Report no. 08/10

Optimal solvers for PDE-Constrained Optimization

Tyrone Rees
Oxford University Computing Laboratory

H. Sue Dollar
Rutherford Appleton Laboratory

Andrew J. Wathen
Oxford University Computing Laboratory

Optimization problems with constraints which require the solution of a partial differential equation arise widely in many areas of the sciences and engineering, in particular in problems of design. The solution of such PDE-constrained optimization problems is usually a major computational task. Here we consider simple problems of this type: distributed control problems in which the 2- and 3-dimensional Poisson problem is the PDE. The large dimensional linear systems which result from discretization and which need to be solved are of saddle-point type. We introduce two optimal preconditioners for these systems which lead to convergence of symmetric Krylov subspace iterative methods in a number of iterations which does not increase with the dimension of the discrete problem. These preconditioners are block structured and involve standard multigrid cycles. The optimality of the preconditioned iterative solver is proved theoretically and verified computationally in several test cases. The theoretical proof indicates that these approaches may have much broader applicability for other partial differential equations.

Key words and phrases: Saddle-point problems, PDE-constrained optimization, preconditioning, optimal control, linear systems, all-at-once methods

Oxford University Computing Laboratory
Numerical Analysis Group
Wolfson Building
Parks Road
Oxford, England OX1 3QD

May, 2009

1 Introduction

In this paper, we consider the distributed control problem which consists of a cost functional (1.1) to be minimized subject to a partial differential equation problem posed on a domain $\Omega \subset \mathbb{R}^2$ or $\mathbb{R}^3$:

$$\min_{u,f} \frac{1}{2}||u - \hat{u}||^2_{L_2(\Omega)} + \beta||f||^2_{L_2(\Omega)} \tag{1.1}$$

$$\text{subject to} \qquad -\nabla^2 u = f \text{ in } \Omega \tag{1.2}$$

$$\text{with} \quad u = g \text{ on } \partial\Omega_1 \qquad \text{and} \qquad \frac{\partial u}{\partial n} = g \text{ on } \partial\Omega_2, \tag{1.3}$$

where $\partial\Omega_1 \cup \partial\Omega_2 = \partial\Omega$ and $\partial\Omega_1$ and $\partial\Omega_2$ are distinct.

Such problems were introduced by J.L. Lions in [21]. Here, the function $\hat{u}$ (the 'desired state') is known, and we want to find u which satisfies the PDE problem and is as close to $\hat{u}$ as possible in the L_2 norm sense. In order to achieve this, the right hand side of the PDE, f, can be varied. The second term in the cost functional (1.1) is added because, in general, the problem would be ill-posed, and so needs this Tikhonov regularization term. The Tikhonov parameter β needs to be determined, although it is often selected a priori—a value around $\beta = 10^{-2}$ is commonly used (see [10],[15],[19]). We include a graph of $\|u - \hat{u}\|$ vs $\log(\beta)$ in Section 5 to demonstrate what are good values of β.

The above problem involves only the simple Poisson equation as the PDE. Our methods are not specific to only this PDE: all that is required is an effective preconditioner—preferably an optimal preconditioner—for the PDE problem. We discuss possible generalization to other PDEs of our two preconditioning methods in Section 6. Here for the Laplacian we employ standard multigrid cycles with both geometric ([9], [27], [16]) and algebraic ([8], [27, Appendix A]) multigrid procedures. For other elliptic PDEs multigrid cycles could also form an important part of algorithms for control problems based on the ideas presented here. The above problem does not involve bound nor inequality constraints; it is also possible that these more general constraints could be included though we have not considered this here.

We also consider one example with boundary control, namely on a domain $\Omega \subset \mathbb{R}^2$, find:

$$\min_{u,f} \frac{1}{2}||u - \hat{u}||^2_{L_2(\Omega)} + \beta||g||^2_{L_2(\partial\Omega)} \tag{1.4}$$

$$\text{s. t.} \quad -\nabla^2 u = 0 \text{ in } \Omega \tag{1.5}$$

$$\frac{\partial u}{\partial n} = g \text{ on } \partial\Omega. \tag{1.6}$$

Here the control, g, is the unknown quantity, which is included in the boundary condition.

In PDE-constrained optimization there is the choice as to whether to discretize-then-optimize or optimize-then-discretize, and there are differing opinions regarding which

route to take (see Collis and Heinkenschloss [10] for a discussion). We have chosen to discretize-then-optimize, as then we are guaranteed symmetry in the resulting linear system. The underlying optimization problems are naturally self-adjoint and by this choice we avoid non-symmetry due to discretization that can arise with the optimize-then-discretize approach (as shown in, for example, Collis and Heinkenschloss [10]). We are then able to use symmetric iterative methods—in particular we use MINRES ([25]) and a projected Conjugate Gradient (PPCG) method ([14]) —with the consequent advantage of rigorous convergence bounds and constant work per iteration not enjoyed by any of the wide variety of non-symmetric Krylov subspace iterative methods (see e.g., [11]). This still leaves the crucial question of preconditioning and this is the main contribution of this paper. We derive and analyse both theoretically and by computation two preconditioning approaches which lead to optimal solution of the PDE-constrained optimization problem. That is preconditioners which when employed with MINRES or PPCG respectively give a solution algorithm which requires $O(n)$ computational operations to solve a discrete problem with n degrees of freedom.

We employ the Galerkin finite element method for discretization here, but see no reason why other approximation methods could not be used with our approach.

We comment that for the specific problem as above for the Poisson equation, Schöberl and Zulehner ([26]) have recently developed a preconditioner based on a non-standard multigrid procedure which is both optimal with respect to the problem size *and* with respect to the choice of regularization parameter, β. It is not so clear how this method would generalize to other PDEs. Biros and Dogan ([5]) 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. We note that the approximate reduced Hessian approximation used by Haber and Asher ([15]) and Biros and Ghattas ([4]) also leads to a preconditioner with $h-$independence.

Other solution methods employing multigrid for this and similar classes of problems are described by Asher and Haber([1]), Engel and Griebel([12]), and Borzi([6]). Domain Decomposition and Model Order Reduction ideas are also successfully applied in this context: see for example Heinkenschloss and Nguyen ([17]) and Heinkenschloss, Sorensen and Sun([18]).

In Section 2, we discuss the formulation and structure of our discretized problem. We then use this structure in Sections 3 and 4 to derive optimal preconditions for MINRES and PPCG, respectively. The effectiveness of our proposed preconditioners is illustrated by applying them to five different problems, see Section 5. Finally, we draw our conclusions in Section 6.

2 Formulation and Structure

In order to use finite elements, we require the weak formulation of (1.2) and (1.3). For definiteness and clarity we describe this for the purely Dirichlet problem; the formulation for the mixed and purely Neumann problem is also standard (see for example [11]). The

Dirichlet problem is: find $u \in H^1_g(\Omega) = \{u \ : \ u \in H^1(\Omega), \ u = g \text{ on } \partial\Omega\}$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega v f \qquad \forall v \in H^1_0(\Omega). \tag{2.1}$$

We assume that $V^h_0 \subset H^1_0$ is an n-dimensional vector space of test functions with $\{\phi_1, ..., \phi_n\}$ as a basis. Then, for the boundary condition to be satisfied, we extend the basis by defining functions $\phi_{n+1}, ..., \phi_{n+\partial n}$ and coefficients U_j so that $\sum_{j=n+1}^{n+\partial n} U_j \phi_j$ interpolates the boundary data. Then, if $u_h \in V^h_g \subset H^1_g(\Omega)$, it is uniquely determined by $\mathbf{u} = (U_1 \dots U_n)^T$ in

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

Here the ϕ_i, $i = 1, \dots, n$, define a set of shape functions. We also assume that this approximation is conforming, i.e. $V^h_g = \text{span}\{\phi_1, \dots, \phi_{n+\partial n}\} \subset H^1_g(\Omega)$. Then we get the finite-dimensional analogue of (2.1): find $u_h \in V^h_g$ such that

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega v_h f \qquad \forall v_h \in V^h_0.$$

We also need a discretization of f, as this appears in (1.1). We discretize this using the same basis used for u, so

$$f_h = \sum_{j=1}^{n} F_j \phi_j$$

since it is well known that $f_h = 0$ on $\partial\Omega$. Thus we can write the discrete analogue of the minimization problem as

$$\min_{u_h, f_h} \frac{1}{2} ||u_h - \hat{u}||^2_2 + \beta ||f_h||^2_2 \tag{2.2}$$

$$\text{such that} \qquad \int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega v_h f \qquad \forall v_h \in V^h_0. \tag{2.3}$$

We can write the discrete cost functional as

$$\min_{u_h, f_h} \frac{1}{2} ||u_h - \hat{u}||^2_2 + \beta ||f_h||^2_2 = \min_{\mathbf{u}, \mathbf{f}} \frac{1}{2} \mathbf{u}^T M \mathbf{u} - \mathbf{u}^T \mathbf{b} + \alpha + \beta \mathbf{f}^T M \mathbf{f}, \tag{2.4}$$

where $\mathbf{u} = (U_1, \dots, U_n)^T$, $\mathbf{f} = (F_1, \dots, F_n)^T$, $\mathbf{b} = \{\int \hat{u} \phi_i\}_{i=1\dots n}$, $\alpha = \|\hat{u}\|^2_2$ and $M = \{\int \phi_i \phi_j\}_{i,j=1\dots n}$ is a mass matrix.

We now turn our attention to the constraint: (2.3) is equivalent to finding $\mathbf{u}$ such that

$$\int_\Omega \nabla \left(\sum_{i=1}^{n} U_i \phi_i \right) \cdot \nabla \phi_j + \int_\Omega \nabla \left(\sum_{i=n+1}^{n+\partial n} U_i \phi_i \right) \cdot \nabla \phi_j = \int_\Omega \left(\sum_{i=1}^{n} F_i \phi_i \right) \phi_j, \quad j = 1, \dots, n$$

which is

$$\sum_{i=1}^{n} U_i \int_\Omega \nabla\phi_i \cdot \nabla\phi_j = \sum_{i=1}^{n} F_i \int_\Omega \phi_i\phi_j - \sum_{i=n+1}^{n+\partial n} U_i \int_\Omega \nabla\phi_i \cdot \nabla\phi_j, \quad j = 1, \ldots, n$$

or

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

where the matrix $K = \{\int \nabla\phi_i \cdot \nabla\phi_j\}_{i,j=1\ldots n}$ is the discrete Laplacian (the stiffness matrix) and $\mathbf{d}$ contains the terms coming from the boundary values of u_h. Thus (2.4) and (2.5) together are equivalent to (2.2) and (2.3).

One way to solve this minimization problem is by considering the Lagrangian

$$\mathcal{L} := \frac{1}{2}\mathbf{u}^T M\mathbf{u} - \mathbf{u}^T\mathbf{b} + \alpha + \beta\mathbf{f}^T M\mathbf{f} + \lambda^T(K\mathbf{u} - M\mathbf{f} - \mathbf{d}),$$

where λ is a vector of Lagrange multipliers. Using the stationarity conditions of $\mathcal{L}$, we find that $\mathbf{f}$, $\mathbf{u}$ and λ are defined by the linear system

$$\underbrace{\begin{bmatrix} 2\beta M & 0 & -M \\ 0 & M & K^T \\ -M & K & 0 \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \mathbf{f} \\ \mathbf{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{2.6}$$

Note that this system of equations has saddle-point system structure, i.e. it is of the form

$$\begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}, \tag{2.7}$$

where $A = \begin{bmatrix} 2\beta M & 0 \\ 0 & M \end{bmatrix}$, $B = [-M \quad K]$, $C = 0$.

This system is usually very large—each of the blocks K is itself a discretization of the PDE—and sparse, since as well as the zero blocks, K and M are themselves sparse because of the finite element discretization. Thus matrix-vector multiplications can be easily achieved and the work in a symmetric Krylov subspace iteration method will be linear at each iteration. In general the system is symmetric and indefinite, so the Minimal Residual (MINRES) method of Paige and Saunders [25] is robustly applicable and is the method of choice for such systems when a symmetric positive definite preconditioner is employed: one of our optimal preconditioners is of this type. Our second preconditioner is a *constraint preconditioner* [20] which we may use in conjunction with the projected conjugate gradient (PPCG) method [14]. The crucial step to ensure that acceptably rapid convergence is guaranteed is preconditioning: we consider in the next two sections our two preconditioning approaches.

We now consider a boundary control problem, as in (1.4)-(1.6). In weak formulation of (1.5)-(1.6) we find $u \in H^1(\Omega)$ such that

$$\int_\Omega \nabla u \nabla\phi \,\mathrm{d}x \; - \int_{\partial\Omega} u\gamma(\phi) \,\mathrm{d}s \; = \int_\Omega f\phi \,\mathrm{d}x \qquad \forall\phi \in H^1(\Omega),$$

where γ is the trace operator. We discretize the problem using the finite element method, using a triangulation where the total number of vertices is m_t and the number of vertices on the boundary is m_b. Then on optimizing as above we get the following system:

$$\begin{bmatrix} 2\beta M_g & 0 & -E^T \\ 0 & M_u & K^T \\ -E & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{g} \\ \mathbf{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{2.8}$$

Here $K = \{\int \nabla\phi_i \cdot \nabla\phi_j\}_{i,j=1\ldots m_t} \in \mathbb{R}^{m_t \times m_t}$ is the usual stiffness matrix, $M_u = \{\int \phi_i\, \phi_j\}_{i,j=1\ldots m_t} \in \mathbb{R}^{m_t \times m_t}$ is the mass matrix, where $\{\phi_i\}$ is the finite element basis as defined above. If we define $\{\widehat{\phi_i}\}$ to be the analogous functions defined on the boundary, then we can write $E = \{\int \widehat{\phi_j}\ \phi_i\}_{i=1\ldots m_t, j=1\ldots m_b} \in \mathbb{R}^{m_t \times m_b}$ and $M_g = \{\int \widehat{\phi_j}\ \widehat{\phi_i}\}_{i,j=1\ldots m_t} \in \mathbb{R}^{m_b \times m_b}$. The constant vectors are given by $\mathbf{d} = \{\int_\Omega f\phi_i\}_{i=1\ldots m_t} \in \mathbb{R}^{m_t}$ and $\mathbf{b} = \{\int_\Omega \hat{u}\phi_i\}_{i=1\ldots m_t} \in \mathbb{R}^{m_t}$.

Note that in this case the blocks in the system (2.8) are not all square, and we include this example to demonstrate how at least one of our methods described in Section 3 can be applied in such cases. The methods described in Section 4 are not immediately applicable for this class of problem and will form the subject of a follow-up paper in which similar ideas are used by only replicating a subset of the constraints in the preconditioner and using a different projected iterative method.

We comment that there are a number of generalizations of the problem above which lead to systems with similar algebraic structure. The $L_2(\Omega)$ norms in (1.1) could be changed to any Hilbert space norms and the (1,1) and (2,2) blocks would correspondingly be the Gram matrices associated with these norms: our technique with appropriate modification should handle this.

Another common case would be when the (2,2) block in (2.6) is singular. This would occur when, for example, one only wants to control u on part of Ω, or if one has few measurements. In this case it is not clear at this stage how our methods in Section 3 could be applied but there is scope for the methods proposed in Section 4 because these do not assume that A is non-singular.

3 Block Diagonally Preconditioned MINRES

In general, the system (2.6) will be symmetric but indefinite, so we solve it using the MINRES algorithm: this is a Krylov subspace method for symmetric linear systems. MINRES has to be coupled with a preconditioner to get satisfactory convergence - i.e. we want to find a matrix (or a linear process) $\mathcal{P}$ for which $\mathcal{P}^{-1}\mathcal{A}$ has better spectral properties (and such that $\mathcal{P}^{-1}\mathbf{v}$ is cheap to evaluate for any given vector $\mathbf{v}$). We then solve a symmetric preconditioned system equivalent to

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

The aim of preconditioning is to choose a matrix $\mathcal{P}$ such that the eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$ are clustered. The following result by Murphy, Golub and Wathen [24] illustrates this idea:

Theorem 1 *If*

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

is preconditioned by

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

Then the preconditioned matrix $\mathcal{T} = \mathcal{P}^{-1}\mathcal{A}$ *satisfies*

$$\mathcal{T}(\mathcal{T} - I)(\mathcal{T}^2 - \mathcal{T} - I) = 0.$$

This shows us that $\mathcal{T}$ is diagonalizable and has at most four distinct eigenvalues $(0, 1, \frac{1\pm\sqrt{5}}{2})$ or only the three non-zero eigenvalues if $\mathcal{T}$ is nonsingular. This means that the Krylov subspace $\mathcal{K}(\mathcal{T}; \mathbf{r}) = \text{span}(\mathbf{r}, \mathcal{T}\mathbf{r}, \mathcal{T}^2\mathbf{r}, \dots)$ will be of dimension at most three if $\mathcal{T}$ is nonsingular or four if $\mathcal{T}$ is singular. Therefore, any Krylov subspace method with an optimality property (such as MINRES) will terminate in at most three iterations (with exact arithmetic).

If we apply this approach to the matrix in our saddle-point system (2.6) then we obtain the preconditioner

$$\mathcal{P}_{\text{MGW}} = \begin{bmatrix} 2\beta M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & \frac{1}{2\beta}M + KM^{-1}K^T \end{bmatrix}.$$

MINRES with this preconditioner will always terminate (in exact arithmetic) in at most three steps and so satisfies one requirement of a preconditioner. However, it fails on another count as it is, in general, not cheap to solve a system with $\mathcal{P}_{\text{MGW}}$. However, we could still make use of the properties of this preconditioner by approximating it in such a way that the eigenvalues remain clustered. Looking at the structure of $\mathcal{P}_{\text{MGW}}$, the mass matrices in the (1,1) and the (2,2) blocks do not pose too much of a problem: they can be cheaply solved by, for example, using PCG with the diagonal as the preconditioner, as shown in [29]. Thus the difficulty comes from the (3,3) block, which is the only part that contains the PDE.

One way to approximate this is to consider only the dominant term in the (3,3) block which is, for all but the very smallest values of β, the $KM^{-1}K^T$ term, thus forming the preconditioner

$$\mathcal{P}_{\text{D1}} = \begin{bmatrix} 2\beta M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & KM^{-1}K^T \end{bmatrix}. \tag{3.1}$$

The following result, which is an application and extension of a result in [2], tells us about the clustering of the eigenvalues using this preconditioner:

Proposition 2 *Let* λ *be an eigenvalue of* $\mathcal{P}_{\text{D1}}^{-1}\mathcal{A}$*, with* $\mathcal{P}_{\text{D1}}$ *defined by (3.1) and* $\mathcal{A}$ *defined as in (2.6) . Then either* $\lambda = 1$,
$\frac{1}{2}(1 + \sqrt{1 + 4\sigma_1}) \leq \lambda \leq \frac{1}{2}(1 + \sqrt{1 + 4\sigma_m})$ *or* $\frac{1}{2}(1 - \sqrt{1 + 4\sigma_m}) \leq \lambda \leq \frac{1}{2}(1 - \sqrt{1 + 4\sigma_1})$, *where* $0 \leq \sigma_1 \leq \cdots \leq \sigma_m$ *are the eigenvalues of* $\frac{1}{2\beta}(KM^{-1}K^T)^{-1}M + I$.

Proof First note that the eigenvalues of $\mathcal{P}_{\rm D1}^{-1}\mathcal{A}$ are identical to the eigenvalues of $\tilde{\mathcal{A}} := \mathcal{P}_{\rm D1}^{-\frac{1}{2}}\mathcal{A}\mathcal{P}_{\rm D1}^{-\frac{1}{2}}$, as this is just a the result of a similarity transformation. It is readily seen that

$$\tilde{\mathcal{A}} = \begin{bmatrix} I & 0 & \tilde{K_1}^T \\ 0 & I & \tilde{K_2}^T \\ \tilde{K_1} & \tilde{K_2} & 0 \end{bmatrix} \text{ or equivalently } \begin{bmatrix} I & B^T \\ B & 0 \end{bmatrix}$$

where $\tilde{K_1} = -\frac{1}{\sqrt{2\beta}}(KM^{-1}K^T)^{-\frac{1}{2}}M^{\frac{1}{2}}$, $\tilde{K_2} = (KM^{-1}K^T)^{-\frac{1}{2}}KM^{-\frac{1}{2}}$ and $B = [\tilde{K_1}\ \tilde{K_2}]$.

Let $\left(\lambda, [\mathbf{x}\ \mathbf{y}]^T\right)$ be an eigenpair for $\tilde{\mathcal{A}}$. Then $\begin{bmatrix} I & B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$. This can be written as

$$\begin{aligned} \mathbf{x} + B^T\mathbf{y} &= \lambda\mathbf{x} \\ B\mathbf{x} &= \lambda\mathbf{y} \end{aligned}$$

By inspection, one solution of this problem is $\lambda = 1$, and this has multiplicity n with eigenvectors of the form $\begin{bmatrix} \mathbf{x} & \mathbf{0} \end{bmatrix}^T$, where $B\mathbf{x} = \mathbf{0}$.

Now we will consider the two cases (I) $\lambda > 0$ and (II) $\lambda < 0$ separately. λ cannot equal 0, since $\tilde{\mathcal{A}}$ is non-singular.

CASE (I): $\lambda > 0$ and $\lambda \neq 1$ (the case $\lambda = 1$ has been treated above). Clearly

$$\mathbf{x} = -(1-\lambda)^{-1}B^T\mathbf{y}$$

and

$$\begin{aligned} -(1-\lambda)^{-1}BB^T\mathbf{y} &= \lambda\mathbf{y}, \\ -(1-\lambda)^{-1}\mathbf{y^T}BB^T\mathbf{y} &= \lambda\mathbf{y^T}\mathbf{y}, \\ -(1-\lambda)\lambda &= \frac{\mathbf{y^T}BB^T\mathbf{y}}{\mathbf{y^T}\mathbf{y}}. \end{aligned}$$

Now $\frac{\mathbf{y^T}BB^T\mathbf{y}}{\mathbf{y^T}\mathbf{y}} = \frac{||B\mathbf{y}||^2}{||\mathbf{y}||^2} =: b$, so

$$\lambda^2 - \lambda - b = 0,$$

i.e.

$$\lambda = \frac{1 \pm \sqrt{1+4b}}{2}.$$

But by assumption $\lambda > 0$, so

$$\lambda = \frac{1 + \sqrt{1+4b}}{2}.$$

We know that $\sigma_1 \leq b \leq \sigma_m$, where σ_i are the eigenvalues of BB^T, so we have

$$\frac{1+\sqrt{1+4\sigma_1}}{2} \leq \lambda \leq \frac{1+\sqrt{1+4\sigma_m}}{2}.$$

CASE (II): $\lambda < 0$. Let $\mu = -\lambda$. Then from above,

$$\begin{aligned} \mathbf{x} &= -(1+\mu)^{-1}B^T\mathbf{y}, \\ b &= \mu(1+\mu), \\ \mu^2 + \mu - b &= 0. \end{aligned}$$

So

$$\mu = \frac{-1 \pm \sqrt{1+4b}}{2}.$$

Again, $\mu > 0$ by assumption, so

$$\mu = \frac{-1 + \sqrt{1+4b}}{2}$$

or

$$\lambda = \frac{1 - \sqrt{1+4b}}{2},$$

i.e.

$$\frac{1 - \sqrt{1+4\sigma_m}}{2} \leq \lambda \leq \frac{1 - \sqrt{1+4\sigma_1}}{2}.$$

Finally,

$$\begin{aligned} BB^T &= \left[\begin{array}{cc} -\frac{1}{\sqrt{2\beta}}(KM^{-1}K^T)^{-\frac{1}{2}}M^{\frac{1}{2}} & (KM^{-1}K^T)^{-\frac{1}{2}}KM^{-\frac{1}{2}} \end{array} \right] \left[\begin{array}{c} -\frac{1}{\sqrt{2\beta}}M^{\frac{1}{2}}(KM^{-1}K^T)^{-\frac{1}{2}} \\ M^{-\frac{1}{2}}K^T(KM^{-1}K^T)^{-\frac{1}{2}} \end{array} \right] \\ &= \frac{1}{2\beta}(KM^{-1}K^T)^{-\frac{1}{2}}M(KM^{-1}K^T)^{-\frac{1}{2}} + I \end{aligned}$$

and so the eigenvalues of BB^T are the same as those of $\frac{1}{2\beta}(KM^{-1}K^T)^{-1}M + I$, as required. ■

We can use this general result to obtain more concrete bounds that are dependent both on the PDE in the problem being considered and on what finite element discretization is used. In our tests, we have discretized the problem (1.1) using bi-linear quadrilateral $\mathbf{Q_1}$ finite elements, and for this choice one can prove the following.

Corollary 2A *Let λ be an eigenvalue of $\mathcal{P}_{\mathrm{D1}}^{-1}\mathcal{A}$, with $\mathcal{P}_{\mathrm{D1}}$ and $\mathcal{A}$ as defined in Proposition 2. Then λ satisfies one of*

$$\begin{aligned} \lambda &= 1, \\ \frac{1}{2}\left(1 + \sqrt{5 + \frac{2\alpha_1 h^4}{\beta}}\right) \leq &\ \lambda \ \leq \frac{1}{2}\left(1 + \sqrt{5 + \frac{2\alpha_2}{\beta}}\right) \\ \text{or } \frac{1}{2}\left(1 - \sqrt{5 + \frac{2\alpha_2}{\beta}}\right) \leq &\ \lambda \ \leq \frac{1}{2}\left(1 - \sqrt{5 + \frac{2\alpha_1 h^4}{\beta}}\right), \end{aligned}$$

where $\alpha_1,\ \alpha_2$ are positive constants independent of h and independent of β.

Proof Proposition 2 tells us that the clustering of eigenvalues of the preconditioned system depends on finding the eigenvalues of the matrix $T := I + \frac{1}{2\beta}(K^T M^{-1} K)^{-1} M$. In this case, if λ is an eigenvalue of T, then

$$\begin{aligned} Tx &= \lambda x, \\ \text{i.e. } (\frac{1}{2\beta}(K^T M^{-1} K)^{-1} M + I)\mathbf{x} &= \lambda \mathbf{x}, \\ K^{-1} M K^{-T} M \mathbf{x} &= 2\beta(\lambda - 1)\mathbf{x}. \end{aligned}$$

Here $K = K^T$ and if we let $\mu = 2\beta(\lambda - 1)$, we have

$$(K^{-1}M)^2\mathbf{x} = \mu\mathbf{x}.$$

If ν is an eigenvalue of $K^{-1}M$, then $\mu = \nu^2$, and

$$\begin{aligned} K^{-1}M\mathbf{x} &= \nu\mathbf{x}, \\ \text{i.e.} \quad M\mathbf{x} &= \nu K\mathbf{x}, \\ \text{so} \quad \mathbf{x}^T M\mathbf{x} &= \nu\mathbf{x}^T K\mathbf{x}, \\ \nu &= \frac{\mathbf{x}^T M\mathbf{x}}{\mathbf{x}^T K\mathbf{x}}. \end{aligned}$$

We now make use the following results, which are Proposition 1.29 and Theorem 1.32 respectively in [11], applied to our case.

Theorem 3 *For* **Q1** *approximation on a quasi-uniform subdivision of* $\mathbb{R}^2$ *for which a shape regularity condition holds the mass matrix* M *approximates the scaled identity matrix in the sense that*

$$c_1 h^2 \leq \frac{\mathbf{x}^T M\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \leq c_2 h^2$$

$\forall \mathbf{x} \neq 0 \in \mathbb{R}^n$. *The constants* c_1 *and* c_2 *are independent of both* h *and* β.

Theorem 4 *For* **Q1** *approximation on a quasi-uniform subdivision of* $\mathbb{R}^2$ *for which a shape regularity condition holds the Galerkin matrix* K *satisfies*

$$d_1 h^2 \leq \frac{\mathbf{x}^T K\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \leq d_2$$

$\forall \mathbf{x} \neq 0 \in \mathbb{R}^n$. *The constants* d_1 *and* d_2 *are positive and independent of both* h *and* β.

From Theorem 4 we obtain

$$\frac{1}{d_2} \leq \frac{\mathbf{x}^T\mathbf{x}}{\mathbf{x}^T K\mathbf{x}} \leq \frac{1}{d_1 h^2}.$$

Therefore,

$$\begin{aligned} \frac{c_1 h^2}{d_2} \leq & \quad \nu \quad & \leq \frac{c_2}{d_1}, \\ \left(\frac{c_1}{d_2}\right)^2 h^4 \leq & \quad \nu^2 \quad & \leq \left(\frac{c_2}{d_1}\right)^2, \\ \left(\frac{c_1}{d_2}\right)^2 h^4 \leq & \quad 2\beta(\lambda - 1) \quad & \leq \left(\frac{c_2}{d_1}\right)^2, \\ \frac{1}{2\beta}\left(\frac{c_1}{d_2}\right)^2 h^4 + 1 \leq & \quad \lambda \quad & \leq \frac{1}{2\beta}\left(\frac{c_2}{d_1}\right)^2 + 1. \end{aligned}$$

Hence, for the 2D case, we have the bounds

$$\frac{1}{2\beta}\alpha_1 h^4 + 1 \leq \lambda \leq \frac{1}{2\beta}\alpha_2 + 1, \tag{3.2}$$

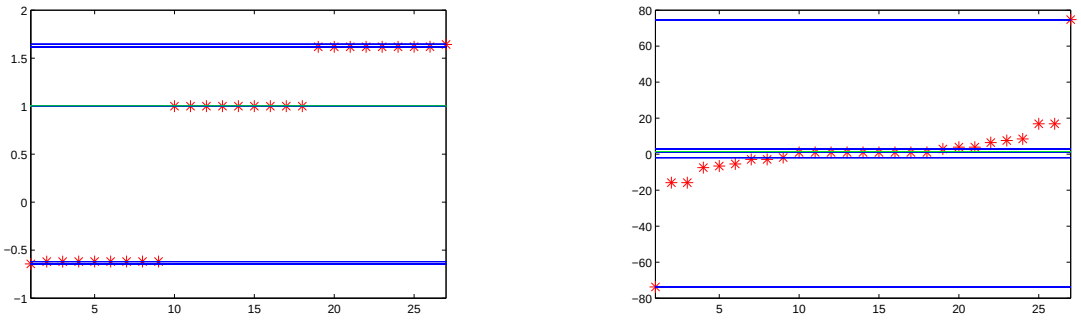
where α_1 and α_2 are constants independent of h and independent of β. For the 3D case, the equivalent results to Theorems 3 and 4 are

$$\begin{aligned} c_1 h^3 \leq \quad & \frac{\mathbf{x}^T M \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \quad \leq c_2 h^3 \ \forall \mathbf{x} \neq 0 \\ \text{and} \quad d_1 h^3 \leq \quad & \frac{\mathbf{x}^T K \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \quad \leq d_2 h \ \forall \mathbf{x} \neq 0. \end{aligned}$$

The extra h on each side will cancel, meaning (3.2) also holds in the 3D case (although the constants will be different). ■

The bounds in Corollary 2A include two constants, α_1 and α_2, which are independent of both h and β but will change depending on the discretization used. In 2D for **Q1** square elements, as used here, Fourier analysis gives that $c_1 = 1/9$, $c_2 = 1$, $d_1 = 2\pi^2$ and $d_2 = 4$, so $\alpha_1 = 1/1296$ and $\alpha_2 = 1/4\pi^2$ for this discretization.

Note also that the bounds depend on h in a multiplicative way only, so they remain bounded away from 0 as $h \to 0$ with a bound independent of h. This suggests that this is an optimal preconditioner in the sense that its performance is independent of the mesh size. Figure 1 shows plots of the actual eigenvalues and the bounds of Corollary 2A for two choices of β. We see that the eigenvalues are much more clustered for the larger value of β, and we see that for values around this our method is most successful. Taking β around this value is common in the literature – see Collis and Heinkenschloss [10], Haber and Ascher [15], Maurer and Mittelmann [22],[23] or Ito and Kunisch [19], for example.



(a) $\beta = 10^{-2}$ (b) $\beta = 10^{-7}$

Figure 1: Eigenvalues of $\mathcal{P}_{\mathrm{D1}}^{-1}\mathcal{A}$ and eigenvalue bounds predicted by Proposition 2 (lines are the predicted bounds, *s are the eigenvalues)

Remark 5 *We have proved bounds for one possible approximation of the ideal preconditioner $\mathcal{P}_{MGW}$, giving us the preconditioner $\mathcal{P}_{D1}$, which was formed by approximating the (3,3) block in the ideal case — $\frac{1}{2\beta}M + KM^{-1}K^T$ — by $KM^{-1}K^T$. Other such approximations are possible, for example, one could approximate the (3,3) block by taking the first term in the sum, $\frac{1}{2\beta}M$. However, in our experience this is only a good approximation for extremely small values of β (around 10^{-12}), which are not very useful in practice due to the highly irregular control functions that result.*

For the values of β in between (around $\beta = 10^{-8}$) we need both the additive terms in the (3,3) block to have an effective preconditioner. We note that in the case we are considering here, with Poisson's equation as the constraint, the (3,3) block is itself an elliptic operator for which effective multigrid methods exist. Therefore, approximating $\frac{1}{2\beta}M + KM^{-1}K^T$ by, for example, a fixed number of V-cycles of this type should give a method which is both β and h independent. However, it is not clear how a such a method would generalize to other PDE constraints.

When using $\mathcal{P}_{\text{D1}}$ as a preconditioner, the main drawback is that at each iteration we have to solve for K and K^T once each, which is equivalent to solving the forward problem twice per iteration. This is costly, especially for more complicated PDEs. As these solves are only needed for the preconditioner, which is itself just an approximation, all we need is to solve these approximately. Thus we want to consider

$$\mathcal{P} = \begin{bmatrix} 2\beta\tilde{M} & 0 & 0 \\ 0 & \tilde{M} & 0 \\ 0 & 0 & \tilde{K}M^{-1}\tilde{K}^T \end{bmatrix}, \tag{3.3}$$

where $\tilde{K}$ and $\tilde{M}$ are some approximations to K and M, respectively. Note that we do not need to approximate M in the Schur complement as a solve with M^{-1} is just a matrix-vector multiplication with the mass matrix. If our approximations are good enough, then the spectral bounds should be close to those shown above. Using (3.3) as a preconditioner will take only slightly more Krylov subspace iterations, but with a solve for $\tilde{K}$ being much faster than, say, a sparse direct solve for K, hence giving us a much more effective preconditioner.

Note that if, for example, the $L_2(\Omega)$ norm in (1.1) were replaced by a Hilbert space norm with Gram matrix H, then we would require approximations $2\beta\tilde{H}$ and $\tilde{H}$ in the (1,1) and (2,2) blocks respectively.

For any choice of PDE in a problem of the type in (1.1), it is likely that there has been work done on solving the forward problem (i.e. solving just for a solution to the PDE) and we propose to draw from ideas here to help us develop effective preconditioners. If we have an effective preconditioner for the forward problem, then we can incorporate it into our methods to give us an effective preconditioner for the PDE constrained optimization problem.

In the case of our PDE, the Poisson equation, a fixed number of multigrid iterations is a good preconditioner [11]. We apply both algebraic and geometric multigrid routines. We thus have two Preconditioners,

$$\mathcal{P}_{D2} = \begin{bmatrix} 2\beta\tilde{M} & 0 & 0 \\ 0 & \tilde{M} & 0 \\ 0 & 0 & \tilde{K}M^{-1}\tilde{K}^T \end{bmatrix} \quad \text{and} \quad \mathcal{P}_{D3} = \begin{bmatrix} 2\beta\tilde{M} & 0 & 0 \\ 0 & \tilde{M} & 0 \\ 0 & 0 & \widehat{K}M^{-1}\widehat{K}^T \end{bmatrix},$$

where $\tilde{K}$ denotes two geometric V-cycles and $\widehat{K}$ denotes two AMG V-cycles of `HSL` package `HSL_MI20` [7] applied via a MATLAB interface. For both multigrid methods we

use the relaxed Jacobi method for the smoother, i.e. if we have to solve an arbitrary system $G\mathbf{u} = \mathbf{f}$ for some matrix $G \in \mathbb{R}^{k\times k}$ and vectors $\mathbf{u}, \mathbf{f} \in \mathbb{R}^k$ for some k, take $D = \mathrm{diag}(G)$ and iterate

$$\mathbf{u}^{(m+1)} = (I - \omega D^{-1}G)\mathbf{u}^{(m)} + \omega D^{-1}\mathbf{f} \tag{3.4}$$

where $\mathbf{u}^{(0)} = \mathbf{0}$ and for some relaxation parameter ω. For 2D problems we use 2 pre- and 2 post-smoothing steps of relaxed Jacobi with the optimal relaxation parameter of $\omega = \frac{8}{9}$ in the 2D case (see [11, Section 2.5]). In 3D, we use 3 pre- and 3 post-smoothing steps of unrelaxed Jacobi, i.e. we take $\omega = 1$ in the above, which is optimal here. We have not experimented excessively, but found that this number of smoothing steps gives a reasonable overall efficiency: using fewer steps, the required number of iterations of MINRES is still mesh independent, though it is higher. We emphasize that this is the most CPU-intensive part of our algorithm. We have used a multigrid procedure here essentially as a 'black box'; we obtain similar results (not presented here) by using a Gauss-Seidel smoother, and by varying the number of smoothing steps per level.

For the mass matrix, M, there are a number of options for the approximation. One could use a lumped mass matrix, or even the diagonal of M – from our experience both of these methods lead to an optimal preconditioner. However, the approximate mass matrix solve is the the cheapest part of the application of the preconditioner, and if we can have a more accurate approximation here, this will bring down the number of (outer) MINRES iterations needed – i.e. we will need fewer of the relatively expensive multigrid solves – hence reducing the total solution time. What we would like to use is a few steps of the preconditioned conjugate gradient method with, say, the diagonal as a preconditioner applied to the matrix, as this will give us a good approximation. However, PCG is not linear in the right hand side, so we cannot use it as a preconditioner without applying a flexible outer Krylov iteration.

The Chebyshev semi-iteration [13] is a method of accelerating convergence of a simple iterative method which is linear, so we can employ it here. In 2D, we use relaxed Jacobi with a relaxation parameter of $\frac{4}{5}$, which, when applied to a $\mathbf{Q_1}$ mass matrix, gives an iteration matrix with eigenvalues satisfying $|\lambda| \leq \frac{4}{5} =: \rho$. In 3D, the optimal relaxation parameter is $\frac{4}{7}$, which gives eigenvalues such that $|\lambda| \leq \frac{13}{14} =: \rho$. In both cases, if we want to solve $M\mathbf{u} = \mathbf{f}$, say, then the k^{th} iterate of the Chebyshev semi-iteration is given by

$$\mathbf{w}^{(k)} = \sum_{i=0}^{k} \nu_i \mathbf{u}^{(i)},$$

where $\mathbf{u}^{(i)}$ are the iterates of the underlying iterative method (so $\mathbf{u}^{(i)} = S\mathbf{u}^{(i-1)} + g$ where S is some iteration matrix, defined here by relaxed Jacobi) and ν_i are the coefficients of the scaled Chebyshev polynomial $\hat{T}_k(z) = \frac{T_k(z/\rho)}{T_k(1/\rho)}$. This can be implemented more efficiently by performing the iteration

$$\mathbf{w}^{(k+1)} = w_{k+1}(S\mathbf{w}^{(k)} + g - \mathbf{w}^{(k-1)}) + \mathbf{w}^{(k-1)}, \tag{3.5}$$

where $w_{k+1} = \frac{T_k(1/\rho)}{\rho T_{k+1}(1/\rho)}$ (see Varga [28, Chapter 5]). It is very cheap to carry out an iteration using this scheme. Moreover, we get the following convergence result, which

shows this method has essentially the same convergence behaviour as classical conjugate gradients:

$$||\mathbf{u} - \mathbf{w}^{(k)}||_2 \leq \max_{r\in[-\rho,\rho]} |\hat{T}_k(r)| \; ||\mathbf{u} - \mathbf{u}^{(0)}||_2. \tag{3.6}$$

Indeed this bound using Chebyshev polynomials is the one usually applied for Conjugate Gradient convergence. This suggests that a fixed number of these iterations will give us a good approximation to M. This is a linear operation which is cheap to implement, so it is valid to use as a preconditioner with a standard Krylov subspace iteration such as MINRES. We therefore let $\tilde{M}$ in $\mathcal{P}_{\mathrm{D2}}$ and $\mathcal{P}_{\mathrm{D3}}$ denote 20 iterations of the Chebyshev semi-iteration, as defined above. In 2D, $\max_{r\in[-\rho,\rho]} |\hat{T}_{20}(r)| \approx 10^{-6}$. This bound shows that $\tilde{M}$ is almost exactly M but is still a very inexpensive way of inverting this operator.

To recap, we have introduced two block diagonal preconditioners – $\mathcal{P}_{D2}$, where a solve with K is approximated by two geometric multigrid V-cycles, and $\mathcal{P}_{D3}$, which uses two V-cycles of the `HSL` algebraic multigrid routine to approximate a solve with K. In both preconditioners a solve with M is approximated by 20 steps of relaxed Jacobi accelerated by the Chebyshev semi-iteration.

Recall for the boundary control case we had the system

$$\mathcal{A}_B = \begin{bmatrix} 2\beta M_g & 0 & -E^T \\ 0 & M_u & K^T \\ -E & K & 0 \end{bmatrix},$$

where, in this case, E is not square. Using the same reasoning as above an 'ideal' preconditioner would be

$$\begin{bmatrix} 2\beta M_g & 0 & 0 \\ 0 & M_u & 0 \\ 0 & 0 & \frac{1}{2\beta} E M_g^{-1} E^T + K M_u^{-1} K^T \end{bmatrix}.$$

This can, in turn, be approximated in exactly the same way as described above, giving practical preconditioners

$$\mathcal{P}_{D2}^B = \begin{bmatrix} 2\beta \tilde{M}_g & 0 & 0 \\ 0 & \tilde{M}_u & 0 \\ 0 & 0 & \tilde{K} M_u^{-1} \tilde{K}^T \end{bmatrix} \quad \text{and} \quad \mathcal{P}_{D3}^B = \begin{bmatrix} 2\beta \tilde{M}_g & 0 & 0 \\ 0 & \tilde{M}_u & 0 \\ 0 & 0 & \widehat{K} M_u^{-1} \widehat{K}^T \end{bmatrix},$$

where, as in the distributed control case, in $\mathcal{P}_{D2}^B$ we approximate a stiffness matrix solve with two geometric multigrid V-cycles, and we use the same number of AMG V-cycles in $\mathcal{P}_{D3}^B$. In both cases we approximate a mass matrix solve using 20 Chebyshev iterations.

4 Constraint Preconditioning

As noted in Section 3, the coefficient matrix in (2.6) is of a saddle-point form. In recent years, the preconditioned projected conjugate gradient (PPCG) method [14] has become an increasingly popular method for solving saddle-point systems, particularly in

the optimization literature. The method requires the use of a preconditioner that has a very specific structure. If, as in (2.7), we write the coefficient matrix $\mathcal{A}$ of (2.6) as

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

where $B \in \mathbb{R}^{k \times l}$, then the preconditioner must take the form

$$\mathcal{P} = \begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix},$$

where $G \in \mathbb{R}^{l \times l}$ is a symmetric matrix. Let $Z \in \mathbb{R}^{l \times (l-k)}$ be such that its columns span the nullspace of B. The PPCG method can be reliably used if both $Z^T A Z$ and $Z^T G Z$ are positive definite. The basic principles behind the PPCG method are as follows. Let $W \in \mathbb{R}^{l \times k}$ be such that the columns of W together with the columns of Z span $\mathbb{R}^l$ and any solution x^* in (2.7) can be written as

$$\mathbf{x}^* = W \mathbf{x}_w^* + Z \mathbf{x}_z^*. \tag{4.1}$$

Substituting (4.1) into (2.7) and premultiplying the resulting system by $\begin{bmatrix} W^T & 0 \\ Z^T & 0 \\ 0 & I \end{bmatrix}$, we obtain the linear system

$$\begin{bmatrix} W^T A W & W^T A Z & W^T B^T \\ Z^T A W & Z^T A Z & 0 \\ B W & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_w^* \\ \mathbf{x}_z^* \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} W^T \mathbf{c} \\ Z^T \mathbf{c} \\ \mathbf{d} \end{bmatrix}.$$

Therefore, we may compute x_w^* by solving

$$B W \mathbf{x}_w^* = \mathbf{d},$$

and, having found x_w^*, we can compute x_z^* by applying the PCG method to the system

$$A_{zz} \mathbf{x}_z^* = \mathbf{c}_z,$$

where

$$\begin{aligned} A_{zz} &= Z^T A Z, \\ \mathbf{c}_z &= Z^T \left(\mathbf{c} - A W \mathbf{x}_w^* \right). \end{aligned}$$

If a preconditioner of the form $Z^T G Z$ is used and we define $\mathbf{r}_z = Z^T A Z \mathbf{x}_z - \mathbf{c}_z$ and $\mathbf{g}_z = \left(Z^T G Z \right)^{-1} \mathbf{r}_z$, then Gould *et al* [14] suggest terminating the iteration when the easily computable value $\mathbf{r}_z^T \mathbf{g}_z$ has sufficiently decreased. Note that PCG minimizes $\| \mathbf{x}_z - \mathbf{x}_z^* \|_{Z^T A Z}$ and, if $Z^T G Z$ is a good preconditioner, then

$$\| \mathbf{x}_z - \mathbf{x}_z^* \|_{Z^T A Z} = \mathbf{r}_z^T \left(Z^T A Z \right)^{-1} \mathbf{r}_z \approx \mathbf{r}_z^T \left(Z^T G Z \right)^{-1} \mathbf{r}_z = \mathbf{r}_z^T \mathbf{g}_z.$$

Gould *et al* also show that the PCG algorithm may be rewritten without the need for Z at all: this results in the PPCG algorithm, Algorithm 6.

Algorithm 6 *Choose an initial point* $\mathbf{x}$ *satisfying* $B\mathbf{x} = \mathbf{d}$ *and compute* $\mathbf{r} = A\mathbf{x} - \mathbf{c}$. *Solve* $\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{g} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ 0 \end{bmatrix}$ *and set* $\mathbf{p} = -\mathbf{g}$, $\mathbf{y} = \mathbf{v}$, $r \leftarrow \mathbf{r} - B^T\mathbf{y}$. *Repeat the following steps until a convergence test is satisfied:*

$$\begin{aligned}
\alpha &= \mathbf{r}^T\mathbf{g}/\mathbf{p}^T A\mathbf{p}, \\
\mathbf{x} &\leftarrow \mathbf{x} + \alpha\mathbf{p}, \\
\mathbf{r}^+ &= \mathbf{r} + \alpha A\mathbf{p}, \\
\textit{Solve} &\quad \begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{g}^+ \\ \mathbf{v}^+ \end{bmatrix} = \begin{bmatrix} \mathbf{r}^+ \\ 0 \end{bmatrix}, \\
\delta &= (\mathbf{r}^+)^T\mathbf{g}^+/\mathbf{r}^T\mathbf{g}, \\
\mathbf{p} &\leftarrow -\mathbf{g}^+ + \delta\mathbf{p}, \\
\mathbf{g} &\leftarrow \mathbf{g}^+, \\
\mathbf{r} &\leftarrow \mathbf{r}^+ - B^T\mathbf{v}^+.
\end{aligned}$$

If $\mathbf{y}^*$ is required, then one extra step must be carried out to compute it. However, in our case, $\mathbf{y}^*$ corresponds to the Lagrange multipliers which we are not interested in calculating.

In transforming the PCG algorithm applied to $A_{zz}\mathbf{x}_z^* = \mathbf{c}_z$ into Algorithm 6, Gould *et al* introduced the vectors $\mathbf{x}$, $\mathbf{r}$ and $\mathbf{g}$ which satisfy $\mathbf{x} = W\mathbf{x}_w^* + Z\mathbf{x}_z$, $Z^T\mathbf{r} = \mathbf{r}_z$ and $\mathbf{g} = Z\mathbf{g}_z$. Hence our measure for termination becomes $\mathbf{r}^T\mathbf{g}$, where $\mathbf{g} = Z\left(Z^TGZ\right)^{-1}Z^T\mathbf{r}$. Note the presence of a constraint preconditioner in the equivalent definition of $\mathbf{g}$ that is used in Algorithm 6:

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{g} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ 0 \end{bmatrix}.$$

The following theorem gives the main properties of the preconditioned matrix $\mathcal{P}^{-1}\mathcal{A}$: the proof can be found in [20].

Theorem 7 *Let*

$$\mathcal{A} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \quad \textit{and} \quad \mathcal{P} = \begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix},$$

where $B \in \mathbb{R}^{k\times l}$ *has full rank,* $G \in \mathbb{R}^{l\times l}$ *is symmetric and* $\mathcal{P}$ *is nonsingular. Let the columns of* $Z \in \mathbb{R}^{l\times(l-k)}$ *span the nullspace of* B, *then* $\mathcal{P}^{-1}\mathcal{A}$ *has*

- $2k$ *eigenvalues at 1; and*
- *the remaining eigenvalues satisfy the generalized eigenvalue problem*

$$Z^TAZ\mathbf{x}_z = \lambda Z^TGZ\mathbf{x}_z. \tag{4.2}$$

Additionally, if G *is nonsingular, then the eigenvalues defined by (4.2) interlace the eigenvalues of* $G^{-1}A$.

Keller *et al.* also show that the Krylov subspace

$$\mathcal{K}(\mathcal{P}^{-1}\mathcal{A};\mathbf{r}) = \text{span}(\mathbf{r}, \mathcal{P}^{-1}\mathcal{A}\mathbf{r}, (\mathcal{P}^{-1}\mathcal{A})^2\mathbf{r}, \dots)$$

will be of dimension at most $l - k + 2$, see [20].

Clearly, for our problem (2.6), A is positive definite and, hence, Z^TAZ is positive definite. It remains for us to show that we can choose a symmetric matrix G that satisfies the following properties:

- Z^TGZ is positive definite;
- the eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$ are clustered; and
- we can efficiently carry out solves with $\mathcal{P}$.

We will firstly consider setting $G = diag(A)$. Since A is a block diagonal matrix with the blocks consisting of mass matrices, G is guaranteed to be positive definite. From Theorem 7, the non-unitary eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$ will interlace the eigenvalues of $G^{-1}A$. The eigenvalues of $G^{-1}A$ satisfy

$$Mx = \lambda diag(M)x$$

and, since M is a mass matrix, the eigenvalues of $G^{-1}A$ will be bounded above and below by constant values, see [29]. Therefore, the non-unitary eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$ are bounded above and below by constant values. As we refine our mesh, the rate of convergence of the PPCG method (in exact arithmetic) will not deteriorate and, hence, this preconditioner may be described as "optimal". Unfortunately, it is not clear that we can efficiently apply this preconditioner; in Section 5, we will show that such a preconditioner may be prohibitive to use for small values of h. In the remainder of this section, we will consider a constraint preconditioner that is both efficient to apply and optimal.

It is straightforward to show that the columns of

$$Z = \begin{bmatrix} M^{-1}K \\ I \end{bmatrix}$$

span the nullspace of $\begin{bmatrix} -M & K \end{bmatrix}$ and, therefore,

$$Z^TAZ = M + 2\beta K^T M^{-1} K.$$

Suppose that we set

$$\mathcal{P}_{C1} = \left[\begin{array}{cc|c} 0 & 0 & -M \\ 0 & 2\beta K^T M^{-1} K & K^T \\ \hline -M & K & 0 \end{array}\right],$$

then, if $\mathbf{z} = \begin{bmatrix} \mathbf{z}_1^T & \mathbf{z}_2^T & \mathbf{z}_3^T \end{bmatrix}^T$ and $\mathbf{r} = \begin{bmatrix} \mathbf{r}_1^T & \mathbf{r}_2^T & \mathbf{r}_3^T \end{bmatrix}^T$, we may solve systems of the form $\mathcal{P}_{C1}\mathbf{z} = \mathbf{r}$ by carrying out the following steps:

- Solve
$$M\mathbf{z}_3 = -\mathbf{r}_1, \tag{4.3}$$
- Solve
$$2\beta K^T M^{-1} K \mathbf{z}_2 = \mathbf{r}_2 - K^T \mathbf{z}_3, \tag{4.4}$$
- Solve
$$M\mathbf{z}_1 = K\mathbf{z}_2 - \mathbf{r}_3. \tag{4.5}$$

As noted in Section 3, systems of the form (4.3) and (4.5) may be solved efficiently because M is a mass matrix. We will discuss the efficient (approximate) solution of (4.4) at the end of this section.

Proposition 8 *Let*
$$\mathcal{A} = \left[\begin{array}{cc|c} 2\beta M & 0 & -M \\ 0 & M & K^T \\ \hline -M & K & 0 \end{array}\right]$$
and
$$\mathcal{P}_{C1} = \left[\begin{array}{cc|c} 0 & 0 & -M \\ 0 & 2\beta K^T M^{-1} K & K^T \\ \hline -M & K & 0 \end{array}\right],$$
where $K, M \in \mathbb{R}^{n\times n}$. The preconditioned matrix $\mathcal{P}_{C1}^{-1}\mathcal{A}$ has

- *$2n$ eigenvalues at 1; and*
- *the remaining eigenvalues satisfy the generalized eigenvalue problem*
$$\left(\frac{1}{2\beta}\left(K^T M^{-1} K\right)^{-1} M + I\right)\mathbf{x} = \lambda\mathbf{x}. \tag{4.6}$$

Proof From Theorem 7, $\mathcal{P}_{C1}^{-1}\mathcal{A}$ has

- $2n$ eigenvalues at 1; and
- the remaining eigenvalues satisfy
$$\left(M + 2\beta K^T M^{-1} K\right)\mathbf{x} = 2\lambda\beta K^T M^{-1} K\mathbf{x}.$$

This is equivalent to the generalized eigenvalue problem (4.6). ■

We can now use this general result to give solid bounds that are dependent both on the PDE problem being considered and on the finite element discretization. In our tests, we have discretized problem (1.1) using quadrilateral $\mathbf{Q_1}$ finite elements and, for this choice, one can prove the following.

Corollary 8A *Let λ be an eigenvalue of $\mathcal{P}_{C1}^{-1}\mathcal{A}$. Then λ satisfies either*

$$\lambda = 1$$

or

$$\frac{1}{2\beta}\alpha_1 h^4 + 1 \leq \lambda \leq \frac{1}{2\beta}\alpha_2 + 1,$$

where α_1 and α_2 are positive constants independent of both h and β.

Proof From Proposition 8, $\lambda = 1$ or it satisfies the generalized eigenvalue problem

$$\left(\frac{1}{2\beta}\left(K^T M^{-1} K\right)^{-1} M + I\right) x = \lambda x.$$

From the proof of Corollary 2A we obtain the desired result. ■

Therefore, as we refine the mesh, the bounds in Corollary 8A again depend on h in a multiplicative way only, so they remain bounded away from 0 as $h \to 0$ with a bound independent of h. This suggests that this will be an optimal preconditioner for (2.6). However, as the regularization parameter β decreases, the bounds will worsen and we will expect the PPCG method to take more iterations to reach the same tolerance.

It remains for us to consider how we might solve (4.4). As in Section 3, instead of exactly carrying out solves with K, we may approximate K by a matrix $\widehat{K}$. If our approximation is good enough, then the spectral bounds will be close to those in Corollary 8A. In the case of our PDE, Poisson's equation, we will employ the same approximation as that used within the preconditioner $\mathcal{P}_{D2}$: a fixed number of multigrid V-cycles. This gives us the preconditioner

$$\mathcal{P}_{C2} = \begin{bmatrix} 0 & 0 & -\tilde{M} \\ 0 & 2\beta\widehat{K}^T M^{-1}\widehat{K} & K^T \\ -\tilde{M} & K & 0 \end{bmatrix}.$$

Here, again, $\widehat{K}$ denotes the approximation of the solves with K by two AMG V-cycles applied by using a MATLAB interface to the `HSL` package `HSL_MI20` [7], and $\tilde{M}$ denotes 20 iterations of the Chebyshev semi-iterative method. $\mathcal{P}_{C2}$ is not exactly of the form of a constraint preconditioner since $\tilde{M}$ is not exactly M. However, the bound (3.6) indicates that $\tilde{M}$ is close to M and we see no deterioration in using PPCG in any of our numerical results. Note the exact use of K and K^T in the constraint blocks: this is possible because we only require matrix-vector multiplications with these matrices.

At this point we would like to highlight some of the work of Biros and Ghattas [4] and show how it differs to the approaches proposed in this section. Two of the preconditioners that they consider take the form

$$\mathcal{P}_1 = \begin{bmatrix} 2\beta M & 0 & -M \\ 0 & 0 & K^T \\ -M & K & 0 \end{bmatrix}$$

and

$$\mathcal{P}_2 = \begin{bmatrix} 2\beta M & 0 & -M \\ 0 & 0 & \widehat{K}^T \\ -M & \widehat{K} & 0 \end{bmatrix}.$$

We note that the eigenvalues of $\mathcal{P}_1^{-1}\mathcal{A}$ are the same as the eigenvalues of $\mathcal{P}_{C1}\mathcal{A}$. The solution of systems with coefficient matrix $\mathcal{P}_1$ requires the exact solution of the forward problem $K\mathbf{z} = \mathbf{r}$ (and $K^T\mathbf{z} = \mathbf{r}$). In comparison, the solution of systems with coefficient matrix $\mathcal{P}_{C1}$ requires the exact solution of problems of the $M\mathbf{z} = \mathbf{r}$. We can solve systems of the form $\tilde{M}\mathbf{z} = \mathbf{r}$, where $\tilde{M}$ is a very good approximation to M, much more efficiently than systems of the form $\widehat{K}\mathbf{z} = \mathbf{r}$, where $\widehat{K}$ is a very good approximation to K. Hence, if we were to form a matrix $\mathcal{P}_2$ which is as efficient as to use as $\mathcal{P}_{C2}$ when carrying out the preconditioning step, then we would expect to have to resort to using a different iterative method because we will not expect the approximation $\widehat{K}$ to be close enough to K for PPCG to be a reliable method.

5 Results

We illustrate our methods with five different examples. The first four examples are of distributed control, and we compare the time to solve the system using 'backslash' in MATLAB, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$ and $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with preconditioner with $G = diag(A)$. In the latter method, we factorize the preconditioner once using MATLAB's `ldl` function and then use this factorization when carrying out the solves with the preconditioner. Example 13 is a boundary control problem, and for this we compare the time to solve the system using 'backslash' in MATLAB and MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$ and $\mathcal{P}_{\mathrm{D3}}$. We terminate MINRES when the relative residual (in the 2-norm) has reach the desired tolerance. PPCG is terminated when $r^T g$ has reached the desired tolerance relative to its initial value.

To choose the value of β, it is helpful to look at a graph of $\|u - \hat{u}\|_2$ vs $\log(\beta)$. For the system (2.6) with Dirichlet boundary conditions this is shown in Figure 2. In Example 9 we will demonstrate how our algorithm works for different β – namely we take $\beta = 10^{-2},\ 0.5 \times 10^{-4}$ and 10^{-5}.

In the tables of results that follow, the number of iterations are given in brackets after the CPU time. All tests were done using MATLAB version 7.5.0 on a machine with a dual processor AMD Opteron 244 (1.8GHz). All times are CPU times in seconds. We used the `HSL` package `HSL_MI20` [7] (via a MATLAB interface) as our AMG method. For 2D problems we use two V-cycles, two pre- and two post-smoothing steps of relaxed Jacobi with the optimal relaxation parameter of $\omega = \frac{8}{9}$ in the 2D case (see [11, Section 2.5]), and the relaxed Jacobi method is also used for the coarse solver (i.e., default `control` components of `HSL_MI20` are used apart from the following choices: coarse_solver=2, `damping=` $\frac{8}{9}$, `pre_smoothing=2`, `post_smoothing=2` and `v_iterations=2`). In 3D, we use two V-cycles, three pre- and three post-smoothing steps of unrelaxed Jacobi ($\omega = 1$ in the above), which is optimal here, and unrelaxed Jacobi is also used for the coarse

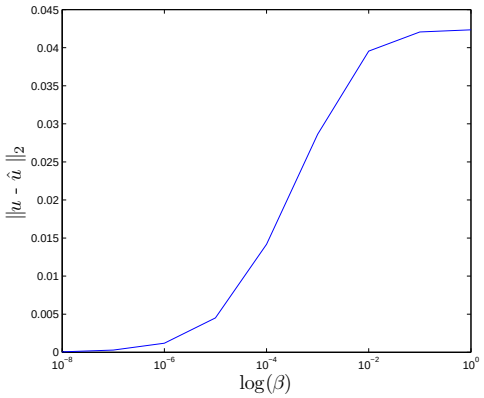


Figure 2: Graph of $\|u - \hat{u}\|_2$ vs $\log(\beta)$

solver (i.e., default `control` components of `HSL_MI20` are used apart from the following choices: coarse_solver=2, `damping= 1`, `pre_smoothing=3`, `post_smoothing=3` and `v_iterations=2`).

Example 9 *Let $\Omega = [0,1]^m$, where $m = 2, 3$, and consider the problem*

$$\min_{u,f} \frac{1}{2}||u - \hat{u}||^2_{L_2(\Omega)} + \beta||f||^2_{L_2(\Omega)}$$

$$\text{s.t.} \qquad -\nabla^2 u = f \qquad \text{in } \Omega \tag{5.1}$$

$$u = \hat{u}|_{\partial\Omega} \qquad \text{on } \partial\Omega \tag{5.2}$$

where, in 2D,

$$\hat{u} = \begin{cases} (2x-1)^2(2y-1)^2 & \text{if } (x,y) \in [0,\frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases}$$

and, in 3D,

$$\hat{u} = \begin{cases} (2x-1)^2(2y-1)^2(2z-1)^2 & \text{if } (x,y,z) \in [0,\frac{1}{2}]^3 \\ 0 & \text{otherwise} \end{cases}$$

i.e. $\hat{u}$ 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 target solution, $\hat{u}$, is given in Figure 3, and plots of the control, f, and the state, u, for the three values of β are given in Figure 5. In Tables 1 and 2, we consider the 2D version of Example 9 with $\beta = 10^{-2}$, with convergence tolerances 10^{-4} and 10^{-8}, respectively. In Tables 4 and 5 we take $\beta = 5\times10^{-5}$, and in Tables 7 and 8 we use $\beta = 10^{-5}$, and solve to the same convergence tolerances. Note that these tolerances are quite small – especially for larger h – given that the finite element method we use is only accurate to $O(h^2)$, but we want to demonstrate the mesh-independent property of

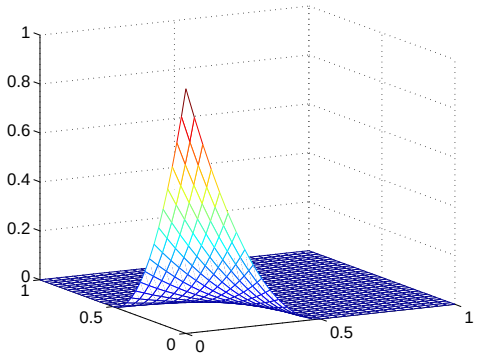


Figure 3: Graph of $\hat{u}$ for Example 9

our preconditioners. In practice one would take the accuracy of the discretization into account when deciding the accuracy with which one solves the system (2.6).

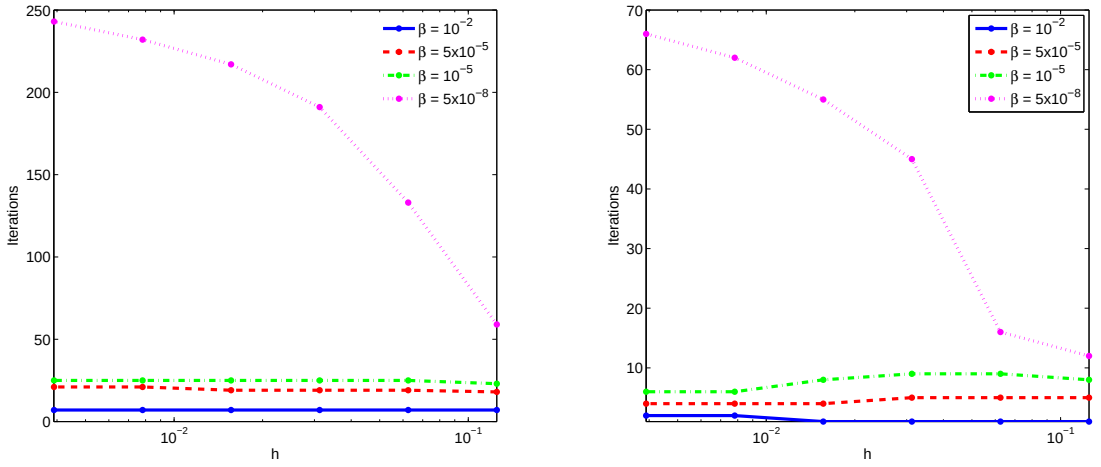
We observe that the number of iterations required by our iterative methods do not grow as we refine our mesh size with the MINRES preconditioners or when using PPCG with $\mathcal{P}_{C2}$. The smaller β we take we require more iterations to converge to the same tolerance, as expected from the theory. For $h = 2^{-9}$, MATLAB's backslash method runs out of memory, as does its `ldl` function when factorizing the constraint preconditioner with $G = diag(A)$. The iteration count for the last preconditioner is good for this example and, indeed, in all the examples considered, but its reliance on a direct method for its implementation makes it infeasible for smaller values of h. It is interesting to note that with the MINRES preconditioner, for the two smaller values of β, the number of iterations drops significantly at $h = 2^{-9}$, although we have no explanation for this behaviour.

Here the geometric multigrid preconditioner ($\mathcal{P}_{D3}$) converges in around the same number of iterations as the corresponding AMG preconditioner ($\mathcal{P}_{D2}$), but that the AMG preconditioner is significantly the faster of the two. This is because our geometric multigrid is interpreted MATLAB code, whereas the AMG solver is public domain software which is optimized `fortran` and connected to MATLAB through a MEX interface. Note also that the MINRES and PPCG methods, whether using a geometric or algebraic multigrid, are both close to linear complexity – the time taken to solve the system increases linearly with the problem size.

In Tables 3, 6 and 9, we consider the 3D version of Example 9 with convergence tolerances 10^{-4} and 10^{-8}, for $\beta = 10^{-2}, 5 \times 10^{-5}$ and 10^{-5} respectively. Again we see much the same behaviour – here MATLAB's backslash and `ldl` methods run out of memory at $h = 2^{-5}$. For this problem the geometric multigrid version of the MINRES preconditioner is the most efficient, both in terms of the number of iterations and the time taken – especially for small β. Again, the timings are close to scaling linearly with the problem size.

In Figure 4, we compare the number of iterations needed to solve the 2D problem by both the MINRES method with preconditioner $\mathcal{P}_{D3}$ (Figure 4(a)) and the PPCG method

with preconditioner $\mathcal{P}_{C2}$ (Figure 4(b)) for the tolerance 10^{-4} and different values of β. For $\beta = 10^{-2}, 5 \times 10^{-5}$ and 10^{-5}, which are the values considered above, we see the number of iterations does not grow as h decreases. For $\beta = 5 \times 10^{-8}$ we see a big increase in the number of iterations needed to converge for both methods, however even this seems to tend to a constant value – although one which is probably too high for the algorithm to be practical. For fixed values of h, decreasing β results in an increasing number of iterations.



(a) MINRES (b) PPCG

Figure 4: Number of iterations for MINRES with $\mathcal{P}_{D3}$ and PPCG with $\mathcal{P}_{C2}$ to solve Example 9 in 2D for $\beta = 10^{-2},\ 5 \times 10^{-5},\ 10^{-5},$ and 5×10^{-8}

Example 10 *Again, let* $\Omega = [0,1]^m$*, where* $m = 2, 3$*, and consider the problem*

$$\min_{u,f} \frac{1}{2}||u - \hat{u}||^2_{L_2(\Omega)} + \beta||f||^2_{L_2(\Omega)}$$

$$\text{s.t.} \qquad -\nabla^2 u = f \qquad \text{in } \Omega \tag{5.3}$$

$$u = 0 \qquad \text{on } \partial\Omega \tag{5.4}$$

where, in 2D,

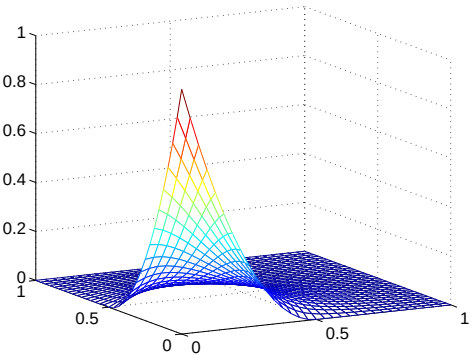
$$\hat{u} = \exp\left(-64((x-0.5)^2 + (y-0.5)^2)\right),$$

and in 3D,

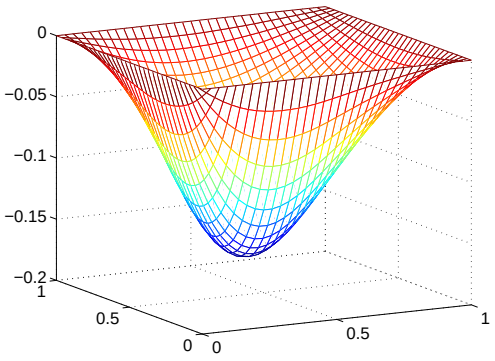
$$\hat{u} = \exp\left(-64((x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2)\right),$$

i.e. $\hat{u}$ *is a Gaussian, with peak of unit height at* $[\frac{1}{2}, \frac{1}{2}]$.

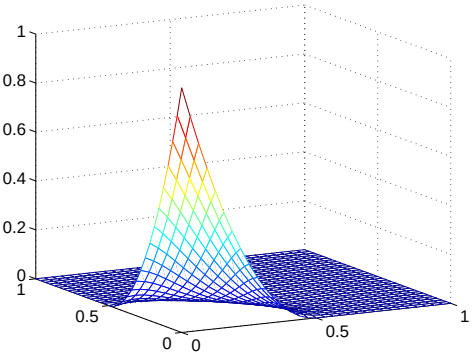
For this and the following two examples, we take $\beta = 5 \times 10^{-5}$. A plot of $\hat{u}$ is given in Figure 6. Tables 10 and 11 show our findings for Example 10 which is another Dirichlet control problem – this time with a different cost functional. Here we see similar behaviour to that in Example 9, which demonstrates that our method appears not to depend on $\hat{u}$.



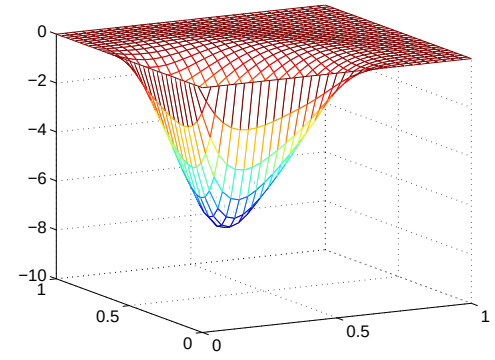
(a) $u,\ \beta = 10^{-2}$



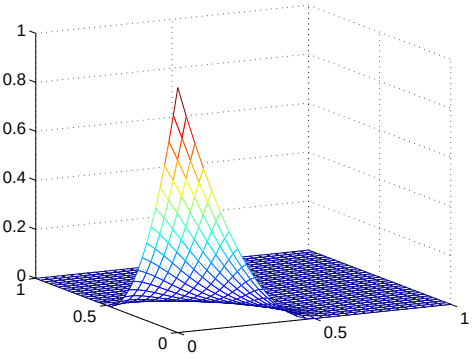
(b) $f,\ \beta = 10^{-2}$



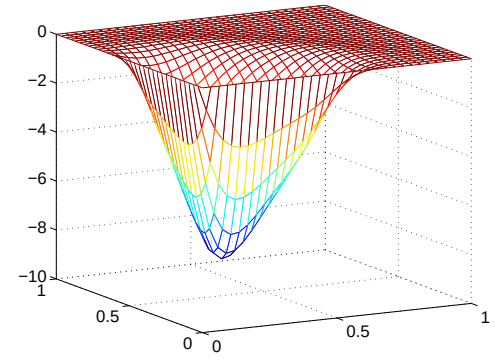
(c) $u,\ \beta = 0.5 \times 10^{-4}$



(d) $f,\ \beta = 0.5 \times 10^{-4}$



(e) $u,\ \beta = 10^{-5}$



(f) $f,\ \beta = 10^{-5}$

Figure 5: Plots of the state, u, and the control, f for $\beta = 10^{-2},\ 0.5 \times 10^{-4}$ and 10^{-5}

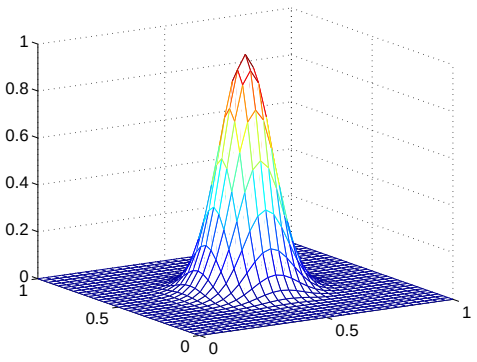


Figure 6: Graph of $\hat{u}$ for Example 10

Example 11 *Let* $\Omega = [0,1]^2$ *and consider the Neumann problem*

$$\min_{u,f} \frac{1}{2}||u - \hat{u}||^2_{L_2(\Omega)} + \beta||f||^2_{L_2(\Omega)}$$

$$\text{s.t.} \qquad -\nabla^2 u = f \qquad \text{in } \Omega \tag{5.5}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega \tag{5.6}$$

where

$$\hat{u} = \begin{cases} (2x-1)^2(2y-1)^2 & \text{if } (x,y) \in [0,\frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases} .$$

This is a distributed control problem with a Neumann boundary condition, therefore the stiffness matrix K is singular. We comment that for simple forward solution of the Neumann problem, a singular multigrid cycle is possible [16, Chapter 12], whereas in the control problem we require a definite preconditioner, hence a singular multigrid cycle is not usable. This is, however, not a difficulty, as we simply pin one of the nodes to remove the singularity; this gives us, in effect, a mixed boundary condition problem with a Dirichlet boundary condition at just one point. In this case we have made u vanish at $(1,1)$ (which is consistent with the objective function). Tables 12 and 13 shows our results in this case. In comparison with the Dirichlet results our methods are slightly less effective here – the iteration count and, correspondingly, the time taken are both larger. However, the overall behaviour is similar - we still get mesh size independent convergence and nearly linear complexity.

Example 12 *Let* $\Omega = [0,1]^2$ *and consider the problem*

$$\min_{u,f} \frac{1}{2}||u - \hat{u}||^2_{L_2(\Omega)} + \beta||f||^2_{L_2(\Omega)}$$

$$\text{s.t.} \qquad -\nabla^2 u = f \qquad \text{in } \Omega \tag{5.7}$$

$$u = \hat{u}|_{\partial\Omega} \text{ on } \partial\Omega_1 \qquad \text{and} \qquad \frac{\partial u}{\partial n} = 0 \text{ on } \partial\Omega_2 \tag{5.8}$$

where

$$\hat{u} = \begin{cases} (2x-1)^2(2y-1)^2 & \text{if } (x,y) \in [0,\frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases}$$

and $\partial\Omega_1 = (0 \times [0,1)) \cup ((0,1] \times 0)$ *and* $\partial\Omega_2 = (1 \times (0,1]) \cup ([0,1) \times 1)$.

Example 12 is a distributed control problem with mixed boundary conditions: half of the boundary satisfies a Dirichlet boundary condition while the other half satisfies a Neumann boundary condition. As we might expect from the nature of the problem, the results in Tables 14 and 15 lie somewhere in between those of Examples 9 and 10 and those of Example 11. The solution timings remain optimal.

Example 13 *Let* $\Omega = [0,1]^2$ *and consider the boundary control problem*

$$\min_{u,f} \frac{1}{2}||u - \hat{u}||^2_{L_2(\Omega)} + \beta||g||^2_{L_2(\partial\Omega)}$$

$$\text{s.t.} \qquad -\nabla^2 u = 0 \text{ in } \Omega \tag{5.9}$$

$$\frac{\partial u}{\partial n} = g \text{ on } \partial\Omega \tag{5.10}$$

where

$$\hat{u} = \begin{cases} (2x-1)^2(2y-1)^2 & \text{if } (x,y) \in [0,\frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases} .$$

This is example of boundary control with a Neumann boundary condition. In this case the (1,3) block in 2.6 is not square, so the constraint preconditioners as presented here cannot be applied without further modification. The block diagonal preconditioners $\mathcal{P}^B_{D2}$ and $\mathcal{P}^B_{D3}$ can be applied, and we give the results for solving this problem (using a value of $\beta = 10^{-2}$) in Table 16. Here, again we see we have mesh size independent convergence and optimal solution timings. We note that the iteration numbers for the preconditioner using algebraic multigrid $\mathcal{P}_{D3}$ are not as uniform as when we use a geometric multigrid routine $\mathcal{P}_{D2}$, although we still get mesh-independent convergence. It is possible that a different choice of parameters would alleviate this effect, however our philosophy here is to use AMG essentially as a 'black box' method and we present the results with a minimal adjustment of the default parameters.

6 Conclusion

We have presented optimal preconditioners for distributed and boundary control problems. These preconditioners are conceived of in block form and include a small fixed

number of multigrid cycles for particular blocks. The preconditioners are employed with appropriate Krylov subspace methods for symmetric linear systems, for which we have reliable convergence bounds. We have demonstrated that our preconditioners work effectively with regularization parameter 5×10^{-5} (and also $\beta = 10^{-2}$ and 10^{-5} for our first example problem) although the approximations become less valid as β approaches zero they still give mesh size independent convergence down to about $\beta = 10^{-6}$, which is comparable with the discretization error in our PDEs for the smallest grid size employed here.

Only a handful of papers in the literature consider the saddle-point structure of the matrices when solving problems of this type: we have used this structure to create efficient algorithms. A large amount of work has been done on solving saddle point systems – e.g. see the survey paper by Benzi, Golub and Liesen [3] – and on block preconditioners [11]. The block preconditioning approach allows the incorporation of powerful techniques such as multigrid.

Two classes of preconditioners for the simplest case of PDE, namely the Poisson equation, have been proposed, but we believe our methods are more general than that. The availability of a good preconditioner for a forward and adjoint PDE operator should allow reasonably straightforward generalizations of the block diagonal preconditioning idea; in further research on incompressible flow control this does appear to be the case. It is less obvious that the constraint preconditioning ideas presented here are so easily generalizable to other PDE problems, though we are having success with related approaches for incompressible flow control. This work is ongoing and the results will be presented in a subsequent paper.

Our numerical results are for idealized examples – in practice there are a number of modifications which lead to systems with similar algebraic structure. For example, one could consider any Hilbert space norms in place of the $L_2(\Omega)$ norms in (1.1) and the (1,1) and (2,2) blocks in (2.6) would be the corresponding Gram matrices: our techniques with appropriate modification should handle this.

In some cases the blocks in the system (2.8) are not all square. We have included a boundary control example to demonstrate how the block diagonal preconditioning method can be applied in such cases. The constraint preconditioning methods are not immediately applicable for this class of problem but we may be able to apply related approaches that are similar to those we are working on for incompressible flow control.

One may only want to control u on part of Ω, or have a few point measurements to match – in this case the (2,2) block in (2.6) is singular. It is not clear at this stage how our block diagonal preconditioning method could be applied but there is scope for the constraint preconditioning method because these do not assume that A in (2.7) is non-singular.

Another common generalization which is relevant in many practical problems would be the inclusion of bound constraints. Many methods for handling these lead to systems of a similar algebraic structure to those we consider here (for example, see [12]): we foresee that our methods should also be applicable for such problems.

Acknowledgement We thank Eldad Haber and an anonymous referee for their careful reading and insightful comments which have greatly improved the content of this paper.

References

[1] Uri M. Asher and Eldad Haber. A multigrid method for distributed parameter estimation problems. *Electronic Transactions in Numerical Analysis*, 15:1–17, 2003.

[2] Owe Axelsson and Maya Neytcheva. Eigenvalue estimates for preconditioned saddle point matrices. *Numer. Linear Algebra Appl.*, 13:339–360, 2006.

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

[4] G Biros and O Ghattas. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: The Krylov-Schur solver. *SIAM Journal on Scientific Computing*, 27, 2000.

[5] George Biros and Günay Dogan. A multilevel algorithm for inverse problems with elliptic PDE constraints. *Inverse Problems*, 24(3):034010 (18pp), 2008.

[6] A. Borzì and V. Schulz. Multigrid methods for PDE optimization. *To appear in SIAM Review*.

[7] 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.

[8] A Brandt, S.C. McCormick, and J. Ruge. *Algebraic multigrid (AMG) for sparse matrix equations*, pages 257–284. Cambridge University Press, 1985.

[9] William L. Briggs, Van Emden Henson, and S.F. McCormick. *A Multigrid Tutorial*. SIAM, second edition, 2000.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] E. Haber and U. Ascher. Preconditioned all–at–once methods for large sparse parameter estimation problems, 2000.

[16] W. Hackbush. *Multi-Grid methods and applications*. Springer-Verlag, 1985.

[17] Heinkenschloss and H. Nguyen. Neumann-Neumann domain decomposition preconditioners for linear-quadratic elliptic optimal control problems. *SIAM Journal on Scientific Computing*, 28:1001–1028, 2006.

[18] 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 Journal on Scientific Computing*, 30(2):1038–1063, 2008.

[19] 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.

[20] Carsten Keller, Nicholas I. M. Gould, and Andrew J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300–1317, 2000.

[21] J. L. Lions. *Optimal Control of Systems*. Springer, 1968.

[22] Helmut Maurer and Hans D. Mittelmann. Optimization techniques for solving elliptic control problems with control and state constraints. Part 1: Boundary control. *Comput. Optim. Appl.*, 16(2):29–55, 2000.

[23] Helmut Maurer and Hans D. Mittelmann. Optimization techniques for solving elliptic control problems with control and state constraints. Part 2: Distributed control. *Comput. Optim. Appl.*, 18(2):141–160, 2001.

[24] Malcolm F. Murphy, Gene H. Golub, and Andrew J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972, 2000.

[25] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[26] Joachim Schöberl and Walter Zulehner. Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems. *SIAM Journal on Matrix Analysis and Applications*, 29(3):752–773, 2007.

[27] U Trottenberg, C Oosterlee, and A Schüller. *Multigrid*. Academic Press, 2000.

[28] R.S. Varga. *Matrix Iterative Analysis*. Prentice Hall, 1962.

[29] A. J. Wathen. Realistic eigenvalue bounds for the Galerkin mass matrix. *IMA J Numer Anal*, 7(4):449–457, 1987.

Table 1: Comparison of times and iterations to solve Example 9 in 2D with $\beta = 10^{-2}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	27	0.0002	0.13 (7)	0.023 (7)	0.008 (2)	0.003 (5)
2^{-3}	147	0.002	0.15 (7)	0.028 (7)	0.009 (1)	0.013 (5)
2^{-4}	675	0.009	0.19 (7)	0.044 (7)	0.015 (1)	0.087 (5)
2^{-5}	2883	0.062	0.34 (7)	0.18 (7)	0.041 (1)	0.50 (4)
2^{-6}	11907	0.37	1.1 (7)	0.51 (7)	0.18 (1)	3.56 (4)
2^{-7}	48387	2.22	4.1 (7)	2.1 (7)	1.11 (2)	24.2 (4)
2^{-8}	195075	15.7	18.9 (7)	10.3 (7)	5.25 (2)	136 (4)
2^{-9}	783363	—	92.1 (7)	64.2 (9)	26.3 (2)	—

Table 2: Comparison of times and iterations to solve Example 9 in 2D with $\beta = 10^{-2}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	27	0.0002	0.15 (10)	0.032 (10)	0.016 (3)	0.004 (6)
2^{-3}	147	0.002	0.17 (10)	0.038 (10)	0.018 (3)	0.022 (11)
2^{-4}	675	0.009	0.26 (12)	0.071 (12)	0.029 (3)	0.16 (11)
2^{-5}	2883	0.062	0.50 (12)	0.18 (12)	0.075 (3)	1.08 (11)
2^{-6}	11907	0.37	1.67 (12)	0.80 (12)	0.34 (3)	6.99 (10)
2^{-7}	48387	2.22	6.60 (12)	3.95 (14)	1.44 (3)	46.1 (10)
2^{-8}	195075	15.7	30.9 (12)	19.0 (14)	6.77 (3)	247 (10)
2^{-9}	783363	—	134 (11)	88.9 (13)	41.6 (4)	—

Table 3: Comparison of times and iterations to solve Example 9 in 3D with $\beta = 10^{-2}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} and 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
			Convergence tolerance = 10^{-4}			
2^{-2}	81	0.001	0.13 (5)	0.034 (5)	0.01 (1)	0.01 (6)
2^{-3}	1029	0.013	0.20 (5)	0.085 (5)	0.04 (1)	0.39 (6)
2^{-4}	10125	25.5	1.16 (5)	1.34 (5)	0.68 (1)	17.4 (5)
2^{-5}	89373	—	13.9 (7)	15.7 (5)	8.11 (1)	—
			Convergence tolerance = 10^{-8}			
2^{-2}	81	0.001	0.14 (8)	0.031 (8)	0.02 (3)	0.01 (6)
2^{-3}	1029	0.13	0.28 (10)	0.14 (10)	0.07 (3)	0.75 (6)
2^{-4}	10125	25.5	2.04 (10)	2.30 (10)	1.12 (3)	34.4 (5)
2^{-5}	89373	—	19.2 (10)	26.7 (10)	13.1 (3)	—

Table 4: Comparison of times and iterations to solve Example 9 in 2D with $\beta = 5 \times 10^{-5}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	27	0.0002	0.16 (13)	0.041 (13)	0.019 (5)	0.003 (5)
2^{-3}	147	0.002	0.23 (18)	0.065 (18)	0.023 (5)	0.008 (5)
2^{-4}	675	0.009	0.34 (19)	0.11 (19)	0.038 (5)	0.083 (5)
2^{-5}	2883	0.063	0.73 (19)	0.28 (19)	0.11 (5)	0.52 (5)
2^{-6}	11907	0.43	2.70 (20)	1.28 (20)	0.41 (4)	3.55 (4)
2^{-7}	48387	2.45	11.1 (21)	5.69 (21)	1.74 (4)	22.4 (4)
2^{-8}	195075	14.4	52.2 (21)	27.9 (21)	8.19 (4)	117 (4)
2^{-9}	783363	—	155 (13)	90.4 (13)	48.4 (5)	—

Table 5: Comparison of times and iterations to solve Example 9 in 2D with $\beta = 5 \times 10^{-5}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	27	0.0002	0.17 (16)	0.044 (14)	0.028 (7)	0.003 (6)
2^{-3}	147	0.002	0.33 (30)	0.10 (30)	0.039 (9)	0.013 (11)
2^{-4}	675	0.009	0.51 (32)	0.17 (32)	0.063 (9)	0.15 (11)
2^{-5}	2883	0.063	1.21 (34)	0.48 (34)	0.15 (8)	1.08 (11)
2^{-6}	11907	0.43	4.40 (34)	2.12 (34)	0.71 (8)	6.93 (10)
2^{-7}	48387	2.45	17.6 (34)	9.03 (34)	3.03 (8)	43.4 (10)
2^{-8}	195075	14.4	88.0 (36)	45.3 (34)	14.2 (8)	224 (10)
2^{-9}	783363	—	282 (24)	156 (24)	79.6 (9)	—

Table 6: Comparison of times and iterations to solve Example 9 in 3D with $\beta = 5 \times 10^{-5}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} and 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
			Convergence tolerance = 10^{-4}			
2^{-2}	81	0.001	0.15 (10)	0.15 (11)	0.02 (5)	0.01 (6)
2^{-3}	1029	0.13	0.34 (14)	0.48 (23)	0.10 (5)	0.18 (6)
2^{-4}	10125	18.9	2.74 (14)	4.36 (23)	1.58 (5)	18.8 (5)
2^{-5}	89373	—	26.5 (14)	44.1 (24)	15.7 (4)	—
			Convergence tolerance = 10^{-8}			
2^{-2}	81	0.001	0.053 (12)	0.060 (17)	0.03 (7)	0.01 (10)
2^{-3}	1029	0.13	0.25 (18)	0.45 (31)	0.15 (8)	0.34 (14)
2^{-4}	10125	18.9	3.76 (18)	7.29 (37)	2.21 (8)	36.4 (13)
2^{-5}	89373	—	44.0 (18)	85.3 (37)	25.6 (8)	—

Table 7: Comparison of times and iterations to solve Example 9 in 2D with $\beta = 10^{-5}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	27	0.0002	0.16 (13)	0.041 (13)	0.027 (7)	0.003 (5)
2^{-3}	147	0.002	0.26 (23)	0.082 (23)	0.033 (8)	0.008 (5)
2^{-4}	675	0.009	0.42 (25)	0.14 (25)	0.061 (9)	0.040 (5)
2^{-5}	2883	0.072	0.91 (25)	0.36 (25)	0.17 (9)	0.46 (4)
2^{-6}	11907	0.42	3.29 (25)	1.59 (25)	0.70 (8)	3.56 (4)
2^{-7}	48387	2.53	13.1 (25)	6.74 (25)	2.40 (6)	22.4 (4)
2^{-8}	195075	14.1	61.7 (25)	33.1 (25)	11.1 (6)	117 (4)
2^{-9}	783363	—	201 (17)	114 (17)	70.8 (8)	—

Table 8: Comparison of times and iterations to solve Example 9 in 2D $\beta = 10^{-5}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	27	0.0002	0.17 (16)	0.050 (16)	0.027 (7)	0.003 (6)
2^{-3}	147	0.002	0.35 (35)	0.12 (36)	0.052 (13)	0.008 (11)
2^{-4}	675	0.009	0.61 (40)	0.22 (40)	0.092 (14)	0.083 (11)
2^{-5}	2883	0.072	1.39 (40)	0.57 (40)	0.24 (14)	0.52 (11)
2^{-6}	11907	0.42	5.11 (40)	2.4 (40)	1.08 (13)	3.55 (10)
2^{-7}	48387	2.53	21.06 (41)	10.8 (41)	4.56 (13)	22.4 (10)
2^{-8}	195075	14.1	102 (42)	53.8 (42)	21.6 (13)	117 (10)
2^{-9}	783363	—	303 (26)	167 (26)	108 (13)	—

Table 9: Comparison of times and iterations to solve Example 9 in 3D $\beta = 10^{-5}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} and 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
			Convergence tolerance = 10^{-4}			
2^{-2}	81	0.001	0.16 (10)	0.053 (12)	0.03 (7)	0.01 (6)
2^{-3}	1029	0.13	0.38 (16)	0.35 (25)	0.14 (8)	0.18 (6)
2^{-4}	10125	18.5	3.12 (16)	4.86 (24)	2.67 (10)	13.0 (5)
2^{-5}	89373	—	29.6 (16)	61.3 (26)	25.5 (8)	—
			Convergence tolerance = 10^{-8}			
2^{-2}	81	0.001	0.16 (11)	0.071 (20)	0.05 (12)	0.01 (6)
2^{-3}	1029	0.13	0.55 (27)	0.51 (39)	0.25 (16)	0.34 (6)
2^{-4}	10125	18.5	5.17 (28)	9.07 (46)	4.14 (17)	26.7 (5)
2^{-5}	89373	—	52.4 (29)	106 (47)	45.6 (16)	—

Table 10: Comparison of times and iterations to solve Example 10 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D2}}$, $\mathcal{P}_{\mathrm{D3}}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	27	0.0003	0.13 (7)	0.024 (7)	0.01 (2)	0.003 (3)
2^{-3}	147	0.002	0.19 (13)	0.047 (13)	0.02 (3)	0.006 (3)
2^{-4}	675	0.009	0.27 (13)	0.076 (13)	0.03 (3)	0.048 (2)
2^{-5}	2883	0.07	0.54 (13)	0.19 (13)	0.07 (3)	0.34 (2)
2^{-6}	11907	0.43	1.80 (13)	0.86 (13)	0.35 (3)	1.83 (1)
2^{-7}	48387	2.36	7.15 (13)	3.68 (13)	1.47 (3)	11.9 (1)
2^{-8}	195075	14.3	33.3 (13)	17.8 (13)	8.26 (4)	117 (4)
2^{-9}	783363	—	153 (13)	92.8 (13)	33.8 (3)	—

Table 11: Comparison of times and iterations to solve Example 10 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\rm D2}$, $\mathcal{P}_{\rm D3}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	27	0.0003	0.13 (8)	0.027 (8)	0.02 (3)	0.003 (3)
2^{-3}	147	0.002	0.24 (20)	0.071 (20)	0.02 (4)	0.01 (7)
2^{-4}	675	0.009	0.35 (20)	0.11 (20)	0.04 (4)	0.07 (4)
2^{-5}	2883	0.07	0.83 (22)	0.31 (22)	0.12 (6)	0.42 (3)
2^{-6}	11907	0.43	2.88 (22)	1.51 (24)	0.57 (6)	2.40 (2)
2^{-7}	48387	2.36	12.5 (24)	6.54 (24)	2.38 (6)	15.3 (2)
2^{-8}	195075	14.3	59.5 (24)	31.5 (24)	14.1 (8)	224 (10)
2^{-9}	783363	—	280 (24)	158 (24)	55.6 (6)	—

Table 12: Comparison of times and iterations to solve Example 11 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\rm D2}$, $\mathcal{P}_{\rm D3}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	72	0.0008	0.28 (29)	0.093 (29)	0.02 (4)	0.005 (4)
2^{-3}	240	0.003	0.39 (35)	0.13 (35)	0.02 (4)	0.008 (2)
2^{-4}	864	0.01	0.62 (35)	0.22 (35)	0.04 (4)	0.07 (2)
2^{-5}	3264	0.08	1.50 (37)	0.58 (37)	0.10 (4)	0.33 (1)
2^{-6}	12672	0.58	5.08 (37)	0.31 (47)	0.36 (3)	2.03 (1)
2^{-7}	49920	3.77	22.7 (39)	12.8 (43)	1.61 (3)	11.5 (1)
2^{-8}	198144	27.5	100 (41)	57.7 (45)	6.82 (3)	65.4 (1)
2^{-9}	789504	—	421 (43)	230 (45)	27.3 (3)	—

Table 13: Comparison of times and iterations to solve Example 11 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\rm D2}$, $\mathcal{P}_{\rm D3}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	72	0.0008	0.36 (42)	0.13 (42)	0.04 (11)	0.007 (9)
2^{-3}	240	0.003	0.61 (60)	0.22 (58)	0.05 (10)	0.02 (7)
2^{-4}	864	0.01	1.05 (64)	0.39 (64)	0.08 (11)	0.10 (4)
2^{-5}	3264	0.08	2.48 (64)	1.07 (68)	0.25 (12)	0.54 (3)
2^{-6}	12672	0.58	8.67 (64)	5.22 (78)	0.92 (10)	2.68 (2)
2^{-7}	49920	3.77	37.4 (65)	22.2 (76)	4.43 (11)	14.9 (2)
2^{-8}	198144	27.5	201 (83)	103 (82)	18.7 (11)	65.4 (1)
2^{-9}	789504	—	813 (84)	433 (86)	74.7 (11)	—

Table 14: Comparison of times and iterations to solve Example 12 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\rm D2}$, $\mathcal{P}_{\rm D3}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	48	0.0005	0.19 (19)	0.06 (19)	0.03 (7)	0.005 (6)
2^{-3}	192	0.002	0.27 (23)	0.087 (23)	0.04 (8)	0.01 (5)
2^{-4}	768	0.01	0.44 (25)	0.15 (25)	0.06 (8)	0.10 (5)
2^{-5}	3072	0.07	0.98 (25)	0.38 (25)	0.15 (7)	0.57 (4)
2^{-6}	12288	0.40	3.64 (27)	1.65 (25)	0.51 (5)	3.73 (4)
2^{-7}	49152	2.43	15.7 (27)	8.19 (27)	2.39 (5)	22.2 (4)
2^{-8}	196608	14.8	66.5 (27)	37.5 (29)	9.68 (5)	120 (4)
2^{-9}	786432	—	278 (28)	151 (29)	51.4 (7)	—

Table 15: Comparison of times and iterations to solve Example 12 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-8} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\rm D2}$, $\mathcal{P}_{\rm D3}$, PPCG with preconditioner $\mathcal{P}_{C2}$, and PPCG with constraint preconditioner containing $G = diag(A)$.

h	3n	backslash	MINRES ($\mathcal{P}_{D2}$)	MINRES ($\mathcal{P}_{D3}$)	PPCG ($\mathcal{P}_{C2}$)	PPCG ($G = diag(A)$)
2^{-2}	48	0.0005	0.24 (28)	0.088 (28)	0.027 (7)	0.003 (6)
2^{-3}	192	0.002	0.39 (38)	0.14 (40)	0.052 (13)	0.01 (11)
2^{-4}	768	0.01	0.70 (44)	0.27 (46)	0.092 (14)	0.07 (11)
2^{-5}	3072	0.07	1.65 (44)	0.68 (46)	0.24 (14)	0.99 (11)
2^{-6}	12288	0.40	5.75 (44)	2.95 (46)	1.08 (13)	7.01 (10)
2^{-7}	49152	2.43	25.2 (44)	13.6 (46)	4.56 (13)	43.9 (10)
2^{-8}	196608	14.8	106 (44)	58.6 (46)	21.6 (13)	224 (10)
2^{-9}	786432	—	432 (44)	246 (48)	108 (13)	—

Table 16: Comparison of times and iterations to solve Example 13 in 2D $\beta = 10^{-2}$ for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-4} and 10^{-8} for MATLAB's backslash method and MINRES with preconditioners $\mathcal{P}^B_{\rm D2}$, $\mathcal{P}^B_{\rm D3}$.

h	size(A)	backslash	MINRES ($\mathcal{P}^B_{D2}$)	MINRES ($\mathcal{P}^B_{D3}$)	MINRES ($\mathcal{P}^B_{D2}$)	MINRES ($\mathcal{P}^B_{D3}$)
			Conv. tolerance = 10^{-4}		Conv. tolerance = 10^{-8}	
2^{-2}	66	0.0006	0.13 (15)	0.044 (7)	0.20 (28)	0.032 (8)
2^{-3}	194	0.005	0.15 (15)	0.034 (7)	0.26 (26)	0.038 (8)
2^{-4}	642	0.011	0.24 (13)	0.085 (13)	0.42 (26)	0.09 (14)
2^{-5}	2306	0.078	0.48 (13)	0.18 (13)	0.96 (26)	0.20 (14)
2^{-6}	8706	0.70	1.62 (13)	1.12 (21)	2.99 (24)	1.26 (23)
2^{-7}	33794	6.73	6.54 (13)	6.99 (31)	11.0 (22)	7.32 (32)
2^{-8}	133122	69.1	27.2 (13)	13.7 (13)	45.8 (22)	14.9 (14)
2^{-9}	528386	—	109 (13)	103 (27)	183 (22)	109 (28)