MINRES

h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	0.308	18	0.023	20	0.001
2^{-3}	0.309	19	0.019	21	0.002
2^{-4}	0.355	21	0.042	25	0.008
2^{-5}	0.498	21	0.119	25	0.046
2^{-6}	1.167	24	0.451	24	0.277
2^{-7}	3.887	24	1.947	24	1.916
2^{-8}	17.461	25	8.207	23	12.366
2^{-9}	68.469	25	33.789	23	97.767

[1]The singularity of K is not, in general, a problem for iterative methods – see, e.g. [39, Section 2.3] [128, Section 6.4]. The main issue here is that the right hand side for the second solve with the Laplacian does not necessarily satisfy the consistency condition required for a solution to exist.



(a) Example 5.1.2 (b) Example 5.1.3

Figure 5.9: Plot of problem size vs MINRES iterations needed for different β for distributed control problems with Dirichlet and Neumann boundary conditions.

Table 5.7 and Figure 5.10(a) give the same results for Example 5.1.4, which had mixed boundary conditions. We see that we again get convergence that is mesh-independent for both approximations to K, and the iteration numbers lie somewhere between those of the purely Neumann case and the purely Dirichlet case, as we might expect.

Table 5.7: Comparison of solution methods for solving Example 5.1.4 using MINRES

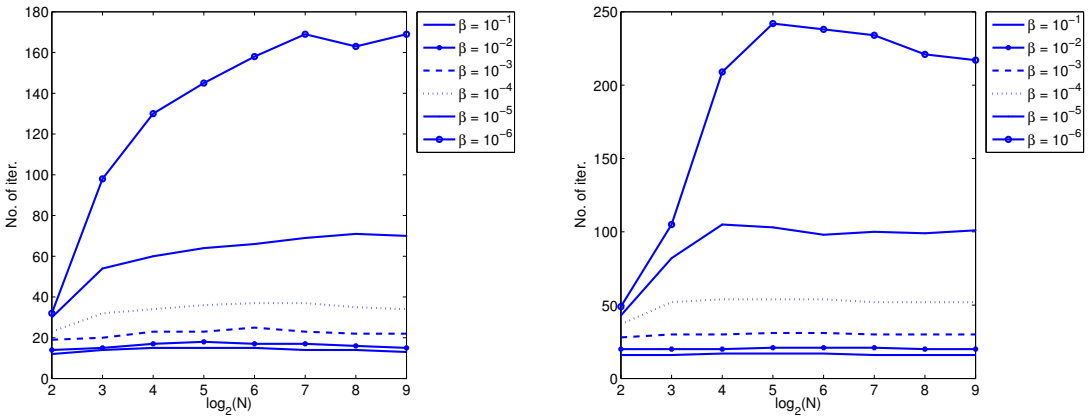
h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	1.078	14	0.392	14	0.001
2^{-3}	0.300	15	0.015	16	0.001
2^{-4}	0.335	17	0.030	18	0.007
2^{-5}	0.453	18	0.092	19	0.039
2^{-6}	0.892	17	0.359	19	0.236
2^{-7}	2.921	17	1.486	17	1.353
2^{-8}	11.223	16	5.929	16	7.788
2^{-9}	42.130	15	25.931	17	47.718

So far we have see that MINRES with block diagonal preconditioner is an optimal method of distributed control problems. Example 5.1.5 is a boundary control problem with Neumann boundary conditions. Table 5.8 and Figure 5.10(b) give the results in this case. Here we see that the number of iterations needed are again mesh-independent.

Table 5.8: Comparison of solution methods for solving Example 5.1.5 using MINRES

h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	0.327	20	0.034	22	0.001
2^{-3}	0.321	20	0.031	22	0.004
2^{-4}	0.358	20	0.038	22	0.009
2^{-5}	0.491	21	0.099	23	0.038
2^{-6}	1.004	21	0.344	23	0.299
2^{-7}	3.228	21	1.517	23	2.338
2^{-8}	12.553	20	6.910	23	22.328
2^{-9}	50.934	20	26.651	22	—[a]

[a] Ran out of memory



(a) Example 5.1.4 (b) Example 5.1.5

Figure 5.10: Plot of problem size vs MINRES iterations needed for different β for a distributed control problem with mixed boundary conditions and a boundary control problem with Neumann boundary conditions.

5.5 Block lower triangular preconditioners

Recall from Section 4.3 that the matrix

$$\begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix}^{-1} \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}$$

is self-adjoint in the inner product defined by $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, where

$$\mathcal{H} = \begin{bmatrix} A - A_0 & 0 \\ 0 & S_0 \end{bmatrix},$$

as long as this does indeed define an inner product – i.e. $A - A_0 > 0$ and $S_0 > 0$. A_0 and S_0 are chosen to be approximations to the (1,1) block and Schur complement

respectively. This enables us to use the block lower triangular preconditioner

$$\mathcal{P} = \begin{bmatrix} A_0 & 0 \\ B & -S_0 \end{bmatrix}$$

as a preconditioner in the conjugate gradient algorithm using the non-standard inner-product defined by $\mathcal{H}$.

The difficulty with this method is that in order for it to be applicable we need $A - A_0$ to be positive-definite. In our case, if $\widetilde{Q}$ is our approximation to the mass matrix, then this condition is the same as needing $Q - \widetilde{Q} > 0$. Now, if we assume that $\widetilde{Q}$ is positive definite, this condition is the same as saying that, for all $\mathbf{x}$, $\mathbf{x}^T Q \mathbf{x} > \mathbf{x}^T Q_0 \mathbf{x}$. Since $\widetilde{Q}$ is positive definite we can write $\widetilde{Q} = \widetilde{Q}^{\frac{1}{2}} \widetilde{Q}^{\frac{1}{2}}$, and so $\mathbf{y}^T \widetilde{Q}^{-\frac{1}{2}} Q \widetilde{Q}^{-\frac{1}{2}} \mathbf{y} > \mathbf{y}^T \mathbf{y}$ for all $\mathbf{y}$. It's therefore clear that the condition that $Q - \widetilde{Q} > 0$ is equivalent to having all the eigenvalues of $\widetilde{Q}^{-1} Q$ bigger than unity.

If we use the approximation $\widetilde{Q} = C_k$, the Chebyshev semi-iteration as defined in Section 5.2, then we can see from Table 5.2 that some of the eigenvalues will be less than one. However, we have such good knowledge of the eigenvalues for this approximation, which is a significant advantage here. Suppose we take our approximation to the (1,1) block to be

$$A_0 = \begin{bmatrix} \beta\, \gamma_k C_k & 0 \\ 0 & \gamma_k C_k \end{bmatrix},$$

for some scaling parameter γ_k. Then, from (5.13), it is easily seen that

$$\frac{\delta_k}{\gamma_k} \leq \frac{\mathbf{x}^T Q \mathbf{x}}{\mathbf{x}^T \gamma_k C_k \mathbf{x}} \leq \frac{\Delta_k}{\gamma_k}.$$

Therefore, provided we pick $\gamma_k < \delta_k$, $A - A_0$ is positive. The value of δ_k can be read off Table 5.2. Hence, in this case we can easily calculate a suitable scaling parameter a priori.

If we absorb the constant γ_k into the upper bound we can write

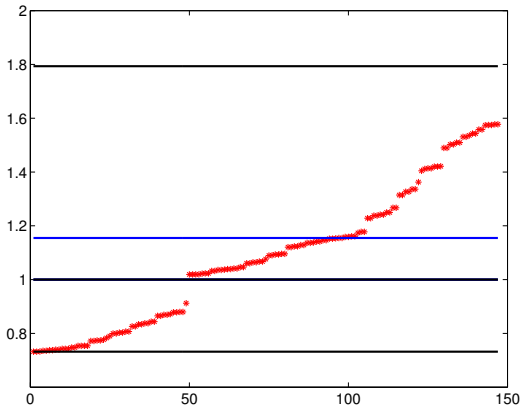
$$1 < \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T A_0 \mathbf{x}} \leq \Delta_k. \tag{5.25}$$

We can use the approximation to the Schur complement described in Section 5.3, i.e. $S_0 = \tilde{K} Q^{-1} \tilde{K}$. With these choices of A_0 and S_0, the conditions for Theorem 4.1.1 are satisfied, so using this along with (5.23) and (5.25), we get that the eigenvalues of the preconditioned system satisfy

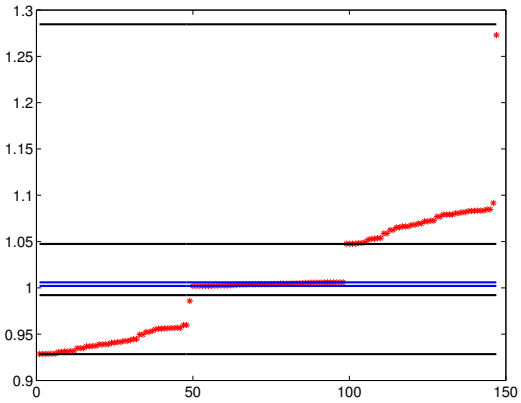
$$\begin{aligned} \frac{(1+\phi)\Delta_k - \sqrt{(1+\phi)^2\Delta_k^2 - 4\phi\Delta_k}}{2} \leq \lambda < \frac{(1+\Phi) - \sqrt{(1+\Phi)^2 - 4\Phi}}{2} \\ 1 < \lambda \leq \Delta_k \qquad \text{or} \\ \frac{(1+\phi) + \sqrt{(1+\phi)^2 - 4\phi}}{2} < \lambda \leq \frac{(1+\Phi)\Delta_k + \sqrt{(1+\Phi)^2\Delta_k^2 - 4\Phi\Delta_k}}{2}, \end{aligned}$$

where $\phi = c^*(\frac{1}{\beta}ch^4 + 1)(1 - \eta_m)^2$ and $\Phi = C^*(\frac{1}{\beta}C + 1)(1 + \eta_m)^2$.

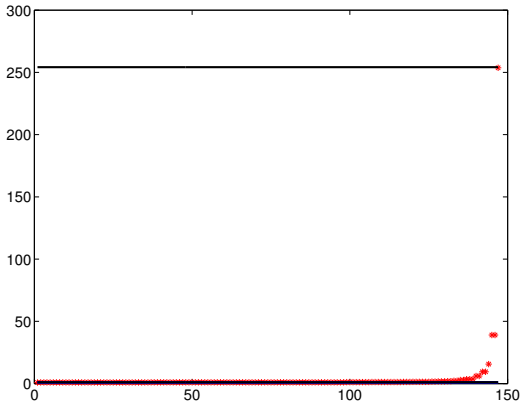
Figure 5.11 shows a plots of these eigenvalues, together with the predicted bounds, for Example 5.1.1 with a mesh size of $h = 2^{-3}$ and for regularization parameters $\beta = 10^{-2}$ and $\beta = 10^{-5}$. Here we give the eigenvalues in the case where solves with the stiffness matrix are exact and the mass matrix is approximated by five or ten steps of the Chebyshev semi-iteration, scaled by parameters $\gamma_5 = 0.92$ and $\gamma_{10} = 0.99$. In calculating the bounds we get δ_k, Δ_k from Table 5.2. Again, the plots suggest the bounds above are quite descriptive.



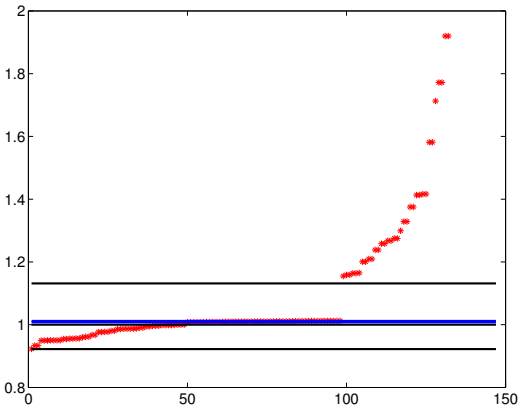
(a) $\beta = 10^{-2}$, five Chebyshev steps



(b) $\beta = 10^{-2}$, ten Chebyshev steps



(c) $\beta = 10^{-5}$, ten Chebyshev steps



(d) $\beta = 10^{-5}$, ten Chebyshev steps (zoomed in)

Figure 5.11: Eigenvalues ('*') and predicted bounds ('-') for the block lower triangular preconditioner where $\beta = 10^{-2}, 10^{-5}$, with exact K solves and approximate Q solves.

We now see how this method performs in practice. We keep the same approximation to the Schur complement we had in the previous section – where a solve with K is either approximated by one GMG V-cycle with 3 pre-smoothing steps – or it's transpose as appropriate – or alternatively by one MI20 AMG V-cycle with the

default parameters.

We again approximate a solve with the mass matrix by five steps of the Chebyshev semi-iteration. We now need to choose the parameter γ_5. Table 5.2 gives us the smallest eigenvalue, $\delta_5 = 0.9375$, hence the parameter can take any value smaller than that. Table A.4 in Appendix A shows that the performance of the method doesn't seem critical on the value of the parameter – even values greater than δ_5 appear to perform well, although there is no theory to guarantee these will always work. However, as we should expect given the results in [20], convergence gets worse the further the ratio δ_5/γ_5 is from unity. We take $\gamma_5 = 9.2$ in the following computations.

The norm that this method minimizes is not easily computable, so in the results presented below we stop the method when the residual measured in the 2-norm is reduced by a factor of 10^{-6}.

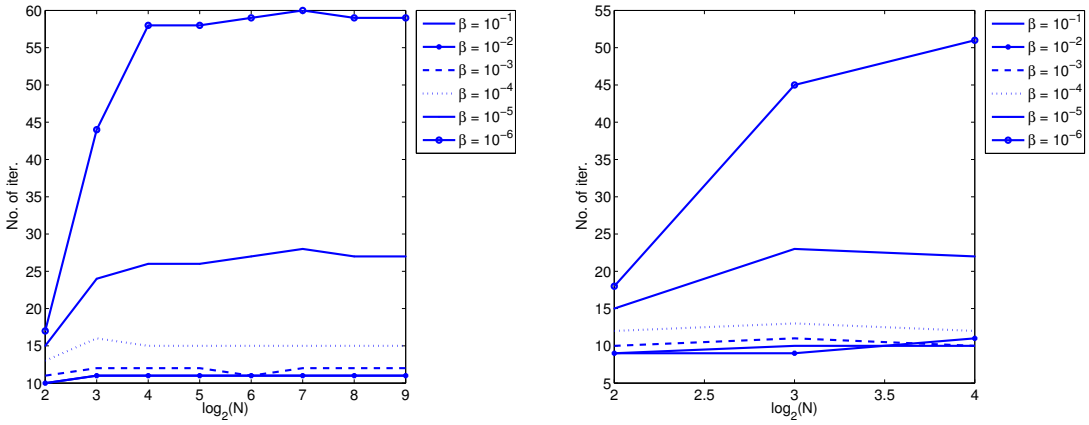
First, consider Example 5.1.1, the distributed control problem with Dirichlet boundary conditions. Table 5.9 gives the results in this case. We see that the method performs well here, giving mesh size-independent convergence, and again the time to solve the system scales linearly with the problem size. Figure 5.12(a) shows how the convergence changes with β, and again we see that the method performs very well here down to $\beta = 10^{-5}$.

Table 5.9: Comparison of solution methods for solving Example 5.1.1 using BPCG

h	GMG		AMG		`backslash`	$\|y_h^G - \hat{y}\|_2$	$J(y_h^G, u_h^G)$	$\|\mathbf{u}^G - \mathbf{u}^B\|_2$
	time	its	time	its	time			
2^{-2}	1.443	11	0.067	11	0.000	1.412e-03	8.103e-04	6.647e-08
2^{-3}	0.299	11	0.011	11	0.010	1.404e-03	7.962e-04	2.781e-07
2^{-4}	0.314	10	0.019	10	0.025	1.396e-03	7.891e-04	2.283e-06
2^{-5}	0.378	10	0.056	10	0.034	1.394e-03	7.871e-04	5.976e-06
2^{-6}	0.664	10	0.232	10	0.192	1.393e-03	7.866e-04	9.857e-06
2^{-7}	1.907	10	1.120	11	1.163	1.393e-03	7.865e-04	1.914e-05
2^{-8}	7.657	10	5.209	11	7.149	1.393e-03	7.864e-04	2.809e-05
2^{-9}	32.136	10	22.240	11	61.681	1.393e-03	7.864e-04	3.619e-05

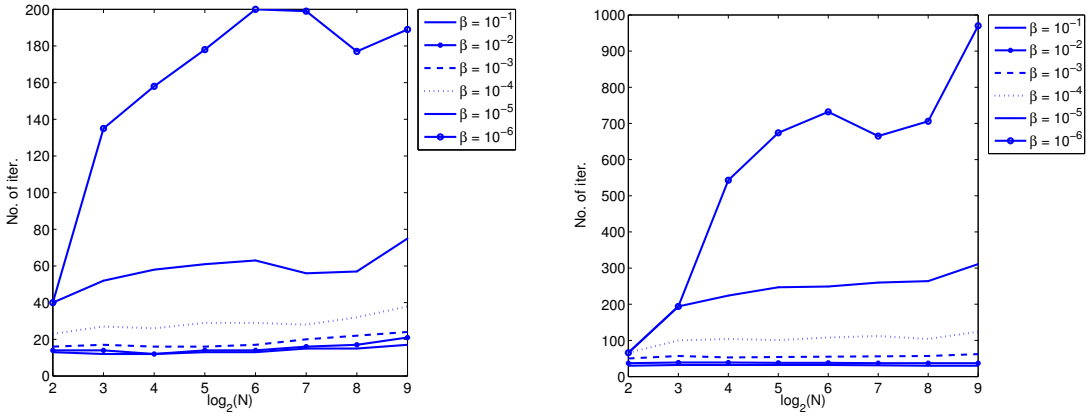
Next, consider Example 5.1.3, a distributed control problem with Neumann boundary conditions. The results in this case are given in Table 5.11 and Figure 5.13(a). The results are slightly worse here, and we have a slight dependence on the mesh size. As we saw when solving the system using MINRES, AMG performs better here than the geometric multigrid routine.

Our final example is the boundary control problem, Example 5.1.5. The results of using BPCG on this problem are given in Table 5.12 and Figure 5.13(b). We see



(a) Example 5.1.1, 2D (b) Example 5.1.1, 3D

Figure 5.12: Plot of problem size vs BPCG iterations needed for different β for distributed control problems with Dirichlet boundary conditions in two and three dimensions.



(a) Example 5.1.3 (b) Example 5.1.4

Figure 5.13: Plot of problem size vs BPCG iterations needed for different β for distributed control problems with Neumann and mixed boundary conditions.

Table 5.10: Comparison of solution methods for solving Example 5.1.1 in three dimensions using BPCG

h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	0.133	10	0.022	10	0.013
2^{-3}	0.138	9	0.055	9	0.108
2^{-4}	0.597	8	0.777	8	6.572
2^{-5}	4.810	7	8.716	7	—[a]
2^{-6}	43.176	7	—[b]		—[a]

[a] Ran out of memory
[b] AMG failed

Table 5.11: Comparison of solution methods for solving Example 5.1.3 using BPCG

h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	0.321	14	0.020	14	0.001
2^{-3}	0.318	14	0.018	14	0.002
2^{-4}	0.327	12	0.032	14	0.008
2^{-5}	0.444	14	0.089	14	0.047
2^{-6}	0.886	14	0.347	14	0.283
2^{-7}	3.028	16	1.565	15	1.910
2^{-8}	13.387	17	7.208	15	12.130
2^{-9}	66.489	21	33.921	17	133.496

that the method takes significantly more iterations to converge here, although it is still mesh-size independent.

Table 5.12: Comparison of solution methods for solving Example 5.1.5 using BPCG

h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	0.361	37	0.050	42	0.001
2^{-3}	0.366	40	0.050	42	0.002
2^{-4}	0.436	39	0.081	41	0.007
2^{-5}	0.673	38	0.205	42	0.038
2^{-6}	1.653	38	0.718	42	0.300
2^{-7}	5.729	38	3.330	44	2.371
2^{-8}	24.500	38	14.675	43	22.435
2^{-9}	97.559	37	58.721	41	—[a]

[a] Ran out of memory

5.6 Constraint preconditioners

The third approach we shall consider for solving these problems is using a projected conjugate gradient method with a constraint preconditioner. The theory behind this method was developed by H.S. Dollar in [95], and is included here for completeness.

A constraint preconditioner in this case would be of the form

$$\mathcal{P} = \begin{bmatrix} P_{(1,1)} & P_{(2,1)} & -Q \\ P_{(1,2)} & P_{(2,2)} & K \\ -Q & K & 0 \end{bmatrix}.$$

Define $P := \begin{bmatrix} P_{(1,1)} & P_{(2,1)} \\ P_{(1,2)} & P_{(2,2)} \end{bmatrix}$. Then Theorem 4.4.1 tells us that if Z is a basis for the nullspace of $[-Q \;\; K]$, then the eigenvalues of the preconditioned system will either satisfy $\lambda = 1$, or $Z^T A Z \mathbf{z} = \lambda Z^T P Z \mathbf{z}$ for some $\mathbf{z}$, where A here is, as usual, $\texttt{blkdiag}(\beta Q, Q)$.

Since we are assuming we have discretized the state and control using the same basis, the matrix we called $\widehat{Q}$ in (5.1) is square, and nonsingular. We can therefore consider the fundamental basis matrix,

$$Z = \begin{bmatrix} -Q^{-1}K \\ I \end{bmatrix}.$$

Thus we have

$$\begin{aligned} Z^T A Z &= \begin{bmatrix} -KQ^{-1} & I \end{bmatrix} \begin{bmatrix} \beta Q & 0 \\ 0 & Q \end{bmatrix} \begin{bmatrix} -Q^{-1}K \\ I \end{bmatrix} \\ &= \beta K Q^{-1} K + Q. \end{aligned}$$

Note the similarity between this expression and Schur complement in Section 5.3. We can use the same ideas presented there to develop an efficient constraint preconditioner. We want a matrix $\mathcal{P}$ such that it is a constraint preconditioner and is such that $Z^T P Z = \beta K Q^{-1} K$. The obvious choice of a matrix with these properties would be

$$\mathcal{P} = \begin{bmatrix} \beta Q & 0 & -Q \\ 0 & 0 & K \\ -Q & K & 0 \end{bmatrix}.$$

This preconditioner was considered by Biros and Ghattas [11]. In this situation an argument as used to get (5.15) tells us that the upper and lower bounds on the eigenvalues of the generalized eigenvalue problem $Z^T A Z = \lambda Z^T P Z$ would be $\frac{1}{\beta} C + 1$

and $\frac{1}{\beta}ch^4 + 1$. Using this as a constraint preconditioner will therefore give us a preconditioned system with eigenvalues such that

$$\begin{aligned} & \lambda = 1, \\ \text{or} \qquad & \frac{1}{\beta}ch^4 + 1 \leq \lambda \leq \frac{1}{\beta}C + 1. \end{aligned}$$

As we have seen in the previous sections, this will be well clustered for β large enough. This just leaves us to consider how easy it is to invert in deciding whether its an usable preconditioner or not.

The first thing to note is that by permuting rows and columns we could transform $\mathcal{P}$ into a triangular matrix – it is, using the definition of John Gilbert, a psychologically block triangular matrix. To solve the system

$$\begin{bmatrix} \beta Q & 0 & -Q \\ 0 & 0 & K \\ -Q & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix}$$

we just need to solve

$$\begin{aligned} K\mathbf{x}_3 &= \mathbf{r}_2 \\ \beta Q\mathbf{x}_1 &= \mathbf{r}_1 + Q\mathbf{x}_3 \\ K\mathbf{x}_2 &= \mathbf{r}_3 + Q\mathbf{x}_1. \end{aligned}$$

In order for constraint preconditioning to be successful we must solve for the constraints exactly so that we remain on the constraint manifold. Here this means that at each iteration we must solve for K – which is equivalent to solving the PDE – with high accuracy twice, which is computationally expensive. Ideally we would like to find a preconditioner with the same spectral properties, but without having to solve exactly for K where this appears in the constraint blocks. Consider the matrix

$$\mathcal{P} = \begin{bmatrix} 0 & 0 & -Q \\ 0 & \beta KQ^{-1}K & K \\ -Q & K & 0 \end{bmatrix}.$$

Here, again, we see that

$$\begin{aligned} Z^T AZ &= \begin{bmatrix} KQ^{-1} & I \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & KQ^{-1}K \end{bmatrix} \begin{bmatrix} Q^{-1}K \\ I \end{bmatrix} \\ &= KQ^{-1}K \end{aligned}$$

and so the eigenvalues of the preconditioned system in this case are the same as that we just considered. This preconditioner differs in that to solve this system we need to solve

$$\begin{aligned} Q\mathbf{x}_3 &= -\mathbf{r}_1 \\ \beta K Q^{-1} K \mathbf{x}_2 &= \mathbf{r}_2 - K\mathbf{x}_3 \\ Q\mathbf{x}_1 &= \mathbf{r}_3 - K\mathbf{x}_2. \end{aligned}$$

These equations seem to require even more work to solve than the preconditioner we just considered. However, on closer inspection we see that the two solves for K that are required here are not from the constraint, but from A, the part we are free to choose, leaving the possibility of approximating the action of the inverse of K available. The only part of the constraint block that we have to solve here is Q, and we have efficient ways of solving for Q. Thus, if we use the same (multigrid) approximation as in the previous two sections, a practical preconditioner would be

$$\mathcal{P}_2 = \begin{bmatrix} 0 & 0 & -Q \\ 0 & \beta\widetilde{K}Q^{-1}\widetilde{K} & K \\ -Q & K & 0 \end{bmatrix}.$$

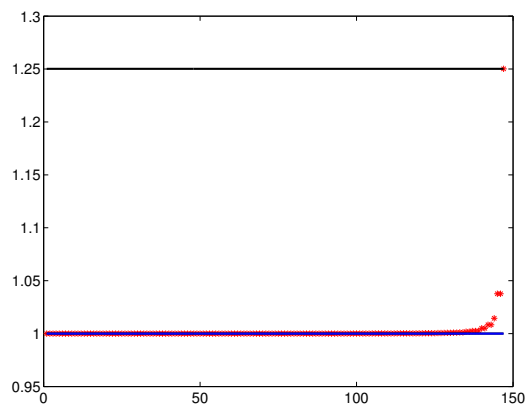
An argument as in Section 5.3 shows that, with this modification, the eigenvalues of the preconditioned system in this case satisfy

$$\begin{aligned} \lambda &= 1, \\ \text{or} \qquad c^*(1-\eta_m)^2(\frac{1}{\beta}ch^4+1) &\leq \lambda \leq C^*(1+\eta_m)^2(\frac{1}{\beta}C+1), \end{aligned}$$

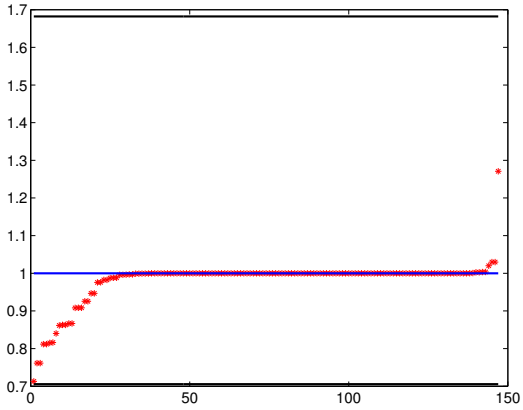
where c^*, c, C^*, C are constants independent of h and β. Note that these eigenvalues are more tightly clustered than in the other two methods.

Figure 5.14 shows a plots of these eigenvalues, together with the predicted bounds, for Example 5.1.1 with a mesh size of $h = 2^{-3}$ and for regularization parameters $\beta = 10^{-2}$ and $\beta = 10^{-5}$. Here we give the eigenvalues in the case where solves with the mass matrix are exact and the stiffness matrix is either solved exactly, or approximated by one GMG V-cycle. In calculating the bounds we take $\eta_k = 0.15$.

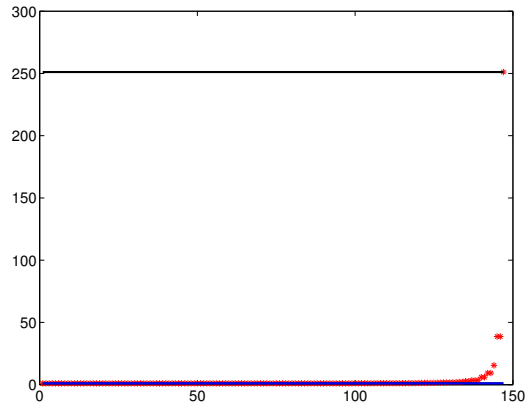
Note that this method relies heavily on the fact that $\widehat{Q}$ is invertible. The method as presented here can therefore only be used if we discretize the control and the state using the same elements. That means that it cannot be used to solve boundary control problems, for example. It may be possible to adapt the method by considering only an invertible subsection of the matrix $\widehat{Q}$, but there is at present no theory on this approach.



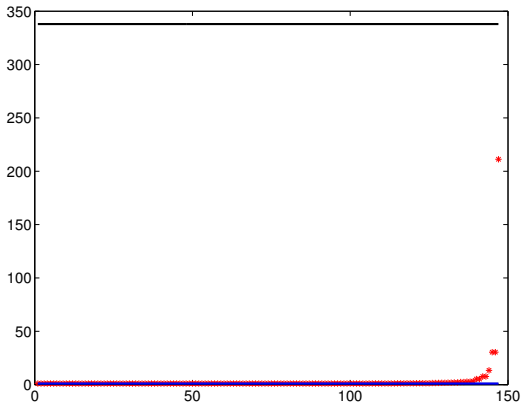
(a) $\beta = 10^{-2}$, exact solves



(b) $\beta = 10^{-2}$, approx solves



(c) $\beta = 10^{-5}$, exact solves



(d) $\beta = 10^{-5}$, approx solves

Figure 5.14: Eigenvalues ('*') and predicted bounds ('-') for the constraint preconditioner where $\beta = 10^{-2}, 10^{-5}$, with exact and approximated K solves.

We now look at some numerical results when using this preconditioner. Again, here we look at two approximations to the Schur complement – a solve with K is approximated by either one GMG V-cycle with 3 smoothing steps, or by one AMG V-cycle using HSL MI20 with the default parameters.

Here we need to solve accurately with the mass matrix, so we run the Chebyshev semi-iteration until it has converged to a tolerance of 10^{-12}.

We test for convergence on $\|\mathbf{r}^{(k)}\|_{(Z^TPZ)^{-1}}$, which is an easily computed approximation to $\|\mathbf{x} - \mathbf{x}^{(k)}\|_{Z^TAZ}$, the norm in which PPCG converges [52]. In the results that follow we run the method until this quantity is reduced by a factor of 10^{-6}.

First, consider Example 5.1.1, the distributed control problem with Dirichlet boundary conditions. The results for this problem are given in Tables 5.13 and 5.14. A graph of the number of iterations for different β is given in Figure 5.15(a).

Table 5.13: Comparison of solution methods for solving Example 5.1.1 using PPCG

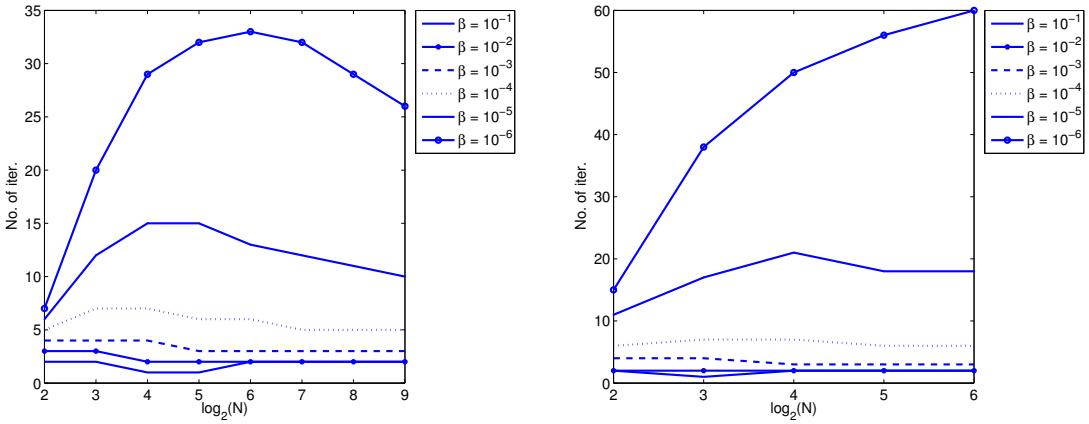
h	GMG		AMG		`backslash`	$\|y_h^G - \hat{y}\|_2$	$J(y_h^G, u_h^G)$	$\|\mathbf{u}^G - \mathbf{u}^B\|_2$
	time	its	time	its	time			
2^{-2}	0.323	3	0.018	3	0.000	1.412e-03	8.103e-04	2.206e-04
2^{-3}	0.290	3	0.008	3	0.001	1.404e-03	7.962e-04	2.832e-03
2^{-4}	0.299	3	0.011	2	0.006	1.397e-03	7.891e-04	3.245e-02
2^{-5}	0.337	3	0.032	2	0.036	1.395e-03	7.873e-04	2.969e-01
2^{-6}	0.497	3	0.130	2	0.199	1.398e-03	7.909e-04	2.725e+00
2^{-7}	0.979	2	0.560	2	1.178	1.294e-03	3.681e-03	1.385e+02
2^{-8}	3.690	2	2.807	2	7.312	1.292e-03	2.545e-02	7.836e+02
2^{-9}	19.930	3	13.153	2	60.739	1.417e-03	2.904e-03	5.096e+02

Table 5.14: Comparison of solution methods for solving Example 5.1.1 in three dimensions using PPCG

h	GMG		AMG		`backslash`
	time	its	time	its	time
2^{-2}	0.696	2	0.132	2	0.036
2^{-3}	0.114	2	0.032	2	0.059
2^{-4}	0.345	2	0.428	2	7.051
2^{-5}	3.215	2	4.761	1	—[a]
2^{-6}	65.770	2	—[b]		—[a]

[a] Ran out of memory
[b] AMG failed

We see from table 5.13 that this method has remarkably good convergence – converging in just two iterations for the AMG approximation to K. However, if we look at value, the cost functional, this is not decreasing as we refine h. Also, the



(a) Example 5.1.1, 2D

(b) Example 5.1.1, 3D

Figure 5.15: Plot of problem size vs PPCG iterations needed for different β for distributed control problems with Dirichlet conditions.

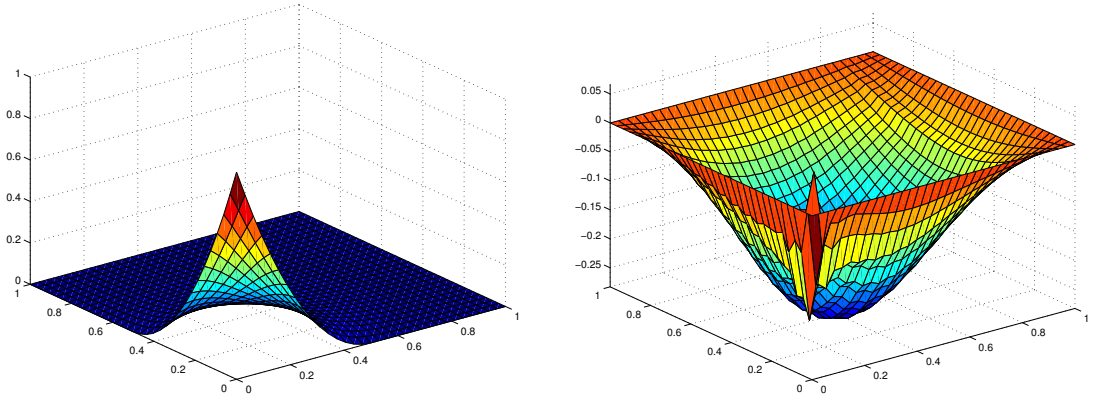
difference between the exact solution and the solution found by iterating using this method grows significantly as h decreases. Figure 5.16 shows the plots of the control and state calculated using this method. To the eye, the state in Figure 5.16(a) looks like the state in Figure 5.1(b), but the control in Figure 5.1(c) differs from that in Figure 5.16(b).

Figure 5.17 shows a plot of the residuals of this method in both the norm described above, in with PPCG converges, and also the 2-norm. We can see from this that, while the residual in the PPCG norm gets reduced by a factor of $\sim 10^{-30}$ in 10 iterations, the residual in the 2-norm stagnates at $\sim 10^{-4}$. Therefore, to find an approximation to the optimal control – the effectiveness of which is measured in the L^2 norm (2.27) – it seems that care must be taken when using this method.

Table 5.15 shows the results when applying the constraint preconditioner to the Neumann problem, Example 5.1.3.

5.7 Comments

In this chapter we looked at fast iterative methods for solving the discrete optimality systems that result with Poisson control, as introduced in Chapter 2. We tested our methods with distributed control – with Dirichlet, Neumann and mixed boundary conditions – and also boundary control with Neumann boundary conditions. The methods we introduced rely on two components: an approximation to the (1,1) block and the Schur complement.



(a) Computed state, y, for $\beta = 10^{-2}$

(b) Computed control, u, for $\beta = 10^{-2}$

Figure 5.16: Desired state, state and control for Example 5.1.1 in two dimensions, $\beta = 10^{-2}$, $h = 2^{-5}$ calculated using PPCG with a constraint preconditioner.

Figure 5.17: Residuals for solution with PPCG

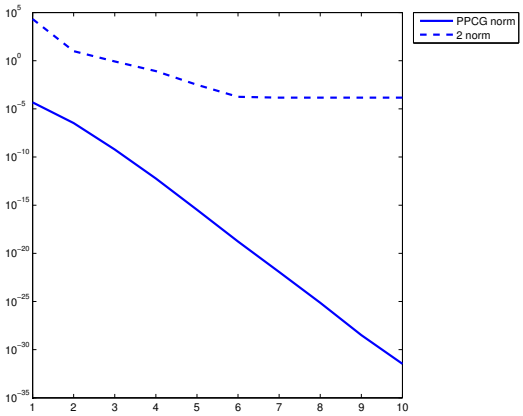


Table 5.15: Comparison of solution methods for solving Example 5.1.3 using PPCG

h	GMG		AMG		backslash
	time	its	time	its	time
2^{-2}	0.043	3	0.007	3	0.003
2^{-3}	0.010	3	0.009	3	0.007
2^{-4}	0.017	3	0.020	3	0.008
2^{-5}	0.043	3	0.042	3	0.055
2^{-6}	0.171	3	0.162	3	0.347
2^{-7}	0.669	3	0.665	3	1.772
2^{-8}	2.989	3	2.995	3	10.303
$t2^{-9}$	12.161	3	12.182	3	—[a]

[a] Ran out of memory

The (1,1) block in the PDE constrained optimization problems we have considered here is made up of mass matrices, and we showed that an inexpensive operation which is spectrally equivalent to these can be obtained by using the Chebyshev semi-iteration to accelerate relaxed Jacobi. This is an ideal situation to use this method, since we have all the required spectral information. This idea was described in Wathen and Rees [126].

For the Schur complement approximation we drop the term that doesn't contain the PDE, which is a good approximation for all but the smallest values of the regularization parameter. The bounds presented in this chapter for this approximation were given in [95]. Care must be taken that the approximation used for the PDE can also be used to approximate the square of the PDE – we show that this is the case for the multigrid approximation used here.

We then discussed three methods for solving the system. The first of these is MINRES with a block diagonal preconditioner – composed of the two approximations described above – which was first described in [95].

The second method we looked at relies on the fact that when we use a block diagonal preconditioner with the above approximations we have good knowledge of the spectrum of the preconditioned (1,1) block. Therefore we can trivially scale our approximation to make the (full) preconditioned system self-adjoint in a non-standard inner product, enabling us to use CG in this inner product. This method was first described in [96].

The final method we considered – due to Dollar – was also first described in [95]. This is a constraint preconditioner which can be used with the projected conjugate gradient method. All of the above methods were shown to be mesh size independent, and scale linearly with the problem size.

The techniques described here can be coupled with a primal-dual active set method to handle bound constraints on the control. For more detail see, for example, Rees, Stoll and Wathen [97] or Stoll and Wathen [111]. The first two methods require that the (1,1) block be non-singular – this would not be the case if, for example, you could only apply the control in part of the domain. How to adapt the methods to cope with this difficulty would be an interesting direction for future research.

There have been many other approaches suggested in the literature to solve the saddle-point system we are interested in, and we'd like to discuss a number of them here. A number of people have looked at solving the reduced Hessian system:

$$H_{red}\mathbf{u} = \mathbf{g}_{red},$$

which is obtained by block elimination. In our notation, $H_{red} = \beta Q_u + \hat{Q}^T K^{-1} Q_y K^{-1} \hat{Q}$, $\mathbf{g}_{red} = \hat{Q}^T K^{-1}\mathbf{b} + \hat{Q}^T K^{-1} Q_y K^{-1}\mathbf{d}$. This is much smaller than the original system, but is also dense, and cannot be explicitly formed or stored for all but the smallest problems. Among people who developed mesh-size independent methods for solving this system were Biros and Ghattas [11] and Haber and Asher [56].

For the specific problem as above for the Poisson equation, Schöberl and Zulehner [103] have developed preconditioners based on non-standard multigrid procedures which are both optimal with respect to the problem size *and* with respect to the choice of regularization parameter, β. The method in [103] was extended by Herzog and Sachs so that it is applicable to problems with bound constraints in [63]. It is, however, not clear how these methods would generalize to other problems. In [104] Schöberl, Zulehner and Simon recently described another multigrid preconditioner for elliptic optimal control problems which has h and β independent properties.

Other solution methods employing multigrid for this and similar classes of problems are described by, for example, Asher and Haber [6], Engel and Griebel [41]. See the survey article by and Borzi and Schulz [13] and the references therein for more information about multigrid solution methods. We mention that Biros and Dogan [12] have also developed a multigrid-based preconditioner which has both h and β independent convergence properties, but again it is not clear how their method would generalize to other PDEs.

Domain Decomposition and Model Order Reduction ideas are also successfully applied in this context; see for example Heinkenschloss and Nguyen ([60]) and Heinkenschloss, Sorensen and Sun([62]).

Chapter 6

Preconditioners for the optimal control of the convection-diffusion equation

6.1 The Convection-Diffusion equation

The convection diffusion equation is an equation of the form

$$-\epsilon\nabla^2 u + \vec{w}\cdot\nabla u = f, \tag{6.1}$$

where ϵ is a small parameter which, without loss of generality, we will take to be positive. This is similar to Poisson's equation, except we have an additional convection term, $\vec{w}\cdot\nabla u$. The equation (6.1) can be thought of as modelling the concentration of some pollutant, u, as it moves in a stream of velocity $\vec{w}$, and subject also to diffusion. The nature of the problem is highly dependent on the value of ϵ. For large ϵ the equation is dominated by the Laplacian term, so the solution will be highly diffusive in nature, like solutions of Poisson's equation. For small ϵ, however, the convection term will dominate and the solution will typically have layers.

Consider the boundary conditions

$$u = g_D \text{ on } \Omega_D, \qquad \frac{\partial u}{\partial n} = g_n \text{ on } \Omega_N,$$

and let $\mathcal{H}^1_E = \{u \in \mathcal{H}^1(\Omega) : u = g_D \text{ on } \partial\Omega_D\}$ and $\mathcal{H}^1_{E_0} = \{u \in \mathcal{H}^1(\Omega) : u = 0 \text{ on } \partial\Omega_D\}$. Then we can write the weak formulation of the problem: find $u \in H^1_E$ such that

$$\epsilon\int_\Omega \nabla u\cdot\nabla v + \int_\Omega (\vec{w}\cdot\nabla u)v = \int_\Omega fv + \epsilon\int_{\partial\Omega_N} g_N v \qquad \forall\ v \in H^1_{E_0}. \tag{6.2}$$

Note that the bilinear form here is not symmetric, due to the convection term – this is a big difference from Poisson's equation. We can try and solve (6.2) using Galerkin

finite elements, namely for some finite dimensional subspaces $V^h_{E_0} \subset \mathcal{H}^1_{E_0}$ and V^h_E, an affine space of functions that satisfy the Dirichlet boundary condition, find $u_h \in V^h_E$ such that

$$\epsilon \int_\Omega \nabla u_h \cdot \nabla v_h + \int_\Omega (\vec{w} \cdot \nabla u_h) v_h = \int_\Omega f v_h + \epsilon \int_{\partial\Omega_N} g_N v_h \qquad \forall\ v_h \in V^h_{E_0}.$$

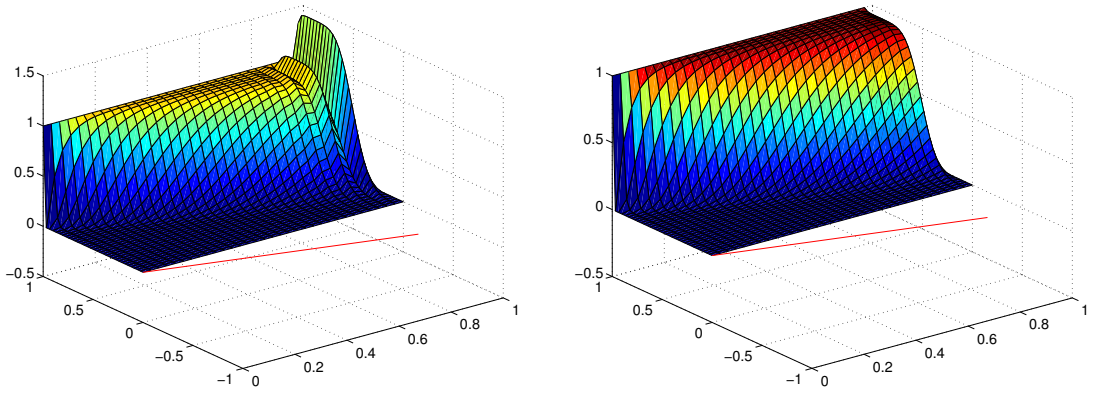
In the case where the convection term dominates it is well known that this method can be inaccurate, since the discrete solution can exhibit non-physical oscillations; see e.g. [39, Section 3.3]. We shall use the following example to illustrate the issues:

Example 6.1.1. *Let $\Omega = [0,1]^2$, and consider the problem*

$$\begin{aligned} -\epsilon \nabla^2 u + \vec{w} \cdot \nabla u &= 0, \\ u &= 0 \text{ on } \partial\Omega_0, \\ u &= 1 \text{ on } \partial\Omega_1, \end{aligned}$$

where $\partial\Omega_1 = [0,1] \times \{1\}$, $\partial\Omega_0 = \partial\Omega \backslash \partial\Omega_1$ and $\vec{w} = [\sin\frac{\pi}{6}, -\cos\frac{\pi}{6}]$.

This example has a boundary layer near $x = 1$, as can be seen on the plot of the solution in Figure 6.1(b). Figure 6.1(a) shows the solution obtained using a Galerkin method – note the presence of the oscillations near the boundary layer. The line in the figure indicates the direction of $\vec{w}$.



(a) Galerkin Solution (b) SUPG Solution

Figure 6.1: Plots of stabilized and unstabilized solutions of Example 6.1.1, where $\epsilon = 1/200$ and $h = 2^{-5}$.

Consider the general problem again, and let $\vec{w} = W\vec{w}_*$, where $\vec{w}_*$ is a vector normalized to have size unity in some norm, and let L be a characterizing length scale for the problem. Then we can consider the *Peclet number* for the problem,

$$\mathcal{P} := \frac{WL}{\epsilon}.$$

If $\mathcal{P} \gg 1$, then the solution can have a steep gradient, and so a very fine mesh is required to resolve this behaviour correctly. If a fine mesh is not used with a Galerkin method one generally sees local oscillations generated by these layers, as in Figure 6.1(a).

This effect is clearly dependent on the mesh size, h, so we want a notion of the Peclet number that is also dependent on h that gives us an intuition about what is a good mesh. This is given by the *mesh Peclet number*:

$$\mathcal{P}_h := \frac{\mathcal{P}h}{2L}.$$

Note here that h here is taken to be the length of the longest element edge. This gives a more quantifiable way of seeing of how changing the discretization can give a better accuracy of the solution; for $\mathcal{P}_h \gg 1$, a Galerkin method is likely to be inaccurate. We refer the interested reader to, for example, the book by Roos, Stynes and Tobiska [98, Section III.3] for more detail.

The Streamline Upwind Petrov-Galerkin (SUPG) method, originally introduced by Hughes and Brooks [70], is one way to alleviate the difficulty described above. This is a Petrov-Galerkin method – i.e. the test spaces and the trial spaces in the finite element method are different. In particular, as above we take the trial space to be V_E^h, but we take the test space to be the set of vectors of the form $v_h + \delta\vec{w}\cdot\nabla v_h$, where $v_h \in V_{E_0}^h$ and δ is a constant parameter which is to be determined a priori and, in fact, can be defined locally on individual elements. Proceeding formally, we thus get the bilinear form defined by

$$\begin{aligned} a_{sd}(u_h, v_h) &= \epsilon\int_\Omega \nabla u_h\cdot\nabla(v_h+\delta\vec{w}\cdot\nabla v_h) - \epsilon\delta\int_{\partial\Omega}(\vec{w}\cdot\nabla v_h)\frac{\partial u_h}{\partial n} \\ &\quad + \int_\Omega(\vec{w}\cdot\nabla u_h)(v_h+\delta\vec{w}\cdot\nabla v_h) \\ &= \epsilon\int_\Omega \nabla u_h\cdot\nabla v_h + \int_\Omega(\vec{w}\cdot\nabla u_h)v_h + \delta\int_\Omega(\vec{w}\cdot\nabla u_h)(\vec{w}\cdot\nabla v_h) \\ &\quad -\epsilon\delta\int_\Omega(\nabla^2 u_h)(\vec{w}\cdot\nabla v_h). \end{aligned}$$

However, u_h is not required to have a second derivative, so the last term may not be well defined. Assuming that the restriction of functions in V_E^h to individual elements Δ_k lies in $\mathcal{H}^2(\Delta_k)$, we can sum this last integral element-wise to give

$$\begin{aligned} a_{sd}(u_h, v_h) &= \epsilon\int_\Omega \nabla u_h\cdot\nabla v_h + \int_\Omega(\vec{w}\cdot\nabla u_h)v_h + \delta\int_\Omega(\vec{w}\cdot\nabla u_h)(\vec{w}\cdot\nabla v_h) \\ &\quad -\epsilon\delta\sum_k\int_{\Delta_k}(\nabla^2 u_h)(\vec{w}\cdot\nabla v_h). \end{aligned}$$

We also need to discretize the right hand side using the same trial and test spaces, giving

$$l_{sd}(v_h) := \int_\Omega f v_h + \delta \int_\Omega f(\vec{w} \cdot \nabla v_h),$$

and hence we can write the SUPG discretization as: Find $u_h \in V_E^h$ such that

$$a_{sd}(u_h, v_h) = l_{sd}(v_h) \quad \forall\ v_h \in V_{E_0}^h.$$

It remains to choose a value of the parameter δ. It is natural to choose different values of δ for each element, thus giving us a δ_k associated with each element Δ_k. Elman, Silvester and Wathen [39, p. 136] suggest using δ_k as given by the formula

$$\delta_k = \begin{cases} \frac{h_k}{2|\vec{w}_k|}\left(1 - \frac{1}{\mathcal{P}_h^k}\right) & \mathcal{P}_h^k > 1 \\ 0 & \mathcal{P}_h^k \leq 1 \end{cases}.$$

Here, h_k is a measure of the element length in the direction of the wind. For example, for a rectangular element of sides h_x and h_y, if $\vec{w} = [\cos\theta, \sin\theta]$,

$$h_k = \min(h_x/|\cos\theta|, h_y/|\sin\theta|).$$

This method was used to produce the plot in Figure 6.1(b). As we can see from the figure, this stabilization method has counteracted the oscillations that were present in the Galerkin solution, even though it cannot precisely resolve the boundary layer on the given mesh. In this sense, SUPG stabilization localizes errors in a boundary layer.

6.2 Iterative solution of the convection-diffusion equation

We saw in section 3.2 that the combination of multigrid and a Krylov subspace method is an efficient way to solve the discrete Poisson equation. The same is true in the case of the convection-diffusion equation – although in this case the multigrid methods are less well understood theoretically.

To get an effective multigrid method for the convection-diffusion equation one must be careful when defining the coarse-grid operator. Recall that in the case of the Poisson equation we advocated the use of the Galerkin coarse grid operator – $\bar{A} = P^T A P$, where P is the prolongation matrix. Here, we need a coarse grid matrix which does not introduce high frequencies. The Galerkin method may not be enough here, so instead we follow the method of Ramage [93] and explicitly construct the

coarse grid matrix using SUPG stabilization on the coarse grid. This choice of $\bar{A}$ will be enough to correctly resolve the features of the solution on the coarse grid – see, e.g. [93, 94].

The other important component of an effective multigrid method is the choice of smoothing operator; in particular, it is important to pick a smoother that takes into account the direction of the flow. Ramage [93] suggests using a block-Gauss-Seidel smoother to achieve this, with the blocks chosen to follow the direction of the flow as much as possible. The post-smoother then should be chosen to be in the opposite direction, so that the post-smoothing operator is the transpose of the pre-smoother.

The last remaining ingredient of a multigrid cycle is the grid-transfer operators. There has been some work on "operator dependent" transfer operators – see, e.g. [128, Section 5.4] and the references therein. However, if we define the smoother and coarse grid matrix as defined above, then the standard restriction and prolongation operators defined in Section 3.2 seem to work adequately here. See [39, Section 4.3] for a more detailed discussion of these issues.

As in Section 3.2, we would like to use multigrid as a preconditioner to accelerate convergence of a Krylov subspace method. Here the matrix that results from the discretization of the equation is not symmetric, so we have to use a non-symmetric solver. Recall from Section 3.1.8 that a suitable solver is GMRES.

Table 6.1 shows the results of solving Example 6.1.1 using GMRES, with the multigrid method described above as a preconditioner. As $h = 2^{-2}$ is the coarsest grid, the preconditioner is a direct solve, so GMRES converges in one iteration. For smaller h, convergence is essentially mesh-independent, as we expect from the theory.

Table 6.1: Solving the Convection-Diffusion equation using GMRES with a multigrid preconditioner

h	its	time
2^{-2}	1	0.297
2^{-3}	4	0.297
2^{-4}	4	0.308
2^{-5}	4	0.325
2^{-6}	4	0.366
2^{-7}	5	0.566
2^{-8}	5	1.415

6.3 Optimal control of the convection-diffusion equation

We now turn our attention to the problem of controlling the convection-diffusion equation. We can formulate this problem, as in Section 2.2, by minimizing a cost functional of tracking-type. In this section we will only consider distributed control problems with Dirichlet boundary conditions; the theory can be extended to other problems in a similar manner as in Chapters 2 and 5.

Given a vector field $\vec{w}$, a function g, and $\epsilon > 0$, the control problem is

$$\min_{y,u} \frac{1}{2}||y - \hat{y}||_2^2 + \frac{\beta}{2}||u||_2^2 \tag{6.3}$$

$$\text{s.t.} \qquad -\epsilon\nabla^2 y + \vec{w} \cdot \nabla y = u \text{ in } \Omega \tag{6.4}$$

$$y = g \text{ on } \partial\Omega, \tag{6.5}$$

for some regularization parameter β. This problem has been recently considered in, for example, [29, 92, 8]. We will illustrate our methods with the aid of an example.

Example 6.3.1. *Let* $\Omega = [0,1]^2$*, and find*

$$\begin{aligned} &\min_{y,u} \frac{1}{2}||y - \hat{y}||_2^2 + \frac{\beta}{2}||u||_2^2 \\ \text{s.t.} \qquad &-\epsilon\nabla^2 y + \vec{w} \cdot \nabla y = u \text{ in } \Omega \\ &y = \hat{y} \text{ on } \partial\Omega, \end{aligned}$$

where $\vec{w} = [\cos\theta, \sin\theta]$ *for some* θ*,* ϵ *is a constant, and*

$$\hat{y} = \begin{cases} (2x_1 - 1)^2(2x_2 - 1)^2 & \text{if } \mathbf{x} \in [0, \frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases}.$$

Example 6.3.1 uses the same objective function, $\hat{y}$, as we used in the case of the Poisson equation, which was plotted in Figure 5.1(a). This is not a natural solution to the convection-diffusion equation, so this state will, in general, not be easy to achieve.

In contrast to the case of Poisson control, since the convection diffusion equation is not self-adjoint there will in general be a difference between the optimize-then-discretize and the discretize-then-optimize approaches. In the next two sections we shall derive the discrete optimality conditions for each case.

6.3.1 Optimize-then-discretize

First, consider the optimize-then-discretize approach applied to the general problem (6.3)-(6.5). We shall use the Lagrange multiplier method, as described in Chapter 2.2.2. Consider the Lagrangian function

$$\mathcal{L}(y,u,p_1,p_2) := \frac{1}{2}||y-\hat{y}||^2_{L_2(\Omega)}+\beta||u||^2_{L_2(\Omega)}+\int_\Omega(-\epsilon\nabla^2 y+\vec{w}\cdot\nabla y-u)p_1\,dx+\int_{\partial\Omega}(y-g)p_2\,ds. \tag{6.6}$$

A necessary and sufficient condition for optimality is that the partial Frechét derivatives with respect to the four functions y, u, p_1 and p_2 vanish. Differentiation with respect to the Lagrange multipliers, p_1 and p_2, gives the state equation, which is the PDE constraint:

$$\left.\begin{array}{c} -\epsilon\nabla^2 y+\vec{w}\cdot\nabla y=u \\ y=g \ \text{ on } \ \partial\Omega. \end{array}\right\} \qquad \text{[State eqn]}$$

Differentiating (6.6) with respect to the state, y, gives the adjoint equation. We know

$$D_y(\mathcal{L}(\bar{y},\bar{u},p_1,p_2))h=0,$$

where $\bar{u}$ and $\bar{y}$ denote the optimal control and state respectively, and $D_y(\cdot)h$ denotes the partial Frechét derivative in the direction of h. Thus we have

$$\int_\Omega(y-\hat{y})h\,dx+\underbrace{\int_\Omega(-\epsilon\nabla^2 h)p_1\,dx}_{\text{[A]}}+\underbrace{\int_\Omega(\vec{w}\cdot\nabla h)p_1\,dx}_{\text{[B]}}+\int_{\partial\Omega}hp_2\,dx=0. \tag{6.7}$$

The two terms that contain h in a form other than simple multiplication are labelled [A] and [B] above, and we will treat these separately. First, consider [A]. Applying the Divergence theorem twice gives us

$$\int_\Omega(\epsilon\nabla^2 h)p_1\,dx=\epsilon\int_\Omega h(\nabla^2 p_1)\,dx-\epsilon\int_{\partial\Omega}p_1\frac{\partial h}{\partial n}\,ds+\epsilon\int_{\partial\Omega}h\frac{\partial p_1}{\partial n}\,ds. \tag{6.8}$$

We now turn our attention to [B].

$$\begin{aligned}\int_\Omega(\vec{w}\cdot\nabla h)p_1\,dx &= \int_\Omega(\nabla\cdot(h\vec{w}))p_1\,dx-\int_\Omega(h\nabla\cdot\vec{w})p_1\,dx \\ &= -\int_\Omega h(\vec{w}\cdot\nabla p_1)\,dx+\int_{\partial\Omega}p_1 h\vec{w}\cdot\vec{n}\,dx+\int_\Omega(h\nabla\cdot\vec{w})p_1\,dx.\end{aligned}$$

Substituting this and (6.8) into (6.7) we get

$$\begin{aligned}0 &= \int_\Omega(y-\hat{y})h\,dx-\epsilon\int_\Omega h(\nabla^2 p_1)\,dx+\epsilon\int_{\partial\Omega}p_1\frac{\partial h}{\partial n}\,ds-\epsilon\int_{\partial\Omega}h\frac{\partial p_1}{\partial n}\,ds \\ &\quad -\int_\Omega h(\vec{w}\cdot\nabla p_1)\,dx+\int_{\partial\Omega}p_1 h\vec{w}\cdot\vec{n}\,dx-\int_\Omega(h\nabla\cdot\vec{w})p_1\,dx+\int_{\partial\Omega}hp_2\,dx.\end{aligned} \tag{6.9}$$

This holds for all $h \in \mathcal{H}^1(\Omega)$. In particular, it must hold for all h such that $h = \frac{\partial h}{\partial n} = 0$ on $\partial\Omega$. Let S_1 denote the set consisting of all such h. Then the following holds for all $h \in S_1$:

$$0 = \int_\Omega (y - \hat{y})h \, dx - \epsilon \int_\Omega h(\nabla^2 p_1) \, dx - \int_\Omega h(\vec{w} \cdot \nabla p_1) \, dx - \int_\Omega h(\nabla \cdot \vec{w}) p_1 \, dx.$$

Since S_1 is sufficiently dense, applying the fundamental lemma of the calculus of variations we get

$$-\epsilon \nabla^2 p_1 - \vec{w} \cdot \nabla p_1 - (\nabla \cdot \vec{w}) p_1 = \hat{y} - y \quad \text{in } \Omega.$$

This relation is independent of h, and so holds for all h. If we return to (6.9) with this new information, and also assuming that $h \in \mathcal{H}_0^1(\Omega)$, then we see that

$$0 = \epsilon \int_{\partial\Omega} p_1 \frac{\partial h}{\partial n} \, ds \quad \forall h \in \mathcal{H}_0^1(\Omega),$$

which, applying the fundamental lemma of the calculus of variations once again, implies that $p_1 = 0$ on $\partial\Omega$. Now, looking at what remains of (6.9) for all $h \in \mathcal{H}^1(\Omega)$, we see that on $\partial\Omega$,

$$p_2 = \epsilon \frac{\partial p_1}{\partial n} - p_1 \vec{w} \cdot \vec{n}.$$

As this last equation is the only place where p_2 appears in the optimality system, we can define $p := p_1$, and so the adjoint equation is given by:

$$\left.\begin{array}{c} -\epsilon \nabla^2 p - \vec{w} \cdot \nabla p - (\nabla \cdot \vec{w}) p = \hat{y} - y \\ p = 0 \quad \text{on} \quad \partial\Omega. \end{array}\right\} \qquad \text{[Adjoint eqn]}$$

Lastly, differentiating (6.6) with respect to the control, u, gives us the gradient equation:

$$\beta u - p = 0. \qquad \text{[Grad. eqn]}$$

We have now derived the continuous optimality system for this control problem – namely, the state equation, adjoint equation and the gradient equation labelled above. If we discretize these, we get the equations that need to be solved to get a solution via the optimize-then-discretize approach.

First, we look at the gradient equation. Discretizing this leads to the equation

$$\beta Q \mathbf{u} - Q \mathbf{p} = 0,$$

where Q is the standard mass matrix. Next, discretizing the state equation is just like discretizing a standard convection diffusion problem, so we get

$$K_{sd} \mathbf{y} = G_{sd} \mathbf{u} + \mathbf{d},$$

where K_{sd} is the matrix defined by the bilinear form $a_{sd}(\cdot,\cdot)$ defined above, i.e.

$$(K_{sd})_{i,j} = \epsilon \int_\Omega \nabla\phi_i \cdot \nabla\phi_j + \int_\Omega (\vec{w}\cdot\nabla\phi_i)\phi_j + \delta \int_\Omega (\vec{w}\cdot\nabla\phi_i)(\vec{w}\cdot\nabla\phi_j) - \epsilon\delta \sum_k \int_{\Delta_k} (\nabla^2\phi_i)(\vec{w}\cdot\nabla\phi_j)$$

and similarly Q_{sd} is defined by $l_{sd}(\cdot)$ as

$$(Q_{sd})_{i,j} = \int_\Omega \phi_i\phi_j + \delta \int_\Omega \phi_i(\vec{w}\cdot\nabla\phi_j).$$

For the discretization of the adjoint equation, notice that this too is an convection diffusion equation. This differs from the state equation in that the 'wind', $\vec{w}$, is blowing from the opposite direction, and a reaction term $-\nabla\cdot\vec{w}$ has been introduced. We will assume this term is zero, as would be the case if the vector $\vec{w}$ describes an incompressible flow. The adjoint equation therefore becomes

$$\left.\begin{array}{c} -\epsilon\nabla^2 p - \vec{w}\cdot\nabla p = \hat{y} - y \\ p = 0 \ \text{ on } \ \partial\Omega. \end{array}\right\} \qquad \text{[Adjoint eqn]}$$

Again, this is just an convection diffusion equation, and – assuming we use the same finite element space for p as we did for u and y – we can discretize this as we did with the state equation to get

$$L_{sd}\mathbf{p} = \mathbf{b} - R_{sd}\mathbf{y},$$

where

$$\begin{aligned}
(L_{sd})_{j,i} &= \epsilon \int_\Omega \nabla\phi_i\cdot\nabla\phi_j - \int_\Omega (\vec{w}\cdot\nabla\phi_i)\phi_j + \delta\int_\Omega (\vec{w}\cdot\nabla\phi_i)(\vec{w}\cdot\nabla\phi_j) \\
&\quad + \epsilon\delta\sum_k \int_{\Delta_k} (\nabla^2\phi_i)(\vec{w}\cdot\nabla\phi_j), \\
(R_{sd})_{i,j} &= \int_\Omega \phi_i\phi_j - \delta\int_\Omega \phi_i(\vec{w}\cdot\nabla\phi_j), \text{ and} \\
\mathbf{b}_i &= \int_\Omega \hat{y}\phi_i.
\end{aligned}$$

Putting these three equations together gives us the following discrete problem, the solution of which gives us an approximation to the solution of the optimal control problem:

$$\begin{bmatrix} \beta Q & 0 & -Q \\ 0 & R_{sd} & L_{sd} \\ -Q_{sd} & K_{sd} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{6.10}$$

This formulation is strongly consistent in the sense that if we replace the finite dimensional u_h, y_h and p_h by their optimal values then the three equations that form the optimality system will be satisfied. It is, however, clearly not a symmetric linear

system, so there is no finite dimensional problem for which this is the optimality system. We also discretized both PDEs using the SUPG stabilization method, which is dependent on a stabilization constant δ. Even if we use the same finite element basis for all the quantities needed (y, u and p), it is not clear that the same value of δ would be applicable for the two PDEs.

6.3.2 Discretize-then-optimize

The second approach is to discretize the PDE first, and then solve the optimal control problem with this - finite dimensional - constraint. On discretizing the PDE we get, as in the last section,

$$K_{sd}\mathbf{y} = Q_{sd}\mathbf{u} + \mathbf{d}.$$

We discretize the cost functional in the same way, giving the minimum of (2.9) is equivalent to finding the minimum of

$$\min_{\mathbf{u},\mathbf{y}} \ \frac{1}{2}\mathbf{y}^T Q\mathbf{y} - \mathbf{y}^T\mathbf{b} + \beta\mathbf{u}^T Q\mathbf{u}.$$

We can therefore write the discrete version of the Lagrangian function:

$$\mathcal{L} := \frac{1}{2}\mathbf{y}^T Q\mathbf{y} - \mathbf{y}^T\mathbf{b} + \beta\mathbf{u}^T Q\mathbf{u} - \mathbf{p}^T(K_{sd}\mathbf{y} - Q_{sd}\mathbf{u} + \mathbf{d}).$$

As above, the optimality is given be the point at which the partial derivatives of $\mathcal{L}$ with respect to $\mathbf{y}, \mathbf{u}$ and $\mathbf{p}$ are equal to zero. This is given by the solution of the linear system

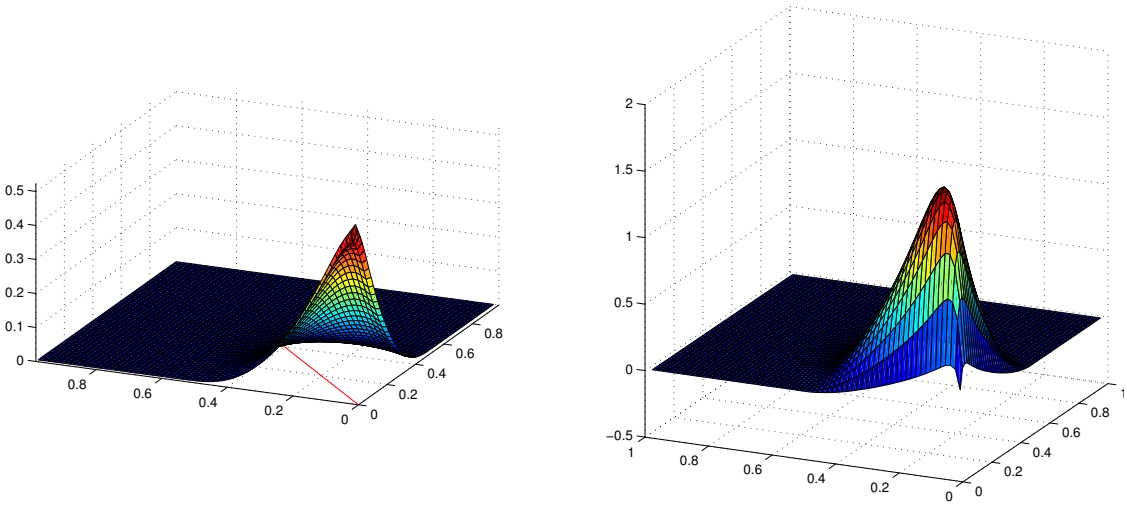
$$\begin{bmatrix} \beta Q & 0 & -Q_{sd}^T \\ 0 & Q & K_{sd}^T \\ -Q_{sd} & K_{sd} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{6.11}$$

This is clearly a symmetric linear system, different in form to that of (6.10). In this case only the last equation is strongly consistent in the sense defined above. The optimal control, state and adjoint would not satisfy the first two equations in this formulation.

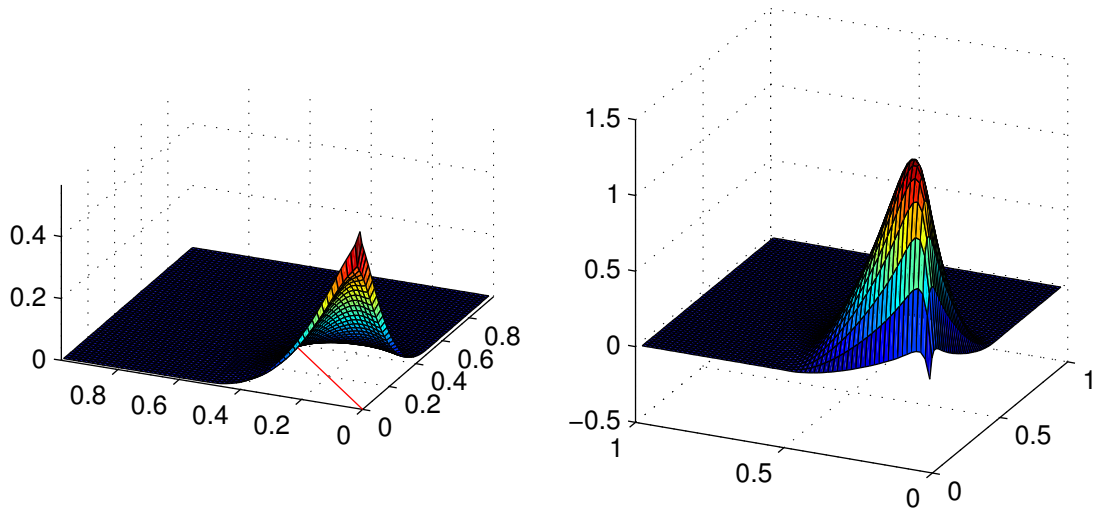
6.3.3 Comparison of the solutions

Figures 6.2 to 6.5 correspond to solving Example 6.3.1 with $\epsilon = 1/200$, $\beta = 0.01$, $N = 2^6$ and various values of θ.

Note that, in the discretize then optimize case, the approximation to the controls, u_{dto}, isn't always zero on the boundary, whereas the corresponding controls in the optimize then discretize case, u_{otd}, don't seem to have this problem. This seems to be

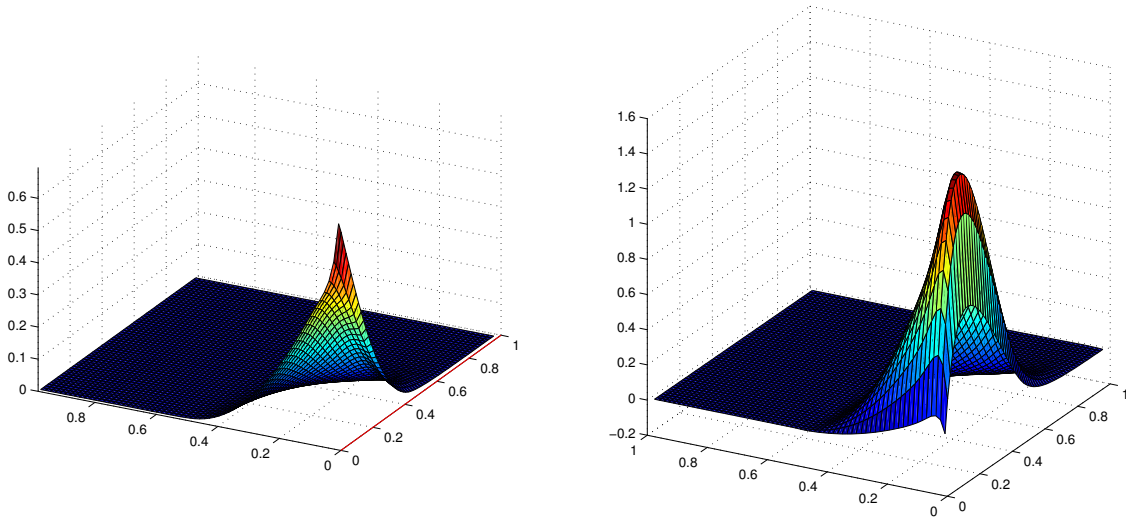


(a) Discretize then optimize.

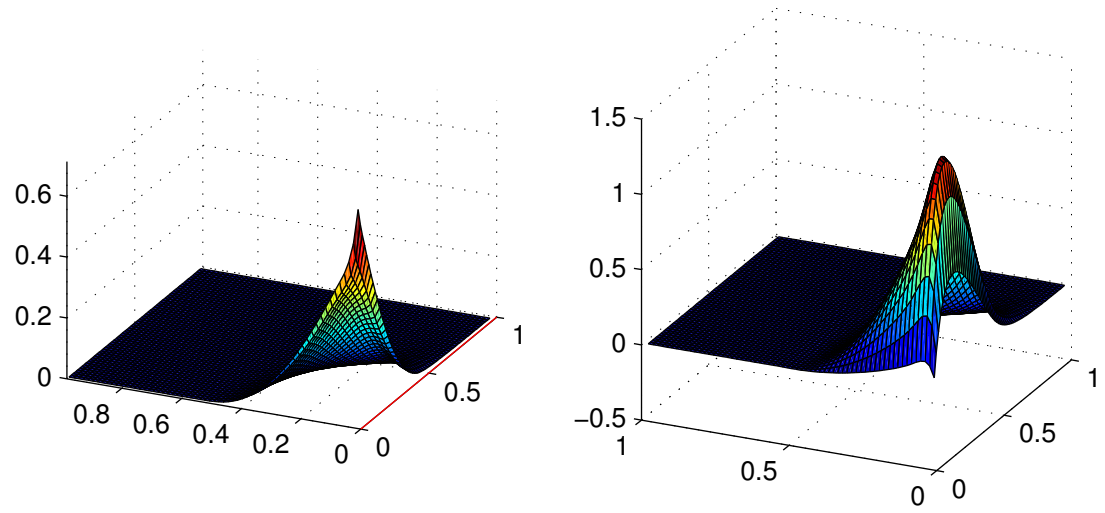


(b) Optimize then discretize.

Figure 6.2: Plot of optimal state (left) and control (right) for $\theta = \pi/4$. $\|u_{dto} - u_{otd}\|_2 = 0.8062$.

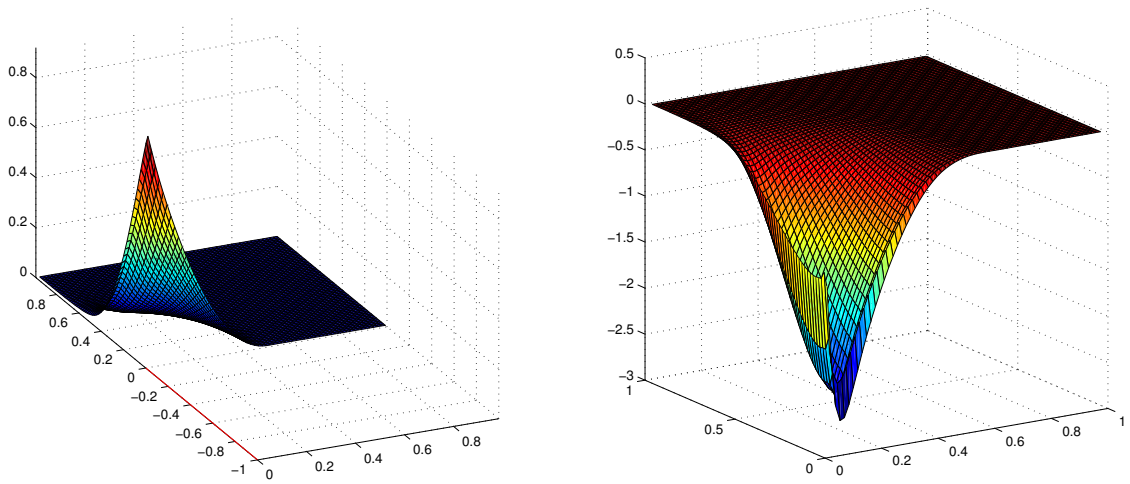


(a) Discretize then optimize.

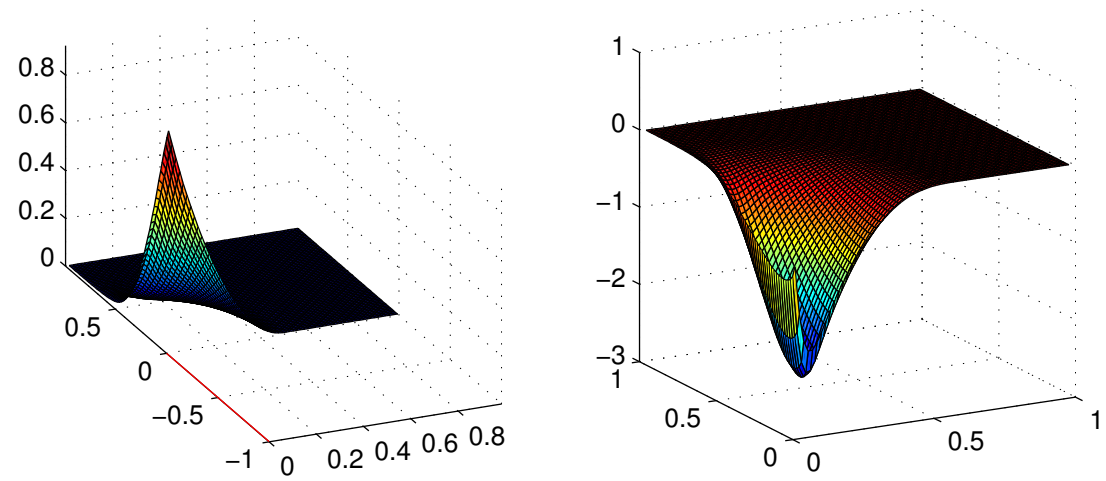


(b) Optimize then discretize.

Figure 6.3: Plot of optimal state (left) and control (right) for $\theta = 0$. $\|u_{dto} - u_{otd}\|_2 = 0.4831$.

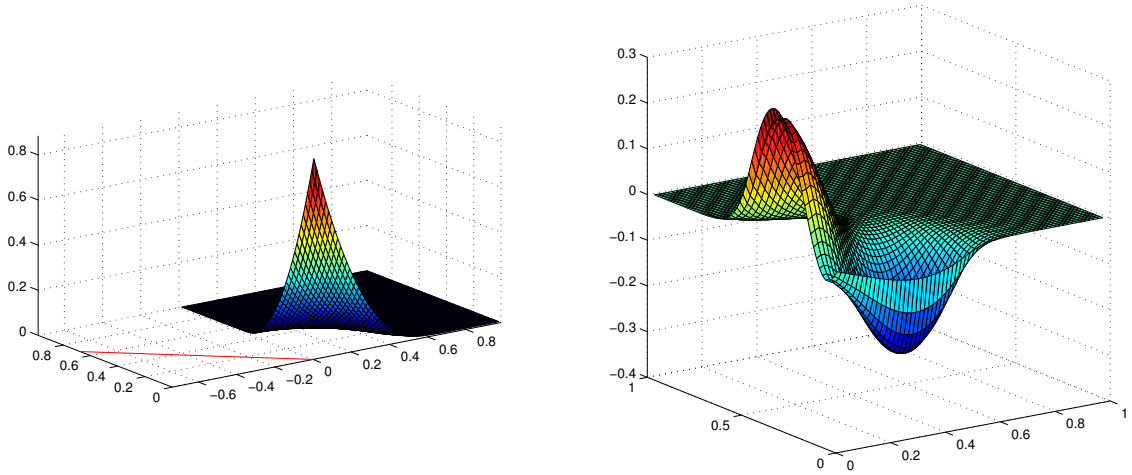


(a) Discretize then optimize.

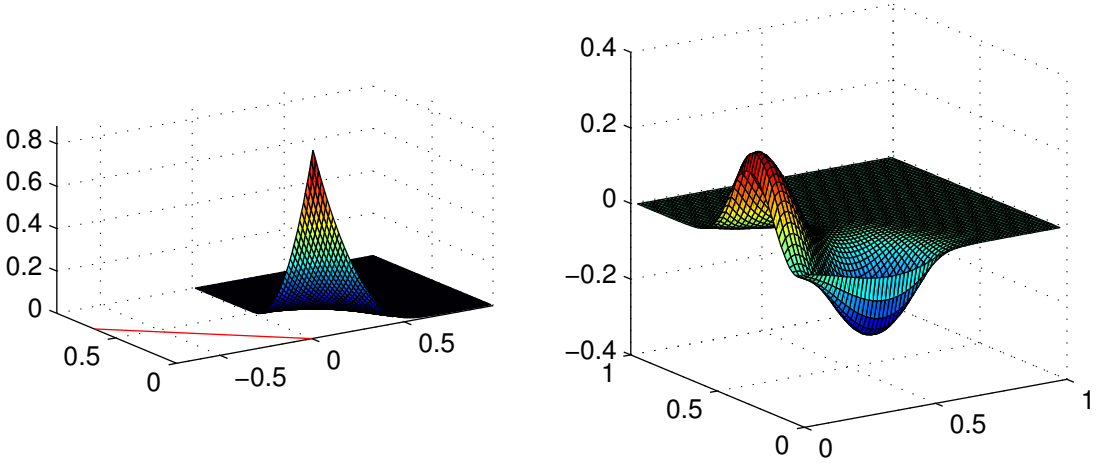


(b) Optimize then discretize.

Figure 6.4: Plot of optimal state (left) and control (right) for $\theta = 3\pi/2$. $\|u_{dto} - u_{otd}\|_2 = 1.904$.



(a) Discretize then optimize.



(b) Optimize then discretize.

Figure 6.5: Plot of optimal state (left) and control (right) for $\theta = 2.4$. $\|u_{dto} - u_{otd}\|_2 = 0.2968$.

due to the fact that the discretize then optimize method is not strongly consistent. In the continuous case the gradient equation tells us that the the optimal control is a scalar multiple of the adjoint variable, which is itself the solution of a convection-diffusion equation. This means that the optimal control can – and often does – have layers, even when the state does not; see, for example, Figures 6.3 and 6.4. We have taken no account of this in the discretize-then-optimize approach, and for this reason it seems that the optimize-then-discretize approach – although corresponding to no finite-dimensional optimization problem – is the better method.

The link between the forward and adjoint equations here mean that the analysis of a numerical method to solve the control problem for the convection-diffusion equation is significantly more complicated than corresponding analysis of the forward problem. Heinkenschloss and Leykekhman [61] show that applying SUPG to the control problem will always give no better than first order accuracy in the presence of a boundary layer.

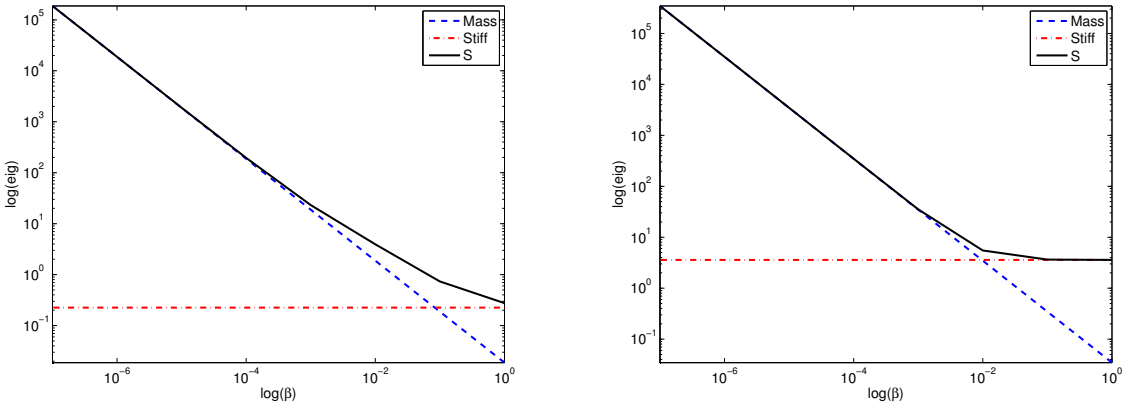
We mention that Becker and Vexler [8] and Braack [16] have recently developed stabilization methods for which the discrete problem is adjoint-consistent – that is, the optimize-then-discretize method gives a symmetric linear system. Heinkenschloss and Leykekhman [61] claim that even with such a method there is an inherent difficulty in the problem that will make the solution only first-order accurate in the presence of any layer.

6.4 Preconditioning

Consider first the system obtained when you discretize-then-optimize, (6.11). This system is symmetric positive definite, so we can again apply MINRES. In this case the exact Schur complement is $\frac{1}{\beta}Q_{sd}Q^{-1}Q_{sd}^T + K_{sd}Q^{-1}K_{sd}^T$. Figure 6.6 shows the maximum and minimum eigenvalues of the Schur complement, along with those of the first and second terms in the sum that forms it. We see that, in this case, the value of β for which the term with the mass matrices becomes important is larger than in the Poisson case. This means that – at least in the case where the control and state are discretized using the same bases – we may be able to consider two ideal preconditioners,

$$\mathcal{P}_1 := \texttt{blkdiag}(\beta Q, Q, K_{sd}Q^{-1}K_{sd}^T), \text{ and } \mathcal{P}_2 := \texttt{blkdiag}(\beta Q, Q, \frac{1}{\beta}Q_{sd}Q^{-1}Q_{sd}^T),$$

the first of which would be good for larger values of β, and the second good for values of β near to zero.



(a) Minimum eigenvalue

(b) Maximum eigenvalue

Figure 6.6: Extremal eigenvalues vs β where the control and the state are both discretized using $\mathbf{Q}_1$ elements – $h = 2^{-2}$.

We can get a practical version of $\mathcal{P}_1$ by approximating the Q solves by the Chebyshev semi-iteration and the K_{sd} and K_{sd}^T solves using the multigrid technique described in Section 6.2. Table 6.2 shows the number of iterations, along with the time taken, to solve the system (6.11) for the problem in Example 6.3.1 for various values of β . We have taken $\theta = 2.4$ and $\epsilon = 1/200$.

Table 6.2: Number of iterations taken to solve the system with preconditioner $\mathcal{P}_1$.

h	δ	δ/h_k	time (its)				
			$\beta = 1$	$\beta = 10^{-2}$	$\beta = 10^{-4}$	$\beta = 10^{-6}$	$\beta = 10^{-8}$
2^{-2}	0.166	0.49	0.008 (9)	0.003 (18)	0.002 (21)	0.011 (21)	0.027 (21)
2^{-3}	0.081	0.48	0.018 (11)	0.040 (33)	0.10 (97)	0.12 (117)	0.15 (123)
2^{-4}	0.039	0.46	0.073 (12)	0.28 (43)	1.26 (203)	2.02 (353)	2.50 (410)
2^{-5}	0.017	0.42	0.38 (12)	1.42 (47)	8.73 (281)	(> 500)	(> 500)
2^{-6}	0.0072	0.34	2.10 (12)	8.22 (49)	57.1 (324)	(> 500)	(> 500)

We can see from the results that this preconditioner is good for the larger values of β, but loses optimality faster than the corresponding preconditioner did in the Poisson case. The preconditioner $\mathcal{P}_2$, on the other hand, should behave counter to this. Table 6.3 gives the results for the same problem, using the alternative preconditioner. We see that this also behaves as the theory predicts. Note that we rely on a direct method to solve the Q_{sd} blocks in the results presented here, as we don't have a spectrally equivalent operator for this block. We therefore lose the linear time scaling with the problem size that we had in Chapter 5.

Table 6.3: Number of iterations taken to solve the system with preconditioner $\mathcal{P}_2$.

h	δ	δ/h_k	time (its)				
			$\beta = 1$	$\beta = 10^{-2}$	$\beta = 10^{-4}$	$\beta = 10^{-6}$	$\beta = 10^{-8}$
2^{-2}	0.166	0.49	0.002 (19)	0.009 (18)	0.009 (7)	0.0006 (3)	0.008 (3)
2^{-3}	0.081	0.48	0.16 (123)	0.059 (54)	0.015 (11)	0.008 (5)	0.004 (3)
2^{-4}	0.039	0.46	(> 500)	0.79 (147)	0.11 (19)	0.03 (5)	0.02 (3)
2^{-5}	0.017	0.42	(> 500)	11.9 (400)	1.27 (43)	0.22 (7)	0.11 (3)
2^{-6}	0.0072	0.34	(> 500)	(> 500)	18.1 (111)	1.90 (11)	0.78 (4)

We saw in the previous section that the optimize-then-discretize approach seemed to give a better approximation to the continuous solution, so we now turn our attention to the linear system (6.10). The big difference here is that the matrix is no longer symmetric, so we cannot use MINRES. Instead we use the non-symmetric equivalent, GMRES. The result by Murphy, Golub and Wathen [87] tells us that even in the non-symmetric case, the block diagonal preconditioner with the exact Schur complement will have three eigenvalues. Hence – as long as our approximations to the ideal case are good enough – our approximate preconditioner should have good convergence.

We therefore consider the analogues to $\mathcal{P}_1$ and $\mathcal{P}_2$ above,

$$\mathcal{M}_1 := \texttt{blkdiag}(\beta Q, R_{sd}, K_{sd}R_{sd}^{-1}L_{sd}), \text{ and } \mathcal{M}_2 := \texttt{blkdiag}(\beta Q, R_{sd}, \frac{1}{\beta}Q_{sd}).$$

The results using these two preconditioners are shown in Tables 6.4 and 6.5. Here we see essentially the same behaviour as in the discretize-then-optimize case, so the lack of symmetry doesn't seem to affect the preconditioner too much. The main difference is with $\mathcal{M}_2$ for the smallest value of β, where the iteration doesn't converge. This is probably due to the fact that the eigenvalues aren't the only thing that effects convergence of GMRES, as described in Section 3.1.8.

We have demonstrated preconditioners that are effective for both large and small values of β. For the values in the middle, it seems that if we could combine these preconditioners, we might get a good convergence. For a symmetric positive definite matrix, Bridson and Greif [25] have derived a 'multipreconditioned' conjugate gradient algorithm, which has exactly this property. An equivalent multipreconditioned GMRES algorithm – if such a method could be constructed – may be the right method for dealing with these mid-range β. This would be a possible direction for further research in this area.

Table 6.4: Number of iterations taken to solve the system given by the optimize then discretize approach with GMRES (without restarts) and preconditioner $\mathcal{M}_1$.

h	δ	δ/h_k	time (its)				
			$\beta = 1$	$\beta = 10^{-2}$	$\beta = 10^{-4}$	$\beta = 10^{-6}$	$\beta = 10^{-8}$
2^{-2}	0.166	0.49	0.002 (9)	0.004 (19)	0.007 (19)	0.009 (19)	0.0087 (19)
2^{-3}	0.081	0.48	0.023 (12)	0.067 (34)	0.13 (74)	0.14 (83)	0.15 (89)
2^{-4}	0.039	0.46	0.160 (14)	0.40 (46)	1.15 (144)	1.75 (199)	2.05 (226)
2^{-5}	0.017	0.42	1.66 (14)	3.59 (52)	9.17 (192)	17.6 (384)	(> 500)
2^{-6}	0.0072	0.34	39.9 (16)	64.72 (54)	85.9 (214)	(> 500)	(> 500)

Table 6.5: Number of iterations taken to solve the system given by the optimize then discretize approach with GMRES (without restarts) with preconditioner $\mathcal{M}_2$.

h	δ	δ/h_k	time (its)				
			$\beta = 1$	$\beta = 10^{-2}$	$\beta = 10^{-4}$	$\beta = 10^{-6}$	$\beta = 10^{-8}$
2^{-2}	0.166	0.49	0.024 (19)	0.007 (19)	0.007 (12)	0.017 (9)	0.006 (7)
2^{-3}	0.081	0.48	0.16 (199)	0.10 (77)	0.032 (21)	0.021 (11)	0.020 (9)
2^{-4}	0.039	0.46	3.21 (433)	1.40 (211)	0.033 (21)	0.15 (15)	0.10 (9)
2^{-5}	0.017	0.42	(> 500)	(> 500)	3.86 (99)	2.24 (21)	1.49 (11)
2^{-6}	0.0072	0.34	(> 500)	(> 500)	79.8 (265)	35.0 (37)	(> 500)

6.5 Comments

In this chapter we looked at control of a non-self adjoint PDE, the convection diffusion equation. We explored two options for solving the problem numerically – optimize then discretize and discretize then optimize. The PDE was discretized using the SUPG method. Numerical results seemed to suggest that the (strongly consistent) optimize-then-discretize method is the better option. This method leads to a non-symmetric discrete optimality system, and so we solve it using GMRES.

The choice of Schur complement approximation is less clear in this case, and we give two possibilities. If this includes the discrete PDE then care must be taken to use a multigrid routine that takes into account any layers; we use a scheme which was described by Ramage [93].

We give numerical results for solving the system using MINRES and GMRES with a block diagonal preconditioner. These show that – apart from very small β and $\beta \approx 10^{-4}$ – this describes a practical method. Note that for the optimize-then-discretize case we cannot use the non-standard CG method of Section 4.3 as we do not know the required spectral information.

Chapter 7

Preconditioners for the optimal control of the Stokes equations

7.1 The Stokes Equations

The Stokes equations describe the motion of an incompressible viscous fluid in the case where the velocity is small, or the flow is tightly confined. The equations, in the case where the viscosity is scaled to be unity, are given by

$$-\nabla^2 \vec{u} + \nabla p = \vec{f} \tag{7.1}$$

$$\nabla \cdot \vec{u} = 0, \tag{7.2}$$

with appropriate boundary conditions. Here $\vec{u}$ is a vector field which denotes the velocity of the fluid, p is a scalar field which represents the pressure, and $\vec{f}$ is an external force, which causes an acceleration of the flow. The first equation represents conservation of momentum, while the second represents conservation of mass, or incompressiblilty of the velocity. For more details on the derivation of the equation see, for example, [39, Chapter 0]. We only consider the two-dimensional case here, for simplicity, although everything should generalize to three-dimensions in the obvious way.

We supplement (7.1-7.2) with boundary conditions of the form

$$\vec{u} = \vec{w} \text{ on } \partial\Omega_D, \qquad \frac{\partial \vec{u}}{\partial n} - \vec{n}p = \vec{s} \text{ on } \partial\Omega_N, \tag{7.3}$$

where $\partial\Omega_N \cup \partial\Omega_D = \partial\Omega$, $\partial\Omega_N \cap \partial\Omega_D = \emptyset$, $\vec{n}$ denotes the outward-pointing normal to $\partial\Omega$ and $\frac{\partial \vec{u}}{\partial n}$ is the directional derivative in the normal direction. As with the Poisson equation, these are called Dirichlet and Neumann boundary conditions respectively.

We must assume that $\int_{\partial\Omega_D} \text{ds} \neq 0$ in order to guarantee a unique velocity solution. Also, if the velocity is prescribed everywhere on the boundary, so $\partial\Omega_N = \emptyset$, then the

pressure p is only unique up to a constant, which is called the hydrostatic pressure level. By integrating (7.2) and applying the Divergence theorem we see that in this case it is necessary that

$$\int_\Omega \nabla \cdot \vec{u} = \int_{\partial\Omega} \vec{w} \cdot \vec{n} = 0.$$

If we divide the boundary up as

$$\begin{aligned}
\partial\Omega_+ &= \{x \in \partial\Omega \ : \ \vec{w} \cdot \vec{n} > 0\}, \\
\partial\Omega_0 &= \{x \in \partial\Omega \ : \ \vec{w} \cdot \vec{n} = 0\}, \\
\partial\Omega_- &= \{x \in \partial\Omega \ : \ \vec{w} \cdot \vec{n} < 0\},
\end{aligned}$$

then the condition above tells us that

$$\int_{\partial\Omega_+} \vec{w} \cdot \vec{n} + \int_{\partial\Omega_-} \vec{w} \cdot \vec{n} = 0. \tag{7.4}$$

This simply says that the volume of fluid entering the domain must equal the fluid leaving the domain. If this condition is not satisfied, then the Stokes equations have no solution. A flow such that $\vec{w} \cdot \vec{n} = 0$ on $\partial\Omega$ automatically satisfies this condition, and such a problem is known as enclosed flow. Example 7.1.1 is of this type.

Example 7.1.1. *Let $\Omega = [0,1]^2$, and let $\vec{\mathbf{i}}$ and $\vec{\mathbf{j}}$ denote unit vectors in the direction of the x and y axis respectively. Let $\vec{u}$ and p satisfy the Stokes equations (7.1 - 7.2) with $\vec{f} = \vec{0}$, and let $\vec{u} = \vec{0}$ on the boundary except for on $x = 1$, $0 \leq y \leq 1$, where $\vec{u} = -\vec{\mathbf{j}}$.*

Example 7.1.1 is a classic test problem in fluid dynamics called leaky cavity flow. Figure 7.1 shows the (exponentially distributed) streamlines and pressure for this problem. The flow has small recirculations – called Moffatt eddies – in the corners of the boundary at $x = 0$, which are counterrotating relative to the main flow. The discontinuity of the velocity at $x = 1,\ y = 1$ and $x = 1,\ y = 0$ gives rise to singularities in the pressure at these points.

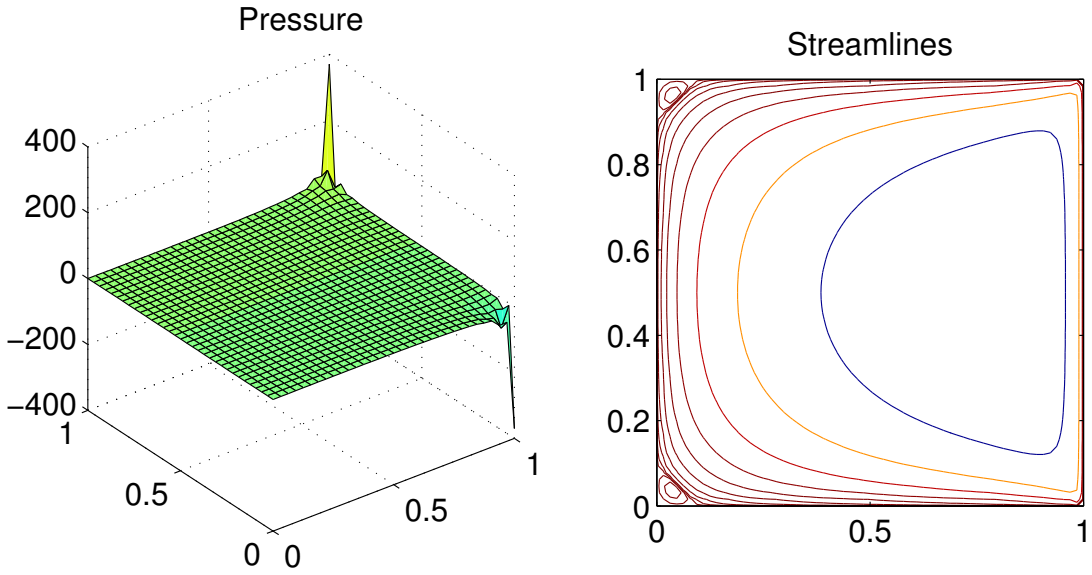
Similar results are given by solving the related problems, watertight cavity flow, where

$$\vec{u} = -\vec{\mathbf{j}} \ \text{ on } \ x = 1,\ 0 < y < 1,$$

and regularized cavity flow, where

$$\vec{u} = (y^4 - 1)\vec{\mathbf{j}} \ \text{ on } \ x = 1,\ 0 < y < 1.$$

Figure 7.1: Pressure and streamlines for Example 7.1.1



For the rest of the section, we consider purely Dirichlet boundary conditions, so $\partial\Omega_D = \partial\Omega$ in (7.3). The treatment in the mixed case is similar – see, for example, [39, Chapter 5] for details. Suppose $\vec{u} \in \mathbb{R}^2$, and define the solution and test spaces

$$\mathbf{H}^1_E := \{\vec{u} \in \mathcal{H}^1(\Omega)^2 \ : \ \vec{u} = \vec{w} \text{ on } \partial\Omega\},$$
$$\mathbf{H}^1_0 := \{\vec{u} \in \mathcal{H}^1(\Omega)^2 \ : \ \vec{u} = \vec{0} \text{ on } \partial\Omega\}.$$

Recall that in the case of the pure Dirichlet problem the pressure is only defined up to a constant, which is usually fixed by enforcing the normalization

$$\int_\Omega p = 0.$$

Let $M = L^{2,0}(\Omega) := \{f \in L^2(\Omega) \ : \ \int_\Omega f = 0\}$. Then the weak formulation of (7.1-7.2), along with (7.3), is: Find $\vec{u} \in \mathbf{H}^1_E$ and $p \in M$ such that

$$\int_\Omega \nabla\vec{u} : \nabla\vec{v} - \int_\Omega p\nabla\cdot\vec{v} = \int_\Omega \vec{f}\cdot\vec{v} \ \text{ for all } \vec{v} \in \mathbf{H}^1_0,$$
$$\int_\Omega q\nabla\cdot\vec{u} = 0 \ \text{ for all } q \in M.$$

Note that here $\nabla\vec{u} : \nabla\vec{v} := \sum_{i,j} \frac{\partial\vec{u}_i}{\partial x_j}\frac{\partial\vec{v}_i}{\partial x_j}$, the componentwise scalar product. An application of the divergence theorem to the second equation shows that the addition of a constant to q does not change the equation; we can therefore identify M with the quotient space $L^2(\Omega)/\mathbb{R}$.

It can be shown – see, for example, Braess [19, Chapter III] – that weak solutions are uniquely defined when an inf-sup condition of the form

$$\inf_{q \neq \text{constant}} \sup_{\vec{v} \neq \vec{0}} \frac{| \langle q, \nabla \cdot \vec{v} \rangle |}{\|\vec{v}\|_{1,\Omega} \|q\|_{0,\Omega}} \geq \gamma, \tag{7.5}$$

is satisfied for some $\gamma > 0$. Here $\|\vec{v}\|_{1,\Omega}^2 = (\int_\Omega \vec{v} \cdot \vec{v} + \nabla \vec{v} : \nabla \vec{v})$ is a norm for functions in $\mathbf{H}_0^1$ and $\|q\|_{0,\Omega}^2 = \int_\Omega q^2$.

When developing finite-element approximations to the Stokes equations one must ensure that an analogous inf-sup condition is satisfied in finite dimensions. In particular, a discrete version of (7.5) must hold, and so the discretization spaces must be chosen carefully. In particular, the choice of basis used to discretize $\vec{u}$ and p will not be independent.

The discretization of (7.1-7.2) is usually done by mixed finite elements. Let $\mathbf{X}_0^h = \text{span}\{\vec{\phi}_j\} \subset \mathbf{H}_0^1,\ j = 1, \ldots, n_u$ and $M^h = \text{span}\{\psi_k\} \subset M,\ k = 1, \ldots, n_p$. Then, as in Section 2.1, let us approximate $\vec{u}$ and p by $\vec{u}_h$ and p_h, where

$$\vec{u}_h = \sum_{j=1}^{n_u} U_j \vec{\phi}_j + \sum_{j=n_u+1}^{n_u+n_\partial} U_j \vec{\phi}_j$$
$$p_h = \sum_{k=1}^{n_p} P_k \psi_k,$$

where the U_j for $j = n_u + 1, \ldots, n_u + n_\partial$ are chosen to interpolate the boundary data. The Galerkin approach then leads to a linear system of the form

$$\begin{bmatrix} \underline{K} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{7.6}$$

where

$$\underline{K} = [k_{i,j}] \in \mathbb{R}^{n_u \times n_u},\ k_{i,j} = \int_\Omega \nabla \vec{\phi}_i : \nabla \vec{\phi}_j,$$
$$B = [b_{k,j}] \in \mathbb{R}^{n_p \times n_u},\ b_{k,j} = -\int_\Omega \psi_k \nabla \cdot \vec{\phi}_j,$$
$$\mathbf{f} = [f_i] \in \mathbb{R}^{n_u},\ f_i = \int_\Omega \vec{f} \cdot \vec{\phi} - \sum_{j=n_u+1}^{n_u+n_\partial} U_j \int_\Omega \nabla \vec{\phi}_i : \nabla \vec{\phi}_j,$$
$$\mathbf{g} = [g_k] \in \mathbb{R}^{n_p},\ g_k = \sum_{j=n_u+1}^{n_u+n_\partial} U_j \int_\Omega \psi_k \nabla \cdot \vec{\phi}_j.$$

Note that one usually uses the same finite element space to discretize the two components of velocity. Let $\{\phi_i\}$, $i = 1, \ldots, n$ be a set of finite element basis functions,

where $2n = n_u$. Then if we use the natural splitting

$$\{\vec{\phi}_1, \ldots, \vec{\phi}_{n_u}\} := \{(\phi_1, 0)^T, \ldots, (\phi_n, 0)^T, (0, \phi_1)^T, \ldots, (0, \phi_n)^T\},$$

we can rewrite (7.6) as

$$\begin{bmatrix} K & 0 & B_x^T \\ 0 & K & B_y^T \\ B_x & B_y & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_\mathbf{x} \\ \tilde{\mathbf{u}}_\mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_\mathbf{x} \\ \mathbf{f}_\mathbf{y} \\ \mathbf{g} \end{bmatrix},$$

where

$$K = [k_{i,j}] \in \mathbb{R}^{n \times n},\ k_{i,j} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j,$$
$$B_x = [{b^x}_{k,i}] \in \mathbb{R}^{n_p \times n},\ {b^x}_{k,i} = -\int_\Omega \psi_k \frac{\partial \phi_i}{\partial x},$$
$$B_y = [{b^y}_{k,j}] \in \mathbb{R}^{n_p \times n},\ {b^y}_{k,j} = -\int_\Omega \psi_k \frac{\partial \phi_j}{\partial y},$$

i.e. $\underline{K} = \texttt{blkdiag}(K, K)$, where K is the usual stiffness matrix from Section 2.1.

As alluded to above, care must be taken when choosing the finite element approximation spaces. In particular, in order for the solvability condition to be satisfied, we require that our discretization satisfies $\text{null}(B^T) = \{\mathbf{1}\}$ [39, pp. 226-227].

Since $\mathbf{1} \in \text{null}(B^T)$, the discrete Stokes system is singular. For a solution to exist we must have the compatibility condition on the right hand side in (7.6) that $\mathbf{g}^T\mathbf{1} = 0$. This is just a discrete representation of conservation of mass (7.4) – see [39, pp. 227-228] for a discussion.

It is also required to pick approximation spaces such that for all possible grids, there exists a constant $\gamma > 0$ such that

$$\min_{q_h \neq \text{constant}} \max_{\vec{v}_h \neq \vec{0}} \frac{|\langle q_h, \nabla \cdot \vec{v}_h \rangle|}{\|\vec{v}_h\|_{1,\Omega} \|q_h\|_{0,\Omega}} \geq \gamma. \tag{7.7}$$

For more details see e.g. Elman, Silvester and Wathen [39, Section 5.3]. Such a condition is satisfied if we used $\mathbf{Q}_2$ elements to discretize the velocity, and $\mathbf{Q}_1$ elements to discretize the pressure. This scheme is known as Taylor-Hood or $\mathbf{Q}_2 - \mathbf{Q}_1$ discretization, and is the discretization we will use in the remainder of this chapter.

7.2 Iterative solution of the discrete Stokes equations

Note that the discrete Stokes equations (7.6) form a saddle point system, as defined in Chapter 4. Therefore we can apply the algorithms described in that chapter for

solving this equation. In particular, we will look at solving the system using MINRES with a block-diagonal preconditioner of the form

$$\mathcal{P} = \begin{bmatrix} \underline{K}_0 & 0 \\ 0 & S_0 \end{bmatrix}, \tag{7.8}$$

where $\underline{K}_0$ approximates $\underline{K}$ and S_0 approximates the Schur complement, $B\underline{K}^{-1}B^T$.

Since $\underline{K}$ is just a block-diagonal matrix comprised of two discrete Laplacian matrices a multigrid method – as described in Section 3.2 – would provide a good approximate inverse. As previously, let ρ be the contraction of a multigrid method. Then if we let $\widetilde{\underline{K}}_m$ denote m cycles, a simple extension of (3.35) tells us that for the generalized Rayleigh quotient, the bounds

$$1 - \rho^m \leq \frac{\mathbf{v}^T \underline{K} \mathbf{v}}{\mathbf{v}^T \widetilde{\underline{K}}_m \mathbf{v}} \leq 1 + \rho^m \tag{7.9}$$

hold for all $\mathbf{v} \in \mathbb{R}^{n_u}$.

We just need a good approximation to the Schur complement. The approach described below is due to Silvester and Wathen [127, 106], and uses the inf-sup condition. Recall that

$$\|\vec{v}_h\|_{1,\Omega}^2 = \int_\Omega \vec{v}_h \cdot \vec{v}_h + \nabla \vec{v}_h : \nabla \vec{v}_h \geq \int_\Omega \nabla \vec{v}_h : \nabla \vec{v}_h = \|\nabla \vec{v}_h\|_{0,\Omega}^2.$$

Therefore if the discrete inf-sup condition (7.7) holds, we must also have

$$\min_{q_h \neq \text{constant}} \max_{\vec{v}_h \neq \vec{0}} \frac{|\langle q_h, \nabla \cdot \vec{v}_h \rangle|}{\|\nabla \vec{v}_h\|_{0,\Omega} \|q_h\|_{0,\Omega}} \geq \gamma. \tag{7.10}$$

As in Section 2.1, $\|\nabla \vec{v}_h\|_{0,\Omega}^2 = \mathbf{v}^T \underline{K} \mathbf{v}$ and $\|q_h\|_{0,\Omega}^2 = \mathbf{q}^T Q_p \mathbf{q}$, where $\mathbf{v}$ and $\mathbf{q}$ are the vectors of coefficients of the expansion of v_h and q_h in their respective finite element bases and Q_p is the pressure mass matrix. Also, $|\langle q_h, \nabla \cdot \vec{v}_h \rangle| = |\mathbf{q}^T B \mathbf{v}|$.

We can now write the inf-sup condition (7.10) in terms of vectors as

$$\begin{aligned} \gamma^2 &\leq \min_{\mathbf{q} \neq \mathbf{1}} \max_{\mathbf{v} \neq \mathbf{0}} \frac{(\mathbf{q}^T B \mathbf{v})^2}{(\mathbf{v}^T \underline{K} \mathbf{v})\,(\mathbf{q}^T Q_p \mathbf{q})} \\ &= \min_{\mathbf{q} \neq \mathbf{1}} \frac{1}{\mathbf{q}^T Q_p \mathbf{q}} \max_{\mathbf{w} = \underline{K}^{1/2}\mathbf{v} \neq \mathbf{0}} \frac{(\mathbf{q}^T B \underline{K}^{-1/2} \mathbf{w})^2}{\mathbf{w}^T \mathbf{w}} \\ &= \min_{\mathbf{q} \neq \mathbf{1}} \frac{1}{\mathbf{q}^T Q_p \mathbf{q}} \max_{\mathbf{w} \neq \mathbf{0}} \frac{((\underline{K}^{-1/2} B^T \mathbf{q})^T \mathbf{w})^2}{\mathbf{w}^T \mathbf{w}}. \end{aligned}$$

The maximum here is attained when $\mathbf{w} = \underline{K}^{-1/2} B^T \mathbf{q}$, so we have

$$\gamma^2 \leq \min_{\mathbf{q} \neq \mathbf{1}} \frac{\mathbf{q}^T B \underline{K}^{-1} B^T \mathbf{q}}{\mathbf{q}^T Q_p \mathbf{q}}. \tag{7.11}$$

Now consider the generalized eigenvalue problem

$$\begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{q} \end{bmatrix} = \sigma \begin{bmatrix} \underline{K} & 0 \\ 0 & Q_p \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{q} \end{bmatrix}. \tag{7.12}$$

It is clear that if $\mathbf{v}$ and $\mathbf{q}$ lie in the null space of B and B^T respectively, then $\sigma = 0$. Now suppose $\mathbf{v} \notin \text{null}(B)$ and $\mathbf{q} \notin \text{null}(B^T)$. Then left-multiplying with $[\mathbf{v}^T \ -\mathbf{q}^T]$ gives us

$$\sigma(\mathbf{v}^T \underline{K}\mathbf{v} - \mathbf{q}^T Q_p \mathbf{q}) = 0,$$

since $\mathbf{v}^T B^T \mathbf{q} - \mathbf{q}^T B\mathbf{v} = 0$. This tells us that if $\sigma \neq 0$, then $\mathbf{v}^T \underline{K}\mathbf{v} = \mathbf{q}^T Q_p \mathbf{q}$. We can write (7.12) as

$$\begin{bmatrix} \sigma\underline{K} & B^T \\ B & \sigma Q_p \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Then if $\sigma \neq 0$, we can perform block-Gaussian elimination, and we can obtain the following relations:

$$(\sigma Q_p - \frac{1}{\sigma} B\underline{K}^{-1} B^T)\mathbf{q} = \mathbf{0}$$
$$(\sigma\underline{K} - \frac{1}{\sigma} B^T Q_p^{-1} B)\mathbf{v} = \mathbf{0}.$$

These can be written in terms of a Rayleigh quotient as

$$\frac{\mathbf{q}^T B\underline{K}^{-1} B^T \mathbf{q}}{\mathbf{q}^T Q_p \mathbf{q}} = \sigma^2 = \frac{\mathbf{v}^T B^T Q_p^{-1} B\mathbf{v}}{\mathbf{v}^T \underline{K}\mathbf{v}}. \tag{7.13}$$

Comparing this with (7.11), and noting that $\text{null}(B) = \mathbf{1}$, as the pressure is unique up to a constant, we see that $\gamma = \sigma_{\min}$, and moreover we can write another characterization of the inf-sup condition as

$$\gamma^2 \leq \min_{\{\mathbf{v}\in\mathbb{R}^{nu} \ : \ \mathbf{v}^T\underline{K}\mathbf{u}=0, \ \mathbf{u}\in\text{null}(B)\}} \frac{\mathbf{v}^T B^T Q_p^{-1} B\mathbf{v}}{\mathbf{v}^T \underline{K}\mathbf{v}}. \tag{7.14}$$

The value of γ clearly depends upon the discretization used. For $\mathbf{Q}_2 - \mathbf{Q}_1$ elements on uniform grids, $\gamma^2 = 0.2$. See [39, p. 271] for more details. The more stable the elements used, the closer to unity this bound will be.

We have just found a lower bound on the eigenvalues of $Q_p^{-1}(B\underline{K}^{-1}B^T)$. If we can show that they are also bounded above, then Q_p will be shown to have been a good approximation to the Schur complement. Note that an application of Cauchy-Schwarz gives

$$|\langle q_h, \nabla \cdot \vec{v}_h \rangle| \leq \|q_h\| \|\nabla \cdot \vec{v}_h\|.$$

For all $\vec{v}_h \in \mathbf{H}_0^1$,

$$\|\nabla \vec{v}_h\|^2 = \|\nabla \cdot \vec{v}_h\|^2 + \|\nabla \wedge \vec{v}_h\| \geq \|\nabla \cdot \vec{v}_h\|^2,$$

hence

$$\frac{|\langle q_h, \nabla \cdot \vec{v}_h \rangle|}{\|\nabla \vec{v}_h\| \|q_h\|} \leq 1.$$

Writing this in terms of matrices and vectors, and combining with (7.11), we have shown that there exist constants γ^2 and Γ^2 such that

$$\gamma^2 \leq \frac{\mathbf{q}^T B \underline{K}^{-1} B^T \mathbf{q}}{\mathbf{q}^T Q_p \mathbf{q}} \leq \Gamma^2, \tag{7.15}$$

which holds for all $\mathbf{q} \in \mathbb{R}^{n_p}$ such that $\mathbf{q} \neq c\mathbf{1}$, for any scalar c. Using (7.13), we can alternatively write this as

$$\gamma^2 \leq \frac{\mathbf{v}^T B^T Q_p^{-1} B \mathbf{v}}{\mathbf{v}^T \underline{K} \mathbf{v}} \leq \Gamma^2, \tag{7.16}$$

which holds for all $\mathbf{v} \in \mathbb{R}^{n_u}$ such that $\mathbf{v}^T \underline{K} \mathbf{u} = 0$ if $\mathbf{u} \in \text{null}(B)$. Note that, due to hydrostatic pressure $\mathbf{q} = 1$ being in the null space of the Schur complement, there will always be an eigenvalue at 0. As long as our starting vector in an iterative method starts with no component in this space – i.e. $\mathbf{x}^{(0)}$ satisfies the discrete compatibility condition – then we will always remain orthogonal to this space (in exact arithmetic), and hence we can ignore the zero eigenvalue.

The mass matrix Q_p is therefore spectrally equivalent to the Schur complement. However, in order to be an effective preconditioner, (7.8) must be easy to invert, and so we must approximate Q_p. Elman, Silvester and Wathen [39] suggest taking it's diagonal, but we can make the approximation better by taking a fixed number of steps of the Chebyshev semi-iteration, as described in Section 5.2. As before, let C_k^{-1} denote the application of k such steps. Then we get that C_m satisfies

$$\delta_k \gamma^2 \leq \frac{\mathbf{q}^T B \underline{K}^{-1} B^T \mathbf{q}}{\mathbf{q}^T C_m \mathbf{q}} \leq \Delta_k \Gamma^2 \quad \forall\ \mathbf{q} \in \mathbb{R}^{n_p} \text{ s.t. } \mathbf{q} \neq \mathbf{1} \tag{7.17}$$

for some constants δ_k, Δ_k independent of h, and which both approach unity as k grows – see Table 5.2.

The above theory suggests that

$$\mathcal{P} = \begin{bmatrix} \widetilde{\underline{K}_m} & 0 \\ 0 & C_k \end{bmatrix},$$

would be a reasonable choice. We can now apply Theorem 4.2.1 – with the obvious modification – along with (7.9) and (7.17), to get that the preconditioned system has one eigenvalue at zero, and the remaining eigenvalues satisfy

$$\frac{\theta - \sqrt{\theta^2 + 4\Theta\Delta_k\Gamma^2}}{2} \leq \lambda \leq \frac{\Theta - \sqrt{\Theta^2 + 4\delta_k\gamma^2\theta}}{2},$$
$$\theta \leq \lambda \leq \Theta,$$
$$\text{or} \qquad \frac{\theta + \sqrt{\theta^2 + 4\theta\delta_k\gamma^2}}{2} \leq \lambda \leq \frac{\Theta + \sqrt{\Theta^2 + 4\Delta_k\Gamma^2\Theta}}{2},$$

where $\theta = 1 - \rho^m$ and $\Theta = 1 + \rho^m$.

Table 7.1 shows the time and number of iterations taken to solve the discrete Stokes system using MINRES with the preconditioner described above. Here we have used five steps of the Chebyshev semi-iteration to approximate Q, and one V-cycle of the HSL MI20 algebraic multigrid routine with the default parameters to approximate the stiffness matrix.

Table 7.1: Solving the Stokes equations using MINRES with the Silvester-Wathen preconditioner

h	time	its
2^{-2}	0.015	25
2^{-3}	0.029	27
2^{-4}	0.076	28
2^{-5}	0.349	30
2^{-6}	1.504	30
2^{-7}	6.616	30

7.3 Optimal control of the Stokes equations

We now want to turn our attention to problems in optimal control, where the PDE constraints are Stokes equations. We will only consider distributed control here, although – as in Chapter 5 – boundary control could be treated in the same way. One way of formulating the Stokes control problem is the following:

$$\min_{v,p,u} \frac{1}{2}\|\vec{v} - \widehat{\vec{v}}\|^2_{L^2(\Omega)} + \frac{\delta}{2}\|p - \hat{p}\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|\vec{u}\|^2_{L^2(\Omega)} \tag{7.18}$$

$$\text{s.t.} -\nabla^2\vec{v} + \nabla p = \vec{u} \quad \text{in } \Omega$$
$$\nabla \cdot \vec{v} = 0 \quad \text{in } \Omega,$$
$$\vec{v} = \vec{w} \quad \text{on } \partial\Omega$$

where $\vec{v}$ is the velocity field, p is the pressure, $\vec{w}$ is the prescribed velocity on the boundary, and $\vec{u}$ is the control field. A constant δ is added in front of the desired pressure to enable us to penalize the pressure, allowing us to stop it from getting too large: usually $\hat{p} = 0$ would be selected. Note here that the control is applied only to the momentum equation, as incompressibility could not generally be controlled in most flow problems.

We use the discretize-then-optimize technique here, although as the Stokes equations are self-adjoint we will get the same system if we optimize-then-discretize and choose to discretize the the variables in the same way.

As in the previous section, let $\{\vec{\phi}_j\},\ j = 1, \ldots, n_v$ and $\{\psi_k\},\ k = 1, \ldots, n_p$ be sets of finite element basis functions that form a stable mixed approximation for the Stokes equations. Furthermore, let $\{\vec{\chi}_j\},\ j = 1, \ldots, n_u$ be a basis such that $\vec{u}_h = \sum_{i=1}^{n_u} U_i \vec{\chi}_i$ is a discrete approximation to the control, $\vec{u}$. The discrete Stokes equations are of the form

$$\begin{bmatrix} \underline{K} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \widehat{Q} \\ 0 \end{bmatrix} \mathbf{u} + \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix},$$

where $\underline{K}$ and B are as defined above and

$$\widehat{Q} = [q_{i,j}] \in \mathbb{R}^{n_v, n_u},\ q_{i,j} = \int_\Omega \vec{\phi}_i \cdot \vec{\chi}_j$$

$$\mathbf{f} = [f_i] \in \mathbb{R}^{n_v},\ f_i = -\sum_{j=n_u+1}^{n_u+n_\partial} U_j \int_\Omega \nabla\vec{\phi}_i : \nabla\vec{\phi}_j,$$

$$\mathbf{g} = [g_k] \in \mathbb{R}^{n_p},\ g_k = \sum_{j=n_u+1}^{n_u+n_\partial} U_j \int_\Omega \psi_k \nabla \cdot \vec{\phi}_j.$$

Similarly to Section 2.2, the discrete cost functional is equivalent to

$$\min_{\mathbf{v},\mathbf{p},\mathbf{u}} \frac{1}{2}\mathbf{v}^T Q_{\vec{v}} \mathbf{v} - \mathbf{v}^T \mathbf{b} + \frac{\delta}{2}\mathbf{p}^T Q_p \mathbf{p} - \delta \mathbf{p}^T \mathbf{d} + \frac{\beta}{2}\mathbf{u}^T Q_{\vec{u}} \mathbf{u}$$

where Q_s is the mass matrix of the basis used to discretize the variable s and

$$\mathbf{b} = [b_i] \in \mathbb{R}^{n_v},\ b_i = \int_\Omega \widehat{\vec{v}}\vec{\phi}_i,$$

$$\mathbf{d} = [d_k] \in \mathbb{R}^{n_p},\ d_k = \int_\Omega \widehat{p}\psi_k.$$

If we introduce two vectors of Lagrange multipliers, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, then the discrete optimality system is of the form

$$\begin{bmatrix} \beta Q_{\vec{u}} & 0 & 0 & -\widehat{Q}^T & 0 \\ 0 & Q_{\vec{v}} & 0 & \underline{K} & B^T \\ 0 & 0 & \delta Q_p & B & 0 \\ -\widehat{Q} & \underline{K} & B^T & 0 & 0 \\ 0 & B & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{p} \\ \boldsymbol{\lambda} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \\ \delta\mathbf{d} \\ \mathbf{f} \\ \mathbf{g} \end{bmatrix}. \tag{7.19}$$

We can relabel this equation so that it is more recognizably of the same form as in the previous chapters; the system becomes

$$\begin{bmatrix} \beta Q_{\vec{u}} & 0 & -\widehat{\mathcal{Q}}^T \\ 0 & \mathcal{Q} & \mathcal{K} \\ -\widehat{\mathcal{Q}} & \mathcal{K} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \boldsymbol{\xi} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{c} \\ \mathbf{h} \end{bmatrix},$$

where $\mathcal{Q} = \texttt{blkdiag}(Q_{\vec{v}}, \delta Q_p)$, $\mathcal{K} = \begin{bmatrix} K & B^T \\ B & 0 \end{bmatrix}$, $\widehat{\mathcal{Q}} = [\widehat{Q}^T \; 0]^T$ and the vectors $\mathbf{y}$, $\boldsymbol{\xi}$, $\mathbf{c}$ and $\mathbf{h}$ take their obvious definitions.

Let us consider the following example:

Example 7.3.1. *Let $\Omega = [0,1]^2$, and consider an optimal control problem of the form (7.18), with Dirichlet boundary conditions as given in Example 7.1.1 (leaky cavity flow). Take the desired pressure as $\widehat{p} = 0$ and let $\widehat{\vec{v}} = y\vec{\mathbf{i}} - x\vec{\mathbf{j}}$. The exponentially distributed streamlines of the desired velocity are shown in Figure 7.2.*

This problem can be interpreted physically as finding a forcing term which reduces the Moffat eddies present in the solution to Example 7.1.1; such eddies may be associated with cavitation in a real life situation. Figure 7.3 shows the streamlines of the velocity and pressure associated with the optimal control for $\beta = 10^{-2}$, and $h = 2^{-5}$, and Figure 7.4 shows the equivalent plots for $\beta = 10^{-5}$. We plot all streamlines as exponentially distributed so that we can see the behaviour in the corners where the eddies are located.

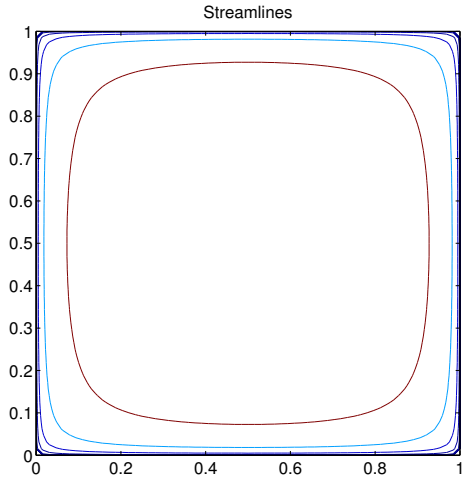
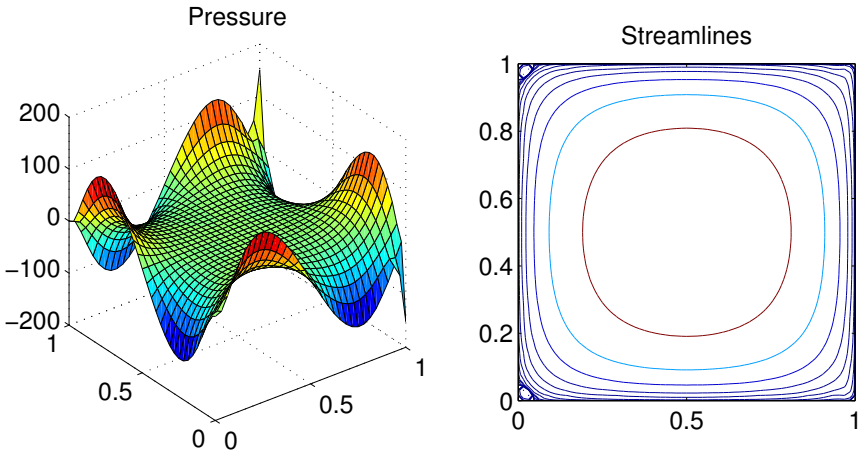


Figure 7.2: Streamlines of $\widehat{\vec{v}}$

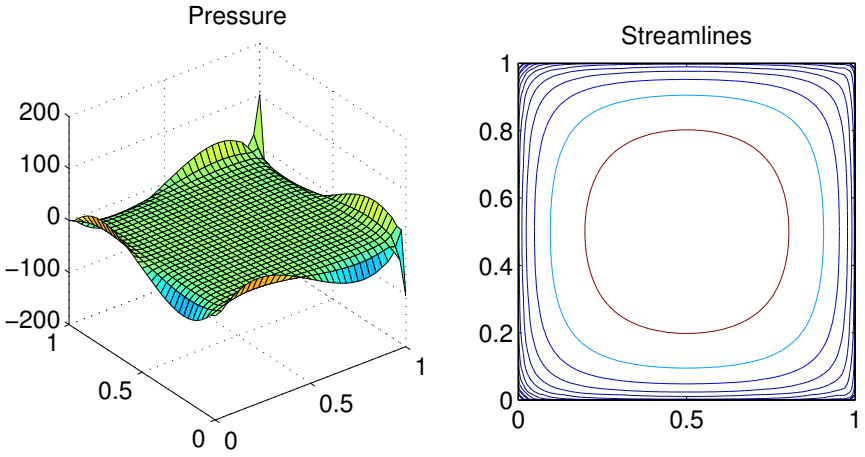
We see in the figures that there appears to be a singularity along the edges in the pressure solution if β is taken too small. Increasing δ does decrease the size of the pressure; $\delta = 1$ seems to correspond to a maximum value at about the level of the peaks in the forward problem.

7.4 Preconditioning the control problem

As in the previous sections, the keys to a good preconditioner are good approximations to the (1,1) block and the Schur complement of the saddle point system. Here the (1,1)



(a) Computed state for $\beta = 10^{-2}$, $\delta = 1$



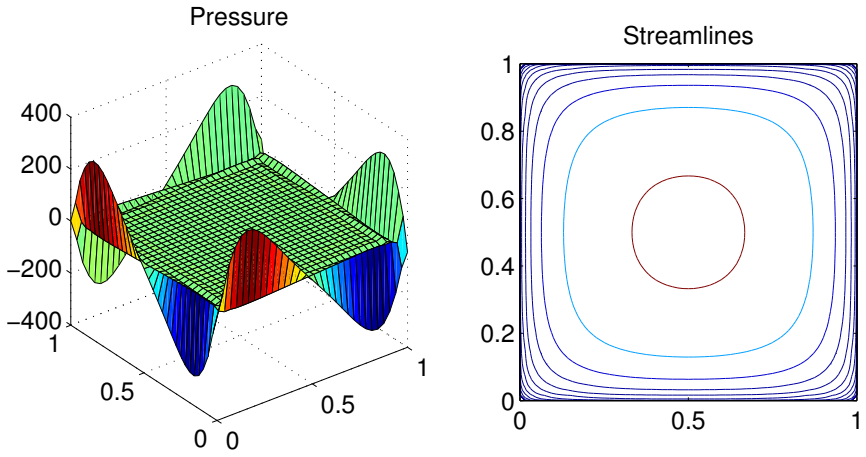
(b) Computed state for $\beta = 10^{-2}$, $\delta = 10$

Figure 7.3: Computed states for Example 7.3.1 in two dimensions, $\beta = 10^{-2}$.

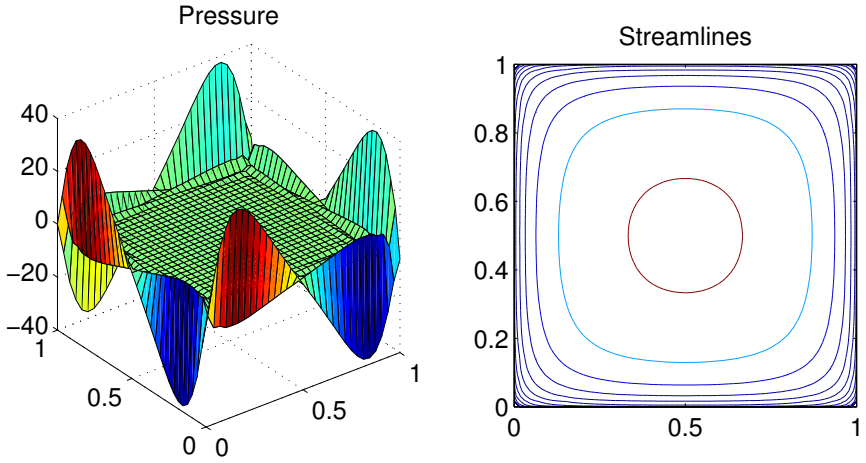
block is $A := \texttt{blkdiag}(\beta Q_{\vec{u}}, \mathcal{Q})$, so as previously we can replace each solve with a mass matrix by a fixed number of steps of the Chebyshev semi-iteration. This will give us a spectrally equivalent operator in the sense that for such an approximation, A_0, there exist constants δ, Δ independent of the mesh size such that for all $\mathbf{x} \in \mathbb{R}^{n_v+n_p+n_u}$,

$$\delta \leq \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T A_0 \mathbf{x}} \leq \Delta.$$

Now consider the Schur complement, $S := \frac{1}{\beta}\widehat{Q}Q_{\vec{v}}^{-1}\widehat{Q}^T + \mathcal{K}\mathcal{Q}^{-1}\mathcal{K}$. As in Section 5.3, $\mathcal{K}\mathcal{Q}^{-1}\mathcal{K}$ is the dominant term in this sum for all but the smallest values of β – this is the term that contains the PDE. Figure 7.5 shows the eigenvalue distribution for this approximation of S for a relatively coarse $\mathbf{Q}_2 - \mathbf{Q}_1$ discretization with $\beta = 10^{-2}$. As we can see from the figure, this clusters the eigenvalues nicely, and so we can expect



(a) Computed state for $\delta = 1$



(b) Computed state for $\delta = 10$

Figure 7.4: Computed states for Example 7.3.1 in two dimensions, $\beta = 10^{-5}$.

good convergence of MINRES if we used this approximation as S_0.

However, a preconditioner must be easy to invert, and solving a system with $\mathcal{K}\mathcal{Q}^{-1}\mathcal{K}$ requires two solves with the discrete Stokes matrix at each iteration, which is not cheap. We therefore would like some matrix, $\widetilde{\mathcal{K}}$, such that $\widetilde{\mathcal{K}}\mathcal{Q}^{-1}\widetilde{\mathcal{K}}$ approximates $\mathcal{K}\mathcal{Q}^{-1}\mathcal{K}$. Note, as in Section 5.2, the mass matrices are not important here and it is sufficient that $\widetilde{\mathcal{K}}\widetilde{\mathcal{K}}^T$ approximates $\mathcal{K}^2$.

In order to find such an approximation we again turn to the result of Braess and Peisker, as described in Section 5.3. Recall that it is not sufficient that $\widetilde{\mathcal{K}}$ approximates $\mathcal{K}$. Indeed, as we saw in the previous section, Silvester and Wathen [106, 127] showed that an ideal preconditioner is $\texttt{blkdiag}(\underline{K}, M_p)$, where $\underline{K}$ is a multigrid cycle, but the eigenvalues of $(\widehat{\mathcal{K}}\widehat{\mathcal{K}}^T)^{-1}\mathcal{K}^2$ are not at all clustered, and the approximation for the

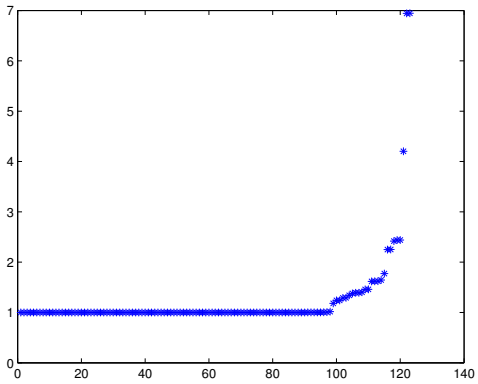


Figure 7.5: Eigenvalues of $(\mathcal{K}\mathcal{Q}^{-1}\mathcal{K})^{-1}S$

Schur complement of the control problem is a poor one in this case. Table 7.2 shows the number of MINRES iterations taken using this approximation – with exact solves. Clearly this is not a good method.

Table 7.2: Comparison of solution methods for solving Example 7.3.1 using MINRES with the block diagonal preconditioner, where $\mathcal{K}$ is approximated by the block diagonal Silvester Wathen preconditioner.

h	size	time	its
2^{-2}	344	0.526	242
2^{-3}	1512	10.668	1320
2^{-4}	6344	217.240	5172

Suppose we wish to solve the equation $\mathcal{K}\mathbf{x} = \mathbf{b}$, for some right hand side vector $\mathbf{b}$. The theory in Section 5.3 shows us that if we take an approximation $\mathcal{K}_m$ which is implicitly defined by an iteration such that $\mathbf{x}^{(m)} = \mathcal{K}_m^{-1}\mathbf{b}$, say, which converges to the solution $\mathbf{x}$ in the sense that

$$\|\mathbf{x}^{(m)} - \mathbf{x}\| \leq \eta_m \|\mathbf{x}\|,$$

then $\eta_m = \|\mathcal{K}_m^{-1}\mathcal{K} - I\|$, and we have

$$(1 - \eta_m)^2 \leq \frac{\mathbf{x}^T\mathcal{K}^2\mathbf{x}}{\mathbf{x}^T\mathcal{K}_m^T\mathcal{K}_m\mathbf{x}} \leq (1 + \eta_m)^2.$$

Hence, such an approximation is suitable for use in this case.

Note that an inner iteration of MINRES cannot be used to approximate $\mathcal{K}$, unless run until convergence, since – like CG – MINRES is a Krylov subspace method, and hence nonlinear. We would therefore have to use a flexible outer method if we were to

make this approximation. Therefore we consider a stationary method – in particular, the inexact Uzawa method of Section 4.1. Recall that this is a Richardson iteration based on a splitting matrix of the form

$$\mathcal{M} = \begin{bmatrix} \underline{K}_0 & 0 \\ B & -\tau Q_0 \end{bmatrix},$$

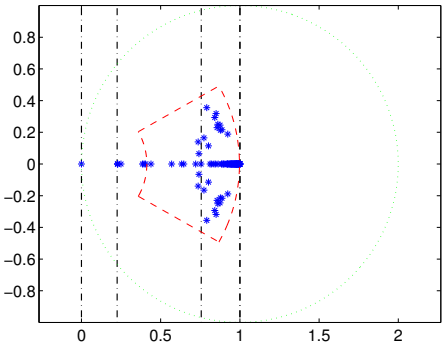
where $\underline{K}_0$ approximates $\underline{K}$ and Q_0 approximates Q_p, which is itself spectrally equivalent to the Schur complement. By the result of Braess and Peisker we just need to show that this iteration converges, and we'll have a usable approximation to the discrete Stokes equations. We ignore the one zero eigenvalue of $\underline{K}$ which is due to the hydrostatic pressure here, and in what follows, since if we start an iteration orthogonal to this kernel, we will remain orthogonal to the kernel [39, Section 2.3].

Recall Corollary 4.1.2, that the convergence of this method depends upon the size of the parameter

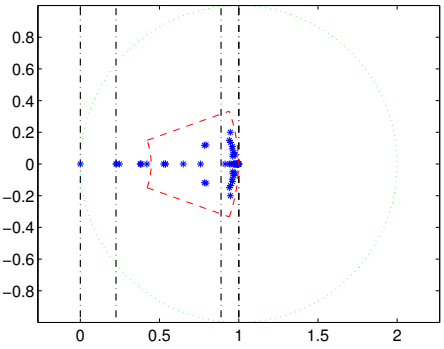
$$\begin{aligned}\xi := \max \Bigg\{ & 1-\upsilon, \Upsilon - 1, 1 - \frac{(1+\psi)\Upsilon - \sqrt{(1+\psi)^2\Upsilon^2 - 4\psi\Upsilon}}{2}, \\ & \frac{(1+\Psi)\Upsilon + \sqrt{(1+\Psi)^2\Upsilon^2 - 4\Psi\Upsilon}}{2} - 1, \sqrt{1+\Psi-2\sqrt{\Psi}\cos\theta}, \\ & \sqrt{1+\psi\upsilon - 2\sqrt{\psi\upsilon}\cos\theta} \Bigg\},\end{aligned}$$

where υ, Υ, ψ, Ψ, θ are constants defined in the statement of the Corollary. Inexact Uzawa will converge if $\xi < 1$, with the asymptotic convergence rate being ξ. Figure 7.6 shows the bounds predicted above and the actual eigenvalues for a number of approximations to the matrix $\underline{K}$.

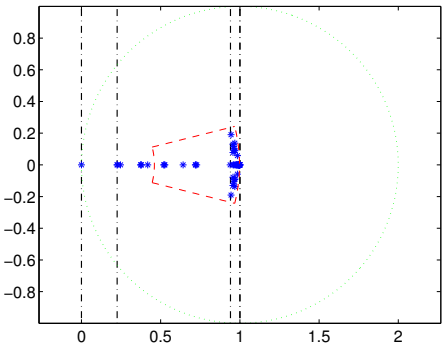
Figure 7.6 shows we will get asymptotic convergence, but in practice we see good results from the first iteration. Also, the theory above is equally valid for the block *upper* triangular approximation to the discrete Stokes matrix, whereas in practice we observe that it takes far more iterations with this upper triangular splitting to get good convergence. Recall from Section 3.1.1 that one way of quantifying any transient behaviour is by using pseudospectra. Figure 7.7 shows plots of the pseudospectra of $I - \mathcal{M}^{-1}\mathcal{K}$, generated by EigTool [129], in the cases where $\mathcal{M}$ is block upper triangular and block lower triangular. It is clear that the eigenvalue at unity (due to the hydrostatic pressure) has no effect on the pseudospectra in the block lower triangular case, but the pseudospectra in the block upper triangular case protrude significantly outside the unit circle. This will lead to some transient effects in the convergence of the simple iteration in the block upper triangular case [116, Chapter 24].



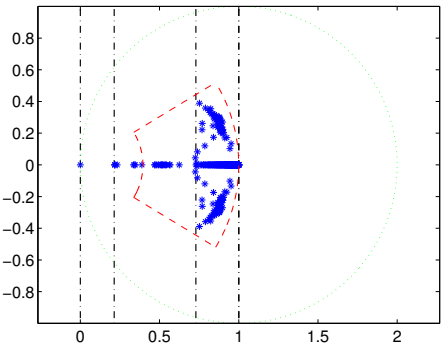
(a) h = 0.25, K_0 given by 1 AMG V-cycle with 1 pre- and 1 post-smoothing step



(b) h = 0.25, K_0 given by 1 AMG V-cycle with 2 pre- and 2 post-smoothing steps



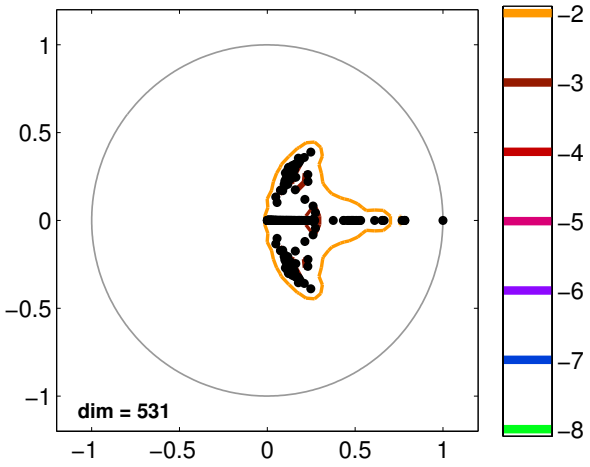
(c) h = 0.25, K_0 given by 2 AMG V-cycles with 2 pre- and 2 post-smoothing steps



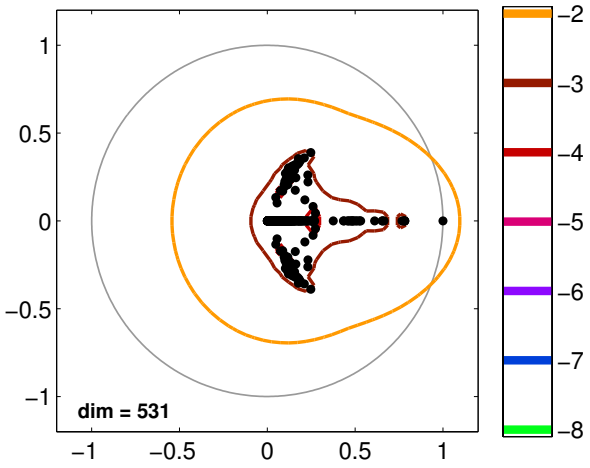
(d) h = 0.125, K_0 given by 1 AMG V-cycle with 1 pre- and 1 post-smoothing step

Figure 7.6: *'s denote computed eigenvalues. Lines, from left to right, are at 0, $\frac{(\psi+1)\Upsilon-\sqrt{(\psi+1)^2\Upsilon^2-4\psi\Upsilon}}{2}$, υ, Υ and $\frac{(\Psi+1)\Upsilon+\sqrt{(\Psi+1)^2\Upsilon^2-4\Psi\Upsilon}}{2}$. (the last two virtually coincide here). Dashed region is the bounds for the complex eigenvalues. Also shown is the unit circle centred at $z = 1$.

.



(a) Block-lower-triangular splitting, $S_0 = M_p$, K_0 is 1 AMG V-cycle, 1 pre- and 1 post-smoothing step



(b) Block-upper-triangular splitting, $S_0 = M_p$, K_0 is 1 AMG V-cycle, 1 pre- and 1 post-smoothing step

Figure 7.7: Pseudospectra of $I - \mathcal{M}^{-1}\mathcal{K}$

Let us again return to the case where $\underline{K} - \underline{K}_0$ is positive definite. Then we know from Section 4.3 that

$$\mathcal{M}^{-1}\mathcal{K} = \begin{bmatrix} \underline{K}_0 & 0 \\ B & -S_0 \end{bmatrix}^{-1} \begin{bmatrix} \underline{K} & B^T \\ B & 0 \end{bmatrix}$$

is self adjoint in the inner product defined by

$$\mathcal{H} = \begin{bmatrix} \underline{K} - \underline{K}_0 & 0 \\ 0 & S_0 \end{bmatrix}.$$

If we define $\widehat{\mathcal{K}} := \mathcal{M}^{-1}\mathcal{K}$, then we have that $\widehat{\mathcal{K}}$ is $\mathcal{H}-$normal, i.e.

$$\widehat{\mathcal{K}}^\dagger\widehat{\mathcal{K}} = \widehat{\mathcal{K}}\widehat{\mathcal{K}}^\dagger,$$

where $\widehat{\mathcal{K}}^\dagger = \mathcal{H}^{-1}\widehat{\mathcal{K}}^T\mathcal{H}$. This means that the iteration matrix $I - \mathcal{M}^{-1}\mathcal{K}$ is $\mathcal{H}-$normal, and so

$$\|I - \mathcal{M}^{-1}\mathcal{K}\|_{\mathcal{H}} = \rho(I - \mathcal{M}^{-1}\mathcal{K}).$$

This tells us that

$$\|\mathbf{x}_k - \mathbf{x}\|_{\mathcal{H}} \leq \rho^k \|\mathbf{x}\|_{\mathcal{H}},$$

where $\rho = \rho(I - \mathcal{P}^{-1}\mathcal{K})$, the spectral radius of the iteration matrix. To apply the result of Braess and Peisker we need a constant η_k such that the error converges the $2-$norm, i.e.

$$\|\mathbf{x}_k - \mathbf{x}\|_2 \leq \eta_k \|\mathbf{x}\|_2.$$

We know that over a finite dimensional vector space all norms are equivalent. Therefore there exists constants ψ and Ψ such that

$$\sqrt{\psi}\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_{\mathcal{H}} \leq \sqrt{\Psi}\|\mathbf{x}\|_2,$$

and hence

$$\begin{aligned} \|\mathbf{x}_k - \mathbf{x}\|_2 &\leq \|\mathbf{x}_k - \mathbf{x}\|_{\mathcal{H}}/\sqrt{\psi} \\ &\leq \frac{\rho^m}{\sqrt{\psi}}\|\mathbf{x}\|_{\mathcal{H}} \\ &\leq \frac{\sqrt{\Psi}\rho^m}{\sqrt{\psi}}\|\mathbf{x}\|_2. \end{aligned}$$

We now need to know the values of the constants ψ and Ψ.

Recall the standard bounds of the two dimensional finite element matrices from Theorems 2.1.1 and 2.1.2:

$$\begin{aligned} dh^2\mathbf{x}^T\mathbf{x} \leq \mathbf{x}^T\underline{K}\mathbf{x} \leq D\mathbf{x}^T\mathbf{x} \\ ch^2\mathbf{x}^T\mathbf{x} \leq \mathbf{x}^T Q\mathbf{x} \leq Ch^2\mathbf{x}^T\mathbf{x}. \end{aligned}$$

Also, we assume the approximation to the stiffness matrix satisfies

$$1 < \delta \leq \frac{\mathbf{x}^T \underline{K} \mathbf{x}}{\mathbf{x}^T \underline{K}_0 \mathbf{x}} \leq \Delta, \tag{7.20}$$

for some constants δ and Δ. Then $\mathbf{x}^T \mathcal{H} \mathbf{x} \leq \Psi \mathbf{x}^T \mathbf{x}$ means that

$$\mathbf{y}^T (\underline{K} - \underline{K}_0)\mathbf{y} + \mathbf{z}^T S_0 \mathbf{z} \leq \Psi(\mathbf{y}^T \mathbf{y} + \mathbf{z}^T \mathbf{z}).$$

Therefore if we have constants Ψ_1 and Ψ_2 such that

$$\mathbf{y}^T (\underline{K} - \underline{K}_0)\mathbf{y} \leq \Psi_1 \mathbf{y}^T \mathbf{y} \quad \text{and} \quad \mathbf{z}^T S_0 \mathbf{z} \leq \Psi_2 \mathbf{z}^T \mathbf{z}$$

then we can take $\Psi = \max(\Psi_1, \Psi_2)$.

First, note that from (7.20)

$$\begin{aligned}
\mathbf{x}^T \underline{K} \mathbf{x} &\leq \Delta \mathbf{x}^T \underline{K}_0 \mathbf{x} \\
\Delta \mathbf{x}^T \underline{K} \mathbf{x} - (\Delta - 1)\mathbf{x}^T \underline{K} \mathbf{x} &\leq \Delta \mathbf{x}^T \underline{K}_0 \mathbf{x} \\
\Delta(\mathbf{x}^T \underline{K} \mathbf{x} - \mathbf{x}^T \underline{K}_0 \mathbf{x}) &\leq (\Delta - 1)\mathbf{x}^T \underline{K} \mathbf{x} \\
\mathbf{x}^T (\underline{K} - \underline{K}_0)\mathbf{x} &\leq \frac{D(\Delta - 1)}{\Delta} \mathbf{x}^T \mathbf{x} \\
\therefore \ \frac{\mathbf{x}^T (\underline{K} - \underline{K}_0)\mathbf{x}}{\mathbf{x}^T \mathbf{x}} &\leq \frac{D(\Delta - 1)}{\Delta}.
\end{aligned}$$

Therefore

$$\Psi_1 = \frac{(\Delta - 1)D}{\Delta}.$$

Let $S_0 = \widetilde{Q_p}$, where

$$\theta \leq \frac{\mathbf{x}^T Q_p \mathbf{x}}{\mathbf{x}^T \widetilde{Q_p} \mathbf{x}} \leq \Theta$$

Then

$$\begin{aligned}
\frac{\mathbf{z}^T S_0 \mathbf{z}}{\mathbf{z}^T \mathbf{z}} &= \frac{\mathbf{z}^T \widetilde{Q_p} \mathbf{z}}{\mathbf{z}^T \mathbf{z}} \\
&= \frac{\mathbf{z}^T \widetilde{Q_p} \mathbf{z}}{\mathbf{z}^T Q_p \mathbf{z}} \cdot \frac{\mathbf{z}^T Q_p \mathbf{z}}{\mathbf{z}^T \mathbf{z}} \\
&\leq \frac{C_p h^2}{\theta}.
\end{aligned}$$

Therefore $\Psi_2 = C_p h^2$, and hence

$$\Psi = \max\left(\frac{(\Delta - 1)D}{\Delta}, \frac{C_p h^2}{\theta}\right).$$

Now we turn our attention to the lower bound. Similarly to above, we take $\psi = \min(\psi_1, \psi_2)$, where

$$\psi_1 \mathbf{y}^T\mathbf{y} \leq \mathbf{y}^T(\underline{K} - \underline{K}_0)\mathbf{y} \quad \text{and} \quad \psi_2 \mathbf{z}^T\mathbf{z} \leq \mathbf{z}^T S_0 \mathbf{z}.$$

Again, we have from (7.20):

$$\begin{aligned}
\delta \mathbf{y}^T \underline{K}_0 \mathbf{y} &\leq \mathbf{y}^T \underline{K} \mathbf{y} \\
&= \delta \mathbf{y}^T \underline{K} \mathbf{y} + (1-\delta)\mathbf{y}^T \underline{K} \mathbf{y} \\
(\delta - 1)\mathbf{y}^T \underline{K} \mathbf{y} &\leq \delta \mathbf{y}^T (\underline{K} - \underline{K}_0)\mathbf{y} \\
\frac{(\delta-1)dh^2}{\delta} &\leq \frac{\mathbf{y}^T(\underline{K} - \underline{K}_0)\mathbf{y}}{\mathbf{y}^T\mathbf{y}}.
\end{aligned}$$

Again following an argument as above,

$$\begin{aligned}
\frac{\mathbf{z}^T S_0 \mathbf{z}}{\mathbf{z}^T\mathbf{z}} &= \frac{\mathbf{z}^T \widetilde{Q_p}\mathbf{z}}{\mathbf{z}^T Q_p \mathbf{z}} \cdot \frac{\mathbf{x}^T Q_p \mathbf{x}}{\mathbf{x}^T\mathbf{x}} \\
&\geq \frac{c_p h^2}{\Theta}.
\end{aligned}$$

Therefore

$$\psi = \min\left(\frac{(\delta-1)c_{\vec{u}} dh^2}{\delta}, \frac{c_p h^2}{\Theta} \right).$$

The contraction constant for convergence in the 2-norm is given by $\rho^m \sqrt{\Psi}/\sqrt{\psi}$. It is clear that $\sqrt{\psi} = \beta h$, where β is a constant. For the numerator, in general, we will have $\Psi = \frac{(\Delta-1)D}{\Delta}$, as h^2 is small. This would mean that

$$\frac{\sqrt{\Psi}}{\sqrt{\psi}} = \mathcal{O}\left(h^{-1}\right),$$

i.e. the contraction constant would be dependent upon h.

However, we have control over the value of Δ, as this measures the accuracy of the approximation to $\underline{K}$. Recall that $\underline{K}_0$ is a good approximation to $\underline{K}$ if Δ is close to unity. As $\underline{K}_0$ is a multigrid process we can make this parameter as close to 1 as required by simply taking more V-cycles, better smoothing, etc. If this approximation is good enough, and $\frac{(\Delta-1)D}{\Delta}$ is smaller than $\frac{C_p h^2}{\theta}$, we will get a constant number of iterations, at least up to some value of h. Note that we have knowledge of all the parameters involved, so given a smallest required value of h – which one will know a priori – one can pick an approximation K_0 which gives a reasonable method. The

quantity ρ^m also appears in the numerator, so convergence can be improved by taking more inexact Uzawa iterations.

Table 7.3 shows the results for solving (7.19) using MINRES, with right hand side as in Example 7.3.1 and with $\beta = 10^{-2}$. As a preconditioner we use the block lower triangular preconditioner, with $\mathcal{K}$ approximated by m steps of the simple iteration with splitting matrix

$$\mathcal{M} = \begin{bmatrix} \underline{K}_0 & 0 \\ B & -S \end{bmatrix},$$

where $S = B\underline{K}^{-1}B^T$ is the exact Schur complement of the Stokes equation. $\underline{K}_0$ is given by k AMG V-cycles. This is not a practical preconditioner, but we can see clearly that if the approximation $\underline{K}_0$ is not good enough we don't get an optimal preconditioner. This phenomenon is explained by the theory above.

Table 7.3: Comparison of solution methods for solving Example 7.3.1 using MINRES preconditioned with the block diagonal preconditioner with m steps of inexact Uzawa approximating $\mathcal{K}$ and k AMG V-cycles approximating $\underline{K}$.

h	size	Exact, m=1		m=1, k=1		m=1, k=2		m=1, k=3		m=1, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.089	25	0.092	29	0.079	27	0.082	27	0.085	27
2^{-3}	1512	0.382	27	0.432	35	0.352	27	0.365	27	0.380	27
2^{-4}	6344	3.192	25	7.359	65	3.179	27	3.235	27	3.296	27
2^{-5}	25992	60.063	25	403.933	179	72.858	31	64.028	27	64.055	27
h	size	Exact, m=2		m=2, k=1		m=2, k=2		m=2, k=3		m=2, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.073	21	0.100	27	0.099	25	0.096	23	0.101	23
2^{-3}	1512	0.408	23	0.429	29	0.400	25	0.423	25	0.450	25
2^{-4}	6344	3.466	23	3.954	31	3.347	25	3.193	23	3.319	23
2^{-5}	25992	57.284	21	98.885	39	65.489	25	60.051	23	61.398	23

Even though the above argument only holds for $\underline{K} - \underline{K}_0$ positive definite, we see the same behaviour in practice for the general case. Because solving the approximation to $\mathcal{K}$ is particularly expensive here it is worth getting the approximation to the mass matrix, Q_0, as close to Q as possible. Therefore, in the results that follow, we take Q_0 to be given by 20 steps of the Chebyshev semi-iteration applied to the appropriate mass matrix. As discussed in Section 4.1, the inexact Uzawa method can be improved with the introduction of a parameter τ in front of the approximation to the Schur complement. In the inexact case we saw that the optimal parameter is hard to obtain, but a good approximation is $(\phi + \Phi)/2$, where $\lambda(S_0^{-1}S) \in [\phi, \Phi]$. We saw in Section 7.2 that for $\mathbf{Q}_1$ elements and a Dirichlet problem, $\lambda(Q_p^{-1}S) \in [0.2, 1]$, so we take our

scaling parameter as $\tau = 3/5$. We therefore advocate a practical splitting matrix for inexact Uzawa of

$$\mathcal{M} = \begin{bmatrix} \underline{K}_0 & 0 \\ B & -\tau Q_0 \end{bmatrix}.$$

Having defined approximations to the Schur complement and the (1,1) block of the saddle point system we can now look at practical ways of solving the system (7.19). As the block $\hat{\mathcal{Q}}$ is not square, we cannot use the constraint preconditioning technique derived in Section 5.6 here. However, the other two methods – a block diagonal preconditioner for MINRES and a block lower triangular preconditioner for CG in a non-standard inner product – can be used here.

Tables A.5 and A.6 in Appendix A show some experiments with different numbers of V-cycles and iterations of inexact Uzawa in the approximations for $\mathcal{K}$. These suggest that taking two steps of the inexact Uzawa method, with $\underline{K}_0$ given by three AMG V-cycles, will be a good preconditioner. The results for solving Example 7.3.1 with $\beta = 10^{-2}$, $\delta = 1$ to a tolerance of 10^{-6} (in the appropriate norm) are given in Table 7.4.

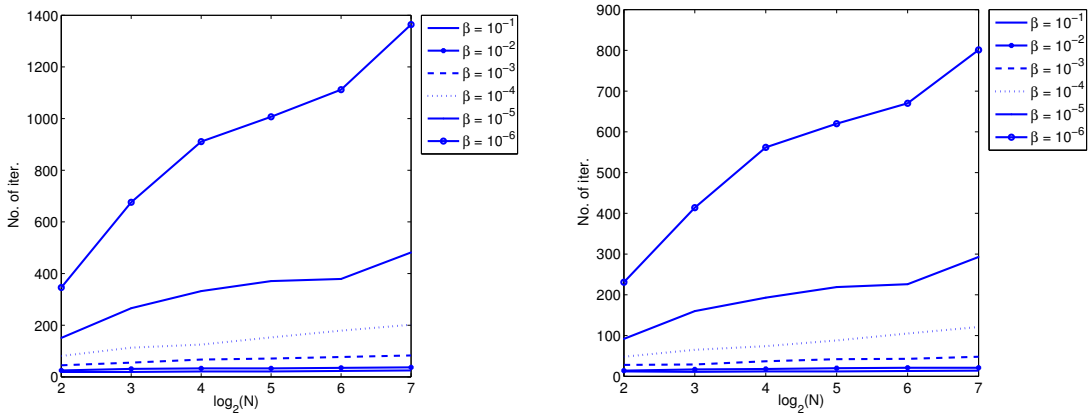
Table 7.4: Comparison of solution methods for solving Example 7.3.1 using MINRES and BPCG preconditioned with the block diagonal and block lower triangular preconditioners respectively with 2 steps of inexact Uzawa approximating $\mathcal{K}$ and 3 AMG V-cycles approximating $\underline{K}$.

h	size	MINRES		BPCG		`backslash`
		time	its	time	its	time
2^{-2}	344	0.189	25	0.083	14	0.016
2^{-3}	1512	0.358	31	0.194	17	0.059
2^{-4}	6344	1.176	33	0.679	18	0.601
2^{-5}	25992	4.965	33	3.133	20	7.300
2^{-6}	105224	22.704	35	14.584	21	—

As we see from Table 7.4, the overall technique which we have described seems to be a good method for solving the Stokes control problem. Comparing the results here with those to solve the forward problem in Table 7.1 the iteration numbers aren't that much more, and solving the control problem using the block-triangular preconditioner is just over a factor of ten more expensive that solving a single forward problem for every grid size.

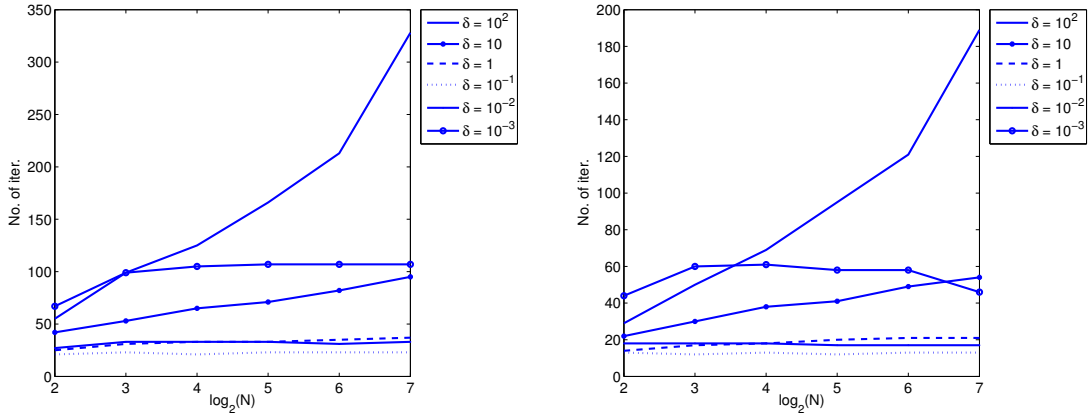
Figures 7.8 and 7.9 show the number of iterations taken to solve this problem for different values of β and δ respectively. These show that – as we might expect from the theory – lowering β and increasing δ make the number of iterations required to solve the system higher using our methods. From the plots in Figures 7.3 and 7.4

it seems that the value $\delta = 1$ gives a pressure of the same order as the uncontrolled problem, the solution of which is shown in Figure 7.1. However, one can conceive of situations where we require a tighter bound on the pressure, and hence a higher value of δ.



(a) MINRES (b) BPCG

Figure 7.8: Plot of problem size vs iterations needed for different β, where $\delta = 1$.



(a) MINRES (b) BPCG

Figure 7.9: Plot of problem size vs iterations needed for different δ, where $\beta = 10^{-2}$.

7.5 Comments

In this chapter we applied a block diagonal preconditioner with MINRES and a block lower triangular preconditioner with CG in a non-standard inner product to a flow control problem. We based our approximation to the (1,1) block on the same approximation to the mass matrix introduced previously.

We had to be a bit more careful when approximating the PDE here – it is not enough to simply use a preconditioner for the forward problem. Our approximation to the Stokes equation is based on the inexact Uzawa method. We use the fact that the pressure mass matrix is spectrally equivalent to the Schur complement of the discrete Stokes equations, which we get from the LBB condition. We showed that, for a given h, we can pick an approximation to the stiffness matrix such that we get an efficient method for that value of h; this approximation will also be effective for all larger h. The CPU times taken to solve the control problem are about an order of magnitude larger than those to solve the forward problem on the same size mesh using block-diagonally preconditioned MINRES, which seems an acceptable cost for a much harder problem.

Chapter 8

Conclusion

The discrete optimality system for PDE-constrained optimization problems has a saddle point structure, and we typically know a lot of information about the individual blocks. This thesis describes how to use this information to develop block preconditioners to speed up convergence of Krylov subspace methods when applied to such problems.

In the early chapters we saw how iterative methods can be split into linear methods – which could be written in the form of a matrix – and non-linear methods. The non-linear methods, such as CG and MINRES, automatically choose the optimal parameters that the method needs, which often leads to superlinear convergence. Therefore these should be the iterative methods of choice.

We looked at three iterative methods which can be applied to these problems – MINRES, conjugate gradients with a non-standard inner product, and projected conjugate gradients. These are preconditioned by different types of block matrices – namely block diagonal, block lower triangular and constraint preconditioners respectively. Ideal preconditioners in each of these cases generally involve the (1,1) block and the Schur complement of the saddle point system. We introduced an approximation to the Schur complement that involves dropping one of the additive terms, and this seems to work well for the range of regularization parameters we're interested in.

Our aim in this thesis was to develop preconditioners that are factorization free, and for which the time to solve the system scales linearly with the problem size (at least until the computer runs out of memory). Therefore we needed to approximate the blocks in the ideal preconditioner by spectrally equivalent – but cheaper to solve – matrices.

A good method to approximate a block is often a few steps of an (inner) iterative process. Here we must use a linear method unless we use a flexible outer iteration.

In this thesis we saw two cases – the Chebyshev semi-iteration for the mass matrix and inexact Uzawa for the Stokes equations – where this philosophy was used.

The (1,1) block is invariably a block diagonal matrix composed of mass matrices or scalar multiples of mass matrices. The Chebyshev semi-iteration is a relatively inexpensive algorithm that enables you to get as good an approximation to this block as we need, so this is an integral part of our approach. We also saw how this approximation can be used to improve current preconditioners for the Stokes equations.

For the Schur complement we require an approximation to the PDE for which its square approximates the square of the PDE – a preconditioner for the forward problem isn't always enough. This property holds whenever we use a convergent linear iteration as our approximation – e.g. multigrid or inexact Uzawa.

We demonstrated our methods with Poisson's equation, the convection-diffusion and the Stokes equations, and also for a number of different boundary conditions. They seemed to perform well for a range of values of the regularization parameter, but were not independent of this parameter. The methods – in particular the approximation used for the Schur complement – break down for 'small' β.

Systems of a similar form result when we include bound constraints on the control via a primal-dual active set method, and it has been demonstrated elsewhere that these ideas can be used effectively there. Similarly, they should be able to handle state constraints when incorporated by Moreau-Yosida relaxation.

There are many more topics that could be covered. The cases included here are all idealized cases, and a practical situation would be a lot more difficult. Further complications could include multiple PDE constraints, the control only acting on a part of the domain, different regularization methods and minimizing different norms, to name but a few. It appears that the methods described here, possibly with some modification, may be applicable in at least some of these cases, and exploring these would be an interesting area for future research.

Appendix A

Further Numerical Results

All results were obtained using MATLAB 7.9.0 (R2009b) on a PC with a triple core 2800MHz AMD Phenom II X3 720 Processor with 3.3GB of RAM, running Ubuntu Linux 8.04.

A.1 Poisson control with MINRES

Below we consider Example 5.1.1, and use a GMG approximation for K. Table A.1 shows the results for $\beta = 0.01$, and we vary the number of V-cycles and the pre- and post-smoothing steps. The mass matrix is approximated in the preconditioner by five steps of the Chebyshev semi-iteration. The best choice in terms of CPU time here appears to be taking the GMG approximation to be one V-cycle, and three pre-smoothing steps on the first K solve, and corresponding post-smoothing steps on the second K solve, so that $\widetilde{K}M^{-1}\widetilde{K}^T$ is symmetric.

Now consider the same problem – where we take $\widetilde{K}$ as described above, but where we vary the number of steps of the Chebyshev semi-iteration. Table A.2 shows these results. Here we see that taking five steps of the Chebyshev semi-iteration seems a reasonable value.

The results above also show that the method is fairly robust with respect to these parameters; we have not tried to find the optimal set of parameters for all situations, but have found some are good in this particular situation, and hence we infer that they'll be reasonable – if not the best – in the other examples.

Table A.3 shows the results of varying the approximation to the mass matrix in 3D. It seems we need more steps of the Chebyshev semi-iteration here to get good convergence. It also seems that we need a better approximation to the stiffness matrix to get an optimal preconditioner.

Table A.1: CPU times and iterations needed to solve (3.32) for Example 5.1.1 using MINRES preconditioned with a block diagonal preconditioner with 5 steps of the Chebyshev semi-iteration for the mass matrix, and a GMG method or a direct method for the stiffness matrix.

N	`backslash`		1V(1,1)		1V(2,0)		1V(2,2)		1V(3,0)		1V(4,0)	
	time	its	time	its	time	its	time	its	time	its	time	its
2^{-2}	0.0048	11	0.2843	13	0.2839	13	0.2857	11	0.2851	11	0.2824	11
2^{-3}	0.0087	12	0.2929	17	0.2907	16	0.2901	14	0.2901	14	0.2905	14
2^{-4}	0.0303	15	0.3183	22	0.3092	18	0.3085	16	0.3120	16	0.3098	16
2^{-5}	0.1394	16	0.4204	26	0.3856	20	0.3841	18	0.3804	18	0.3778	17
2^{-6}	0.6749	15	0.8269	26	0.6931	20	0.7324	18	0.6771	18	0.6526	17
2^{-7}	3.3118	15	2.6753	27	2.0937	20	1.9647	18	1.9306	18	1.8247	16
2^{-8}	17.3016	14	10.7074	27	7.6996	19	7.4689	17	7.2917	17	7.1032	16
2^{-9}	89.3951	13	43.1284	26	31.4706	19	30.4888	17	25.9341	15	26.7674	15
			2V(1,1)		2V(2,0)		2V(2,2)		2V(3,0)		2V(4,0)	
			time	its	time	its	time	its	time	its	time	its
2^{-2}			0.2875	11	0.2859	11	0.2866	11	0.2884	11	0.2875	11
2^{-3}			0.2951	12	0.2949	12	0.2976	12	0.2968	12	0.2963	12
2^{-4}			0.3248	16	0.3237	15	0.3274	15	0.3251	15	0.3278	15
2^{-5}			0.4337	17	0.4323	17	0.4375	16	0.4436	17	0.4391	16
2^{-6}			0.8532	17	0.8919	17	0.8480	15	0.8133	15	0.8370	15
2^{-7}			2.6505	16	2.6094	16	2.5868	15	2.5599	15	2.6469	15
2^{-8}			10.6491	16	10.6604	16	10.1085	14	9.5502	14	9.9863	14
2^{-9}			40.6688	15	40.2644	15	38.4384	13	37.2299	13	38.6217	13

Table A.2: CPU times and iterations needed to solve (3.32) for Example 5.1.1 in two dimensions using MINRES preconditioned with a block diagonal preconditioner with one V-cycle of a GMG method with 3 pre-smoothing steps for stiffness matrix, and the Chebyshev semi-iteration or a direct method for the mass matrix.

N	`backslash`		Cheb(1)		Cheb(2)		Cheb(3)		Cheb(4)		Cheb (5)	
	time	its	time	its	time	its	time	its	time	its	time	its
2^{-2}	0.2829	9	0.2897	18	0.2888	15	0.2854	13	0.2851	12	0.2833	11
2^{-3}	0.2927	11	0.3080	32	0.2995	25	0.2942	18	0.2904	15	0.2897	14
2^{-4}	0.3135	13	0.3293	30	0.3569	45	0.3125	20	0.3163	20	0.3069	16
2^{-5}	0.4194	13	0.4061	27	0.6248	70	0.3831	21	0.3982	23	0.3747	18
2^{-6}	0.9701	13	0.7728	26	1.7862	79	0.6730	20	0.7384	23	0.6535	18
2^{-7}	3.6334	13	2.1928	24	6.6955	80	1.9909	20	2.1458	22	1.8589	18
2^{-8}	24.3166	14	8.4898	23	27.9162	77	7.5390	19	8.8784	22	7.3544	17
2^{-9}	712.7708	15	34.3914	23	107.9846	72	29.3921	18	36.7658	22	25.8337	15

			Cheb (6)		Cheb (7)		Cheb (8)		Cheb (9)		Cheb (10)	
			time	its	time	its	time	its	time	its	time	its
2^{-2}			0.2856	10	0.2856	10	0.2847	10	0.2843	9	0.2857	10
2^{-3}			0.2910	13	0.2905	13	0.2907	13	0.2891	12	0.2902	12
2^{-4}			0.3056	15	0.3078	15	0.3079	15	0.3061	14	0.3075	14
2^{-5}			0.3761	18	0.3640	15	0.3671	15	0.3688	15	0.3661	14
2^{-6}			0.6617	18	0.6515	17	0.6392	16	0.6529	16	0.6126	14
2^{-7}			1.8069	17	1.8478	17	1.8010	16	1.8221	16	1.8670	16
2^{-8}			7.4505	17	7.1906	16	7.3044	16	7.5308	16	7.7240	16
2^{-9}			29.8898	17	29.4687	16	29.8581	16	28.6421	15	31.2615	16

Table A.3: CPU times and iterations needed to solve (3.32) for Example 5.1.1 in three dimensions using MINRES preconditioned with a block diagonal preconditioner with one V-cycle of a GMG method with 3 pre-smoothing steps for stiffness matrix, and the Chebyshev semi-iteration or a direct method for the mass matrix.

N	`backslash`		Cheb(5)		Cheb(6)		Cheb(7)		Cheb(8)		Cheb (9)	
	time	its	time	its	time	its	time	its	time	its	time	its
2^{-2}	0.0900	9	0.0936	14	0.0908	12	0.0913	12	0.0905	11	0.0909	11
2^{-3}	0.1752	11	0.1393	18	0.1370	17	0.1347	16	0.1361	16	0.1274	13
2^{-4}	2.2062	13	0.7744	22	0.7227	20	0.7267	19	0.6939	18	0.6758	17
2^{-5}	80.8994	15	8.3222	24	10.6165	31	7.8761	22	7.9963	22	7.4102	20

			Cheb (10)		Cheb (11)		Cheb (12)		Cheb (13)		Cheb (14)	
			time	its	time	its	time	its	time	its	time	its
2^{-2}			0.0910	11	0.0905	10	0.0917	10	0.0911	10	0.0914	10
2^{-3}			0.1284	13	0.1298	13	0.1313	13	0.1322	13	0.1300	12
2^{-4}			0.6193	15	0.6321	15	0.6444	15	0.6553	15	0.6684	15
2^{-5}			7.0364	18	7.0979	18	6.9172	17	6.9669	17	7.0282	17

A.2 Poisson control with BPCG

Below we again consider Example 5.1.1 with use a GMG approximation for K. Table A.4 shows the results for $\beta = 0.01$, and we vary the scaling parameter. The mass matrix is approximated in the preconditioner by five steps of the Chebyshev semi-iteration and the GMG approximation is taken to be one V-cycle, with three pre-smoothing steps on the first K solve, and corresponding post-smoothing steps on the second K solve, so that $\widetilde{K}M^{-1}\widetilde{K}^T$ is symmetric. This shows that the number of iterations taken is fairly independent of the choice of parameter, even when $A - A_0$ is indefinite. We take a value of 0.9, as this is theoretically in the right range, and seems to be a good parameter in terms of mesh-independence.

Table A.4: CPU times and iterations needed to solve (3.32) for Example 5.1.1 using BPCG preconditioned with a block diagonal preconditioner with one V-cycle of a GMG method with 3 pre-smoothing steps for stiffness matrix, and the 5 steps of the Chebyshev semi-iteration scaled by the parameter given below.

N	`backslash`		0.8		0.82		0.84		0.86		0.88	
	time	its	time	its	time	its	time	its	time	its	time	its
2^{-2}	0.0066	11	0.2892	12	0.2865	12	0.2870	12	0.2886	11	0.2872	11
2^{-3}	0.0104	11	0.2939	12	0.2930	12	0.2927	12	0.2913	11	0.2928	11
2^{-4}	0.0238	10	0.3068	12	0.3063	12	0.3061	12	0.3064	12	0.3047	11
2^{-5}	0.0956	10	0.3670	13	0.3591	12	0.3601	12	0.3608	12	0.3600	12
2^{-6}	0.5555	10	0.5990	12	0.5997	12	0.5907	12	0.5965	12	0.5990	12
2^{-7}	2.3584	10	1.6910	12	1.6943	12	1.7079	12	1.7417	12	1.6059	11
2^{-8}	12.8185	10	6.6888	12	6.6080	12	6.6104	12	6.5941	12	6.1898	11
2^{-9}	69.9849	10	26.7286	12	26.6583	12	26.8807	12	26.6778	12	24.4598	11
			0.9		0.92		0.94		0.96		0.98	
			time	its	time	its	time	its	time	its	time	its
2^{-2}			0.2864	11	0.2863	11	0.2864	11	0.2818	10	0.2868	12
2^{-3}			0.2910	11	0.2915	11	0.2913	11	0.2917	11	0.2931	12
2^{-4}			0.3038	11	0.3037	11	0.3045	11	0.3073	12	0.3045	11
2^{-5}			0.3525	11	0.3532	11	0.3528	11	0.3525	11	0.3530	11
2^{-6}			0.5674	11	0.5662	11	0.5673	11	0.5718	11	0.5928	12
2^{-7}			1.6020	11	1.5714	11	1.5719	11	1.5730	11	1.5718	11
2^{-8}			6.0851	11	6.1732	11	6.2072	11	6.1557	11	6.2095	11
2^{-9}			24.2881	11	24.6174	11	24.8731	11	24.3915	11	24.8613	11

A.3 Stokes Control

Tables A.5 and A.6 show the CPU times and number of iterations taken to solve Example 7.3.1, with $\delta = 1$, $\beta = 0.01$, and using a $\mathbf{Q}_2 - \mathbf{Q}_1$ discretization. We use for various combinations of steps of inexact Uzawa (m) and numbers of V-cycles (k). In both cases, $m = 2,\ k = 3$ is close to the optimal in terms of CPU times, so this is the approximation we will use in our calculations.

Table A.5: Comparison of solution methods for solving Example 7.3.1 using MINRES preconditioned with the block diagonal preconditioner with m steps of inexact Uzawa approximating $\mathcal{K}$ and k AMG V-cycles approximating $\underline{K}$.

h	size	Exact, m=1		m=1, k=1		m=1, k=2		m=1, k=3		m=1, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.159	33	0.159	39	0.147	37	0.150	37	0.146	35
2^{-3}	1512	0.340	41	0.350	51	0.348	45	0.342	43	0.362	43
2^{-4}	6344	1.373	45	1.498	77	1.075	49	1.189	49	1.300	49
2^{-5}	25992	6.925	49	15.337	202	4.629	53	5.160	53	5.501	51
2^{-6}	105224	33.651	51	256.084	761	22.715	59	23.590	55	25.946	55
2^{-7}	423432	166.125	55	4324.460	2917	156.118	91	118.288	61	123.625	57
h	size	Exact, m=2		m=2, k=1		m=2, k=2		m=2, k=3		m=2, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.127	25	0.155	29	0.152	27	0.145	25	0.150	25
2^{-3}	1512	0.358	29	0.351	37	0.348	33	0.360	31	0.392	31
2^{-4}	6344	1.393	29	1.325	51	1.085	35	1.173	33	1.244	31
2^{-5}	25992	6.692	29	9.135	91	4.547	37	4.690	33	5.049	31
2^{-6}	105224	32.018	29	112.836	251	24.607	45	22.377	35	22.496	31
2^{-7}	423432	149.018	29	1665.296	831	168.235	69	107.146	37	110.390	33
h	size	Exact, m=3		m=3, k=1		m=3, k=2		m=3, k=3		m=3, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.160	25	0.197	29	0.193	27	0.194	26	0.195	25
2^{-3}	1512	0.441	27	0.353	29	0.395	29	0.442	29	0.454	27
2^{-4}	6344	1.767	27	1.085	33	1.161	29	1.359	29	1.562	29
2^{-5}	25992	8.549	27	5.335	43	4.831	31	5.787	31	6.737	31
2^{-6}	105224	45.490	29	42.679	75	24.762	35	26.070	31	30.053	31
2^{-7}	423432	207.394	29	486.900	193	126.073	39	127.852	33	139.232	31
h	size	Exact, m=4		m=4, k=1		m=4, k=2		m=4, k=3		m=4, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.178	23	0.223	27	0.201	23	0.212	23	0.221	23
2^{-3}	1512	0.510	25	0.425	29	0.452	27	0.473	25	0.524	25
2^{-4}	6344	2.076	25	1.226	31	1.420	29	1.574	27	1.825	27
2^{-5}	25992	10.568	25	5.337	35	5.616	29	6.818	29	7.506	27
2^{-6}	105224	50.025	25	32.377	47	26.969	31	30.326	29	32.991	27
2^{-7}	423432	231.266	25	274.496	89	138.780	35	140.510	29	154.589	27

Table A.6: Comparison of solution methods for solving Example 7.3.1 using BPCG preconditioned with the block lower triangular preconditioner with m steps of inexact Uzawa approximating $\mathcal{K}$ and k AMG V-cycles approximating $\underline{K}$.

h	size	Exact, k=1		m=1, k=1		m=1, k=2		m=1, k=3		m=1, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.094	18	0.090	21	0.077	20	0.080	20	0.081	20
2^{-3}	1512	0.194	23	0.213	30	0.185	24	0.196	24	0.207	24
2^{-4}	6344	0.805	25	0.975	46	0.638	27	0.697	27	0.759	27
2^{-5}	25992	4.135	28	9.876	116	2.881	30	2.857	27	3.366	29
2^{-6}	105224	20.453	28	195.805	488	15.557	35	15.289	31	16.075	30
2^{-7}	423432	88.670	27	3335.464	1910	97.563	50	71.348	33	75.632	32
h	size	Exact, k=2		m=2, k=1		m=2, k=2		m=2, k=3		m=2, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.069	14	0.087	17	0.080	15	0.077	14	0.080	14
2^{-3}	1512	0.208	17	0.211	22	0.203	19	0.197	17	0.213	17
2^{-4}	6344	0.681	14	0.806	29	0.680	21	0.667	18	0.623	15
2^{-5}	25992	3.960	17	5.905	54	2.853	22	3.002	20	2.900	17
2^{-6}	105224	20.285	16	78.473	152	16.427	27	14.323	21	13.112	17
2^{-7}	423432	84.896	16	1180.818	522	109.179	41	64.640	21	62.739	18
h	size	Exact, k=3		m=3, k=1		m=3, k=2		m=3, k=3		m=3, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.085	14	0.104	16	0.103	15	0.113	16	0.111	15
2^{-3}	1512	0.248	15	0.204	17	0.216	16	0.238	16	0.263	16
2^{-4}	6344	0.933	14	0.656	19	0.699	17	0.768	16	0.876	16
2^{-5}	25992	5.126	16	3.347	25	3.011	18	3.333	17	3.815	17
2^{-6}	105224	25.932	16	28.122	44	15.390	20	15.210	17	17.418	17
2^{-7}	423432	116.586	16	314.582	114	77.655	23	71.785	18	82.790	18
h	size	Exact, k=4		m=4, k=1		m=4, k=2		m=4, k=3		m=4, k=4	
		time	its	time	its	time	its	time	its	time	its
2^{-2}	344	0.095	13	0.118	15	0.108	13	0.113	13	0.118	13
2^{-3}	1512	0.257	13	0.243	17	0.243	15	0.254	14	0.285	14
2^{-4}	6344	1.159	14	0.734	18	0.801	16	0.768	13	0.885	13
2^{-5}	25992	6.079	14	3.271	21	3.355	17	3.810	16	4.147	15
2^{-6}	105224	29.596	14	21.686	29	15.791	17	17.472	16	16.522	13
2^{-7}	423432	128.962	14	173.559	53	86.672	21	78.532	16	85.994	15

Bibliography

[1] Robert Alexander Adams and John J. F. Fournier. *Sobolev spaces*. Academic Press, 2nd edition, 2003.

[2] S.R. Arridge. Optimal tomography in medical imaging. *Inverse Problems*, 15:R41–R93, 1999.

[3] K Arrow, L. Hurwicz, and H Uzawa. *Studies in Nonlinear Programming*. Stanford University Press, 1958.

[4] S.F. Ashby. CHEBYCODE: A Fortran implementation of Manteuffel's adaptive Chebyshev algorithm. Technical report, University of Illinois, 1985.

[5] S.F. Ashby, T.A. Manteuffel, and J.S. Otto. A comparison of adaptive Chebyshev and least squares polynomial preconditioning for Hermitian positive definite linear systems. *SIAM J. Sci. Comput.*, 13:1–29, 1992.

[6] Uri M. Asher and Eldad Haber. A multigrid method for distributed parameter estimation problems. *Electron. Trans. Numer. Anal.*, 15:1–17, 2003.

[7] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 2nd edition, 1994.

[8] Roland Becker and Boris Vexler. Optimal control of the convection-diffusion equation using stabilized finite element methods. *Numer. Math.*, 106(3):349–367, 2007.

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

[10] Maïtine Bergounioux, Kazufumi Ito, and Karl Kunisch. Primal-dual strategy for constrained optimal control problems. *SIAM J. Control Optim.*, 37(4):1176–1194, 1999.

[11] G. Biros and O. Ghattas. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: The Krylov-Schur solver. *SIAM J. Sci. Comput.*, 27, 2000.

[12] George Biros and Günay Dogan. A multilevel algorithm for inverse problems with elliptic PDE constraints. *Inverse Problems*, 24(3):034010 (18pp), 2008.

[13] A. Borzì and V. Schulz. Multigrid methods for PDE optimization. *SIAM Rev.*, 51(2):361–395, 2009.

[14] Ilia Bouchouev and Victor Isakov. Uniqueness, stability and numerical methods for the inverse problem that arises in financial markets. *Inverse Problems*, 15:R95–R116, 1999.

[15] J. Boyle, M. D. Mihajlovic, and J. A. Scott. HSL_MI20: an efficient amg preconditioner. Technical Report RAL-TR-2007-021, Department of Computational and Applied Mathematics, Rutherford Appleton Laboratory, 2007.

[16] M. Braack. Optimal control in fluid mechanics by finite elements with symmetric stabilization. *SIAM J. Control Optim.*, 48(2):672–687, 2009.

[17] D. Braess and W. Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *SIAM J. Numer. Anal.*, 20(5):967–975, 1983.

[18] D. Braess and D. Peisker. On the numerical solution of the biharmonic equation and the role of squaring matrices for preconditioning. *IMA J Numer. Anal.*, 6:393–404, 1986.

[19] Dietrich Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 3rd edition, 2007.

[20] J.H. Bramble and J. E. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Math. Comp.*, 50:1–17, 1988.

[21] J.H. Bramble, J. E. Pasciak, and A.T. Vassilev. Analysis of the inexact Uzawa algorithm for saddle point problems. *SIAM J. Numer. Anal.*, 34:1072–1092, 1997.

[22] A Brandt. Multi-level adaptive technique MLAT for fast numerical solution to boundary value problems. In *3rd International Conference on Numerical Methods in Fluid Mechanics*, 1973.

[23] A Brandt, S.C. McCormick, and J. Ruge. *Algebraic multigrid (AMG) for sparse matrix equations*, pages 257–284. Cambridge University Press, 1985.

[24] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer, 3rd edition, 2008.

[25] Robert Bridson and Chen Greif. A multipreconditioned conjugate gradient algorithm. *SIAM J. Matrix Anal. Appl.*, 27(4):1056–1068, 2006.

[26] William L. Briggs, Van Emden Henson, and S.F. McCormick. *A Multigrid Tutorial*. SIAM, 2nd edition, 2000.

[27] Zhi-Hao Cao. A note on eigenvalues of matrices which are self-adjoint in symmetric bilinear forms. *SIAM J. Matrix Anal. Appl.*, 30:1421–1423, 2008.

[28] Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2nd revised edition, 2002.

[29] S. S. Collis and M. Heinkenschloss. Analysis of the streamline upwind/Petrov Galerkin method applied to the solution of optimal control problems. Technical Report TR02–01, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005–1892, 2002.

[30] P. Concus, G.H. Golub, and D.P. O'Leary. A generalized conjugate gradient method for the numerical solution of elliptic partial-differential equations. In *Proceedings of the Symposium on Sparse Matrix Computations, Argonne National Laboratory, Academic, New York*, pages 309–332, 1976.

[31] H. S. Dollar. *Iterative linear algebra for constrained optimization*. PhD thesis, University of Oxford, 2005.

[32] H. S. Dollar, Nicholas I. M. Gould, Martin Stoll, and Andrew J. Wathen. Preconditioning saddle-point systems with applications in optimization. *SIAM J. Sci. Comput.*, 32(1):249–270, 2010.

[33] H. Sue Dollar, Nicholas I. M. Gould, Wil H. A. Schilders, and Andrew J. Wathen. Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems. *SIAM J. Matrix Anal. Appl.*, 28(1):170–189, 2006.

[34] H. Sue Dollar and Andrew J. Wathen. Approximate factorization constraint preconditioners for saddle-point matrices. *SIAM J. Sci. Comput.*, 27(5):1555–1572, 2006.

[35] I Duff, A.M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, 1986.

[36] H. Egger and H. W. Engl. Tikhonov regularization applied to the inverse problem of option pricing: Convergence analysis and rates. *Inverse Problems*, 21:1027–1045, 2005.

[37] H.C. Elman. Multigrid and Krylov subspace methods for the discrete Stokes equations. *Internat. J. Numer. Methods Fluids*, 22:755–770, 1995.

[38] H.C. Elman and G.H. Golub. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM J. Numer. Anal.*, 31:1645–1661, 1994.

[39] Howard Elman, David Silvester, and Andy Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2005.

[40] M. Embree. How descriptive are GMRES convergence bounds? Technical Report NA 99/08, Oxford University Computing Laboratory, 1999.

[41] M. Engel and M. Griebel. A multigrid method for constrained optimal control problems. Technical report, SFB-Preprint 406, Sonderforschungsbereich 611, Rheinische Friedrich-Wilhelms-Universität Bonn, 2008.

[42] Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, 1998.

[43] B. Fischer. *Polynomial Based Iteration Methods for Symmetric Linear Systems*. Wiley-Teubner, 1996.

[44] M Fortin and R. Glowinski. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary Value Problems*. North Holland, 1983.

[45] R. Freund and N. Nachtigal. QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315339, 1991.

[46] O. Ghattas and C. Orozco. Massively parallel aerodynamic shape optimization. *Comput. Syst. Eng.*, 1:311–320, 1992.

[47] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press Inc., 1981.

[48] I. Gohberg, P. Lancaster, and L. Rodman. *Matrices and Indefinite Scalar Products*. Birkhäuser-Verlag, 1983.

[49] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[50] G.H. Golub and M.D. Kent. Estimates of eigenvalues for iterative methods. *Math. Comp.*, 53:249–263, 1989.

[51] G.H. Golub and R.S. Varga. Chebyshev semi iterative methods, successive overrelaxation iterative methods and second order Richardson iterative methods. *Numer. Math.*, 3(1):147–156, 1961.

[52] Nicholas I. M. Gould, Mary E. Hribar, and Jorge Nocedal. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM J. Sci. Comput.*, 23(4):1376–1395, 2001.

[53] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, 1997.

[54] Anne Greenbaum, Vlastimil Pták, and Zdeněk Strakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM J. Matrix Anal. Appl.*, 17(3):465–469, 1996.

[55] Piotr P. Grinevich and Maxim A. Olshanskii. An iterative method for the Stokes-type problem with variable viscosity. *SIAM J. Sci. Comput.*, 31:3959–3978, 2009.

[56] E. Haber and U. Ascher. Preconditioned all–at–once methods for large sparse parameter estimation problems. *Inverse Problems*, 17:1847–1864, 2000.

[57] E. Haber and L. Hanson. Model problems in PDE-constrained optimization. Technical Report TR-2007-009, Emory University, 2007.

[58] W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations.* Springer-Verlag, 1994.

[59] W. Hackbush. *Multi-Grid methods and applications.* Springer-Verlag, 1985.

[60] Heinkenschloss and H. Nguyen. Neumann-Neumann domain decomposition preconditioners for linear-quadratic elliptic optimal control problems. *SIAM J. Sci. Comput.*, 28:1001–1028, 2006.

[61] M Heinkenschloss and D. Leykekhman. Local error estimates for SUPG solution of advection-diffusion dominated elliptic linear-quadratic optimal control problems. *SIAM J. Numer. Anal.*, 47(6):4607–4638, 2010.

[62] M. Heinkenschloss, D. C. Sorensen, and K. Sun. Balanced truncation model reduction for a class of descriptor systems with application to the Oseen equations. *SIAM J. Sci. Comput.*, 30(2):1038–1063, 2008.

[63] R. Herzog and E. Sachs. Preconditioned conjugate gradient method for optimal control problems with control and state constraints. Technical Report Preprint-Number SPP1253-088, Deutsche Forschungsgemeinschaft, Priority Program 1253, 2009.

[64] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, 49:409–436, 1952.

[65] N.J. Higham. *Accuracy and Stabitily of Numerical Algorithms.* SIAM, 2nd edition edition, 2002.

[66] M. Hintermüller and K. Kunisch. Feasible and non-interior path-following in constrained minimization with low multiplier regularity. *SIAM J. Control Optim.*, 45:1198–1221, 2006.

[67] M. Hintermüller and K. Kunisch. Path-following methods for a class of constrained minimization problems in function space. *SIAM J. Optim.*, 17:159–187, 2006.

[68] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE Constraints.* Mathematical Modelling: Theory and Applications. Springer, 2008.

[69] C. R. Horn and R. A. Johnson. *Martix Analysis.* Cambridge University Press, 1990.

[70] T.J.R. Hughes and A. Brooks. *Finite Element Methods for Convection Dominated Flows*, chapter "A multidimensional upwind scheme with no crosswind diffusion", pages 120–131. AMD-vol 34. ASME, 1979.

[71] Kazufumi Ito and Karl Kunisch. Augmented Lagrangian–SQP methods for nonlinear optimal control problems of tracking type. *SIAM J. Control Optim.*, 34(3):874–891, 1996.

[72] P Jiránek and M Rozložník. Maximum attainable accuracy of inexact saddle point solvers. *SIAM J. Matrix Anal. Appl.*, 29(4):12971321, 2008.

[73] O.G. Johnson, C.A. Micchelli, and G. Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM J. Numer. Anal.*, 20:362–376, 1983.

[74] Carsten Keller, Nicholas I. M. Gould, and Andrew J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM J. Matrix Anal. Appl.*, 21(4):1300–1317, 2000.

[75] Axel Klawonn. Block-triangular preconditioners for saddle point problems with a penalty term. *SIAM J. Sci. Comput.*, 19:172–184, 1998.

[76] M.V. Klibanov and T.R. Lucas. Numerical solution of a parabolic inverse problem in optical tomography using experimental data. *SIAM J. Appl. Math.*, 59:6516–6534, 1999.

[77] Cornelius Lanczos. Iteration method for the solution of the eigenvalue problem of linear differential and integal operators. *J. Res. Nat. Bur. Stand.*, 45:255–282, 1950.

[78] J. Liesen and B. N. Parlett. On nonsymmetric saddle point matrices that allow conjugate gradient iterations. *Numer. Math.*, 108:605–624, 2008.

[79] J. L. Lions. *Optimal Control of Systems*. Springer, 1968.

[80] T.A. Manteuffel. The Tchebychev iteration for nonsymmetric linear systems. *Numer. Math.*, 28:307–327, 1977.

[81] T.A. Manteuffel. Adaptive procedure for estimation of parameter for the nonsymmetric Tchebychev iteration. *Numer. Math.*, 28:187–208, 1978.

[82] J. C. Mason and D. C. Handscomb. *Chebyshev polynomials*. CRC Press, 2003.

[83] H. Maurer and J. Zowe. First and second order necessary and sufficient optimality conditions for infinite-dimensional programming problems. *Math. Program.*, 16:98–110, 1979.

[84] G Meurant. *Computer Solution of Large Linear Systems*. North Holland, 1999.

[85] A. Meyer and T. Steidten. Improvement and experiments on the Bramble-Pasciak type CG for mixed problems in elasticity. Technical report, TU Chemnitz, Germany, 2001.

[86] Christian Meyer, Arnd Rösch, and Fredi Tröltzsch. Optimal control of PDEs with regularized pointwise state constraints. *Comput. Optim. Appl.*, 33:209–228, 2006.

[87] Malcolm F. Murphy, Gene H. Golub, and Andrew J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.*, 21(6):1969–1972, 2000.

[88] Pekka Neittaanmaki, Jürgen Sprekels, and Dan Tiba. *Optimization of Elliptic Systems*. Springer, 2006.

[89] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

[90] D.P. O'Leary. Yet another polynomial preconditioner for the conjugate gradient algorithm. *Linear Algebra Appl.*, 29:377–388, 1991.

[91] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.

[92] Alfio Quarteroni, Gianluigi Rozza, Luca Dede, and Annalisa Quaini. *IFIP International Federation for Information Processing*, volume 199 of *System Modeling and Optimization*, chapter "Numerical approximation of a control problem for advection-diffusion processes", pages 261–273. Springer, 2006.

[93] A. Ramage. A multigrid preconditioner for stabilised discretisations of advection-diffusion problems. *J. Comput. Appl. Math.*, 110(1):187 – 203, 1999.

[94] Alison Ramage and Howard C. Elman. Some observations on multigrid convergence for convection-diffusion equations. *Comput. Vis. Sci.*, 10(1):43–56, 2007.

[95] Tyrone Rees, H. Sue Dollar, and Andrew J. Wathen. Optimal solvers for PDE-constrained optimization. *SIAM J. Sci. Comput.*, 32(1):271–298, 2010.

[96] Tyrone Rees and Martin Stoll. Block triangular preconditioners for PDE-constrained optimization. Technical Report 15/09, OCCAM, Oxford University Mathematical Institute, March 2009. (to appear in NLAA).

[97] Tyrone Rees, Martin Stoll, and Andrew J. Wathen. All-at-once preconditioning in PDE-constrained optimization, August 2009. (to appear in Kybernetika for special issue on Algorithmy meeting, Podbansk, Slovakia).

[98] H-G Roos, M. Stynes, and L. Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations*. Springer, 2008.

[99] Y. Saad. Practical use of polynomial preconditioning for the conjugate gradient method. *SIAM J. Sci. Comput.*, 6:865–881, 1985.

[100] Y. Saad. *Iterative methods for sparse linear systems*. PWS Publishing, Boston, 1996.

[101] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[102] Saad Y. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14:461–469, 1993.

[103] Joachim Schöberl and Walter Zulehner. Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems. *SIAM J. Matrix Anal. Appl.*, 29(3):752–773, 2007.

[104] Joachim Schöberl, Walter Zulehner, and René Simon. A robust multigrid method for elliptic optimal control problems. Technical Report 2010-01, Institute of Computational Mathematics, Johannes Kepler University, Linz, 2010.

[105] Ajit Shenoy, Matthias Heinkenschloss, and Eugene M. Cliff. Airfoil design by an all-at-once method. *Int. J. Comput. Fluid Dyn.*, 11:3–25, 1998.

[106] D.J. Silvester and A.J. Wathen. Fast iterative solution of stabilised Stokes systems Part II: Using general block preconditioners. *SIAM J. Numer. Anal.*, 31:1352–1367, 1994.

[107] V Simoncini and D. Szyld. Flexible inner-outer Krylov subspace methods. *SIAM J. Numer. Anal.*, 40:2219–2239, 2003.

[108] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10:3652, 1989.

[109] Martin Stoll. *Solving Linear Systems using the Adjoint*. PhD thesis, University of Oxford, 2009.

[110] Martin Stoll and Andy Wathen. Combination preconditioning and the Bramble–Pasciak$^+$ preconditioner. *SIAM J. Matrix Anal. Appl.*, 30(2):582–608, 2008.

[111] Martin Stoll and Andy Wathen. Preconditioning for active set and projected gradient methods as semi-smooth Newton methods for PDE-constrained optimization with control constraints. Technical Report 09/25, Oxford Centre for Collaborative Applied Mathematics, 2009.

[112] Gilbert Strang and Geroge J. Fix. *An analysis of the Finite Element Method*. Wellesley Cambridge Press, 1973.

[113] Angus E. Taylor. *Introduction to Functional Analysis*. John Wiley & Sons, 1958.

[114] H.S. Thorne. Properties of linear systems in PDE-constrained optimization. Part i: Distributed control. Technical Report RAL-TR-2009-017, Rutherford Appleton Laboratory, 2009.

[115] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.

[116] Lloyd N. Trefethen and M. Embree. *Spectra and Pseudospectra*. Princeton, 2005.

[117] F. Tröltzsch. On finite element error estimates for optimal control problems with elliptic PDEs. In *Proceedings of the Conference "Large-Scale Scientific Computations"*, to appear in Springer Lect. Notes in Comp. Sci., 2009.

[118] F. Tröltzsch. *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*. American Mathematical Society, 2010.

[119] U Trottenberg, C Oosterlee, and A Schüller. *Multigrid*. Academic Press, 2000.

[120] H. 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.*, 13:631–644, 1992.

[121] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, 2003.

[122] M.B. van Gijzen. Conjugate gradient-like solution algorithms for the mixed finite element approximation of the biharmonic equation, applied to plate bending problems. *Comput. Methods Appl. Mech. Engrg.*, 121:121–136, 1995.

[123] R.S. Varga. *Matrix Iterative Analysis*. Prentice Hall, 1962.

[124] B. Vexler. Finite element approximation of elliptic Dirichlet optimal control problems. *Numer. Func. Anal. Opt.*, 28:957 – 973, 2007.

[125] A. J. Wathen. Realistic eigenvalue bounds for the Galerkin mass matrix. *IMA J Numer Anal*, 7(4):449–457, 1987.

[126] A. J. Wathen and T. Rees. Chebyshev semi-iteration in preconditioning for problems including the mass matrix. *Electronic Transactions on Numerical Analysis*, 34:125–135, 2009.

[127] Andrew Wathen and David Silvester. Fast iterative solution of stabilised Stokes systems. part I: Using simple diagonal preconditioners. *SIAM J. Numer. Anal.*, 30(3):630–649, 1993.

[128] P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons, 1992.

[129] T. G. Wright. EigTool. Software available at `http://www.comlab.ox.ac.uk/pseudospectra/eigtool`, 2002.

[130] Walter Zulehner. Analysis of iterative methods for saddle point problems: a unified approach. *Math. Comput.*, 71(238):479–505, 2001.