Science & Technology Facilities Council

Technical Report

Optimal solvers for PDE-constrained optimization

Tyrone Rees H. Sue Dollar Andrew J. Wathen

June 18, 2008

RAL-TR-2008-018

Library and Information Services
SFTC Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Didcot
OX11 0QX
UK
Tel: +44 (0)1235 445384
Fax: +44(0)1235 446403
Email: library@rl.ac.uk

ISSN 1358-6254

RAL-TR-2008-018

Optimal solvers for PDE-constrained optimization

Tyrone Rees[1], H. Sue Dollar[2], and Andrew J. Wathen[3]

ABSTRACT

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.

Keywords: saddle-point problems, PDE-constrained optimization, preconditioning, optimal control, linear systems, all-at-once methods.

AMS(MOS) subject classifications: 49M25, 49K20, 65F10, 65N22, 65F50, 65N55.

---

[1] Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, England, UK.
Email: tyrone.rees@comlab.ox.ac.uk
This work was supported by an EPSRC Doctoral Training Award.

[2] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, UK.
Email: s.dollar@rl.ac.uk
Current reports available from `http://www.numerical.rl.ac.uk/reports/reports.html`
This work was supported by EPSRC grant EP/E053351/1.

[3] Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, England, UK.
Email: andy.wathen@comlab.ox.ac.uk

June 18, 2008

1 Introduction

In this paper, we consider the distributed control problem which consists of a cost functional (1) to be minimized subject to a partial differential equation problem posed on a domain $\Omega \subset \mathbb{R}^2$ or $\mathbb{R}^3$:

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

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

$$\text{with} \quad u = g \text{ on } \partial\Omega_1 \qquad \text{and} \qquad \frac{\partial u}{\partial n} = g \text{ on } \partial\Omega_2, \tag{3}$$

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 [17]. 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) is added because, in general, the problem would be ill-posed, and so needs this Tikhonov regularization term. The Tikhonov parameter β in general needs to be determined, although it is often selected a priori—a value around $\beta = 10^{-2}$ is commonly used (see [6],[11],[15]).

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. Here for the Laplacian we employ standard multigrid cycles with both geometric and algebraic multigrid procedures. 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. It is also the case that we consider here only distributed control problems: boundary control problems are also important and are the subject of some of our ongoing research.

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 [6] 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 [6]). We are then able to use symmetric iterative methods—in particular we use MINRES ([21]) and a projected Conjugate Gradient (PPCG) method ([10]) —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., [7]). 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 ([22]) 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. Other solution methods employing multigrid for this and similar classes of problems are described by Biros and Dogan([3]), Engel and Griebel([8]), and Borzi([4]). Domain Decomposition and Model Order Reduction ideas are also successfully applied in this context: see for example Heinkenschloss and Nguyen ([13]) and Heinkenschloss, Sorensen and Sun([14]).

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 four 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 (2) and (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 [7]). 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

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega vf \qquad \forall v \in H^1_0(\Omega). \tag{4}$$

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 (4): 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). 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

$$\min_{u_h, f_h} \frac{1}{2}||u_h - \hat{u}||^2_2 + \beta||f_h||^2_2 \tag{5}$$

$$\text{such that} \qquad \int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega v_h f \qquad \forall v_h \in V^h_0. \tag{6}$$

We can write the discrete cost functional as

$$\min_{u_h, f_h} \frac{1}{2}||u_h - \hat{u}||^2_2 + \beta||f_h||^2_2 = \min_{\mathbf{u},\mathbf{f}} \frac{1}{2}\mathbf{u}^T M\mathbf{u} - \mathbf{u}^T\mathbf{b} + \alpha + \beta\mathbf{f}^T M\mathbf{f}, \tag{7}$$

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: (6) 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

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

or

$$K\mathbf{u} = M\mathbf{f} + \mathbf{d}, \tag{8}$$

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 (7) and (8) together are equivalent to (5) and (6).

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

$$\begin{bmatrix} 2\beta M & 0 & -M \\ 0 & M & K^T \\ -M & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{9}$$

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

$$\begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}, \tag{10}$$

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 [21] 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* [16] which we may use in conjunction with the projected conjugate gradient (PPCG) method [10]. The crucial step to ensure that acceptably rapid convergence is guaranteed is preconditioning: we consider in the next two sections our two preconditioning approaches.

3 Block Diagonally Preconditioned MINRES

In general, the system (9) will be symmetric but indefinite, so we may use the MINRES algorithm to solve the system: this is a Krylov subspace method for symmetric linear systems. MINRES has to be coupled with a preconditioner to get satifactory 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 [20] illustrates this idea:

Theorem 3.1. *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 (9) 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 [24]. 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

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

The following result, which is an application and extension of a result in [1], tells us about the clustering of the eigenvalues using this preconditioner:

Proposition 3.2. *Let λ be an eigenvalue of $\mathcal{P}_{\text{D1}}^{-1}\mathcal{A}$. Then either $\lambda = 1$, $\frac{1}{2}(1 + \sqrt{1 + 4\sigma_1}) \le \lambda \le \frac{1}{2}(1 + \sqrt{1 + 4\sigma_m})$ or $\frac{1}{2}(1 - \sqrt{1 + 4\sigma_m}) \le \lambda \le \frac{1}{2}(1 - \sqrt{1 + 4\sigma_1})$, where $0 \le \sigma_1 \le \cdots \le \sigma_m$ are the eigenvalues of $\frac{1}{2\beta}(KM^{-1}K^T)^{-1}M + I$.*

Proof. First note that the eigenvalues of $\mathcal{P}_{\text{D1}}^{-1}\mathcal{A}$ are identical to the eigenvalues of $\tilde{\mathcal{A}} := \mathcal{P}_{\text{D1}}^{-\frac{1}{2}}\mathcal{A}\mathcal{P}_{\text{D1}}^{-\frac{1}{2}}$, 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

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

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

$$\begin{aligned} BB^T &= \begin{bmatrix} -\frac{1}{\sqrt{2\beta}}(KM^{-1}K^T)^{-\frac{1}{2}} M^{\frac{1}{2}} & (KM^{-1}K^T)^{-\frac{1}{2}} KM^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2\beta}} M^{\frac{1}{2}} (KM^{-1}K^T)^{-\frac{1}{2}} \\ M^{-\frac{1}{2}} K^T (KM^{-1}K^T)^{-\frac{1}{2}} \end{bmatrix} \\ &= \frac{1}{2\beta}(KM^{-1}K^T)^{-\frac{1}{2}} M (KM^{-1}K^T)^{-\frac{1}{2}} + I \end{aligned}$$

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) using bilinear quadrilateral $\mathbf{Q_1}$ finite elements, and for this choice one can prove the following.

Corollary 3.3. *Let λ be an eigenvalue of $\mathcal{P}_{\mathrm{D1}}^{-1}\mathcal{A}$. 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.

Proof. Proposition 3.2 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

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

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 [7], applied to our case.

Theorem 3.4. *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*

$$ch^2 \leq \frac{\mathbf{x}^T M \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \leq Ch^2$$

$\forall \mathbf{x} \neq 0 \in \mathbb{R}^n$. The constants c and C are independent of h.

Theorem 3.5. *For* **Q1** *approximation on a quasi-uniform subdivision of $\mathbb{R}^2$ for which a shape regularity condition holds the Galerkin matrix K satisfies*

$$dh^2 \leq \frac{\mathbf{x}^T K \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \leq D$$

$\forall \mathbf{x} \neq 0 \in \mathbb{R}^n$. The constants d and D are positive and independent of h.

From Theorem 3.5 we obtain

$$\frac{1}{D} \leq \frac{\mathbf{x}^T\mathbf{x}}{\mathbf{x}^T K\mathbf{x}} \leq \frac{1}{dh^2}.$$

Therefore,

$$\begin{aligned}
\frac{ch^2}{D} \leq &\quad \nu &\quad \leq \frac{C}{d}, \\
\left(\frac{c}{D}\right)^2 h^4 \leq &\quad \nu^2 &\quad \leq \left(\frac{C}{d}\right)^2, \\
\left(\frac{c}{D}\right)^2 h^4 \leq &\quad 2\beta(\lambda - 1) &\quad \leq \left(\frac{C}{d}\right)^2, \\
\frac{1}{2\beta}\left(\frac{c}{D}\right)^2 h^4 + 1 \leq &\quad \lambda &\quad \leq \frac{1}{2\beta}\left(\frac{C}{d}\right)^2 + 1.
\end{aligned}$$

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

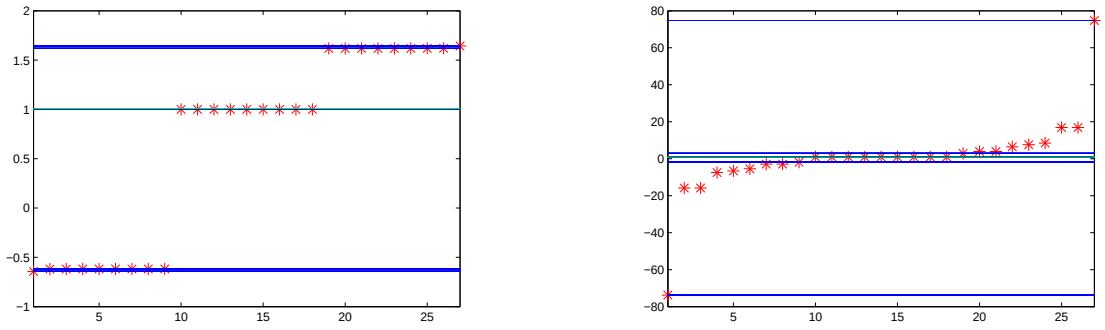
$$\frac{1}{2\beta}\alpha_1 h^4 + 1 \leq \lambda \leq \frac{1}{2\beta}\alpha_2 + 1, \tag{11}$$

where α_1 and α_2 are constants independent of h. For the 3D case, the equivalent results to Theorems 3.4 and 3.5 are

$$\begin{aligned}
ch^3 \leq &\quad \tfrac{\mathbf{x}^T M\mathbf{x}}{\mathbf{x}^T\mathbf{x}} &\quad \leq Ch^3 \ \forall \mathbf{x} \neq 0 \\
\text{and} \quad dh^3 \leq &\quad \tfrac{\mathbf{x}^T K\mathbf{x}}{\mathbf{x}^T\mathbf{x}} &\quad \leq Dh \ \forall \mathbf{x} \neq 0.
\end{aligned}$$

The extra h on each side will cancel, meaning (11) also holds in the 3D case (although the constants will be different). □

Note that the bounds in Corollary 3.3 do not get worse as we refine the mesh – they depend on h in a multiplicative way only – which suggests that this is an optimal preconditioner in the sense that it is independent of the mesh size. Figure 1 shows a plot of the actual eigenvalues and the bounds of Corollary 3.3 for two choices of β. As seen in the figure, 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 [6], Haber and Ascher [11], Maurer and Mittelmann [18],[19] or Ito and Kunisch [15], 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 3.2 (lines are the predicted, *s are the eigenvalues)

When using $\mathcal{P}_{\mathrm{D1}}$ 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

$$\mathcal{P} = \begin{bmatrix} 2\beta\tilde{M} & 0 & 0 \\ 0 & \tilde{M} & 0 \\ 0 & 0 & \tilde{K}M^{-1}\tilde{K}^T \end{bmatrix}, \tag{12}$$

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 (12) 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.

For any choice of PDE in a problem of the type in (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 [7]. We apply both algebraic and geometric multigrid routines. We thus have two preconditoners,

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

where $\tilde{K}$ denotes two geometric AMG V-cycles of `HSL` package `HSL_MI20` applied via a MATLAB interface [5]. For the geometric multigrid, as a smoother we use relaxed Jacobi, i.e. if we have to solve $B\mathbf{u} = \mathbf{f}$, take $D = \text{diag}(B)$ and iterate

$$\mathbf{u}^{(m+1)} = (I - \omega D^{-1}B)\mathbf{u}^{(m)} + \omega D^{-1}\mathbf{f} \tag{13}$$

where $\mathbf{u}^{(0)} = \mathbf{0}$ and for some relaxation parameter ω. Thus, here we have let $\widehat{K}$ denote two multigrid V-cycles with 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 [7, 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.

For the mass matrix, M, 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 [9] 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

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

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

$$\mathbf{y}^{(k+1)} = w_{k+1}(S\mathbf{y}^{(k)} + g - \mathbf{y}^{(k-1)}) + \mathbf{y}^{(k-1)}, \tag{14}$$

where $w_{k+1} = \frac{T_k(1/\rho)}{\rho T_{k+1}(1/\rho)}$ and $S = D^{-1}B$ (see Varga [23, 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,

$$||\mathbf{u} - \mathbf{y}^{(k)}||_2 \leq \max_{r\in[-\rho,\rho]} |\hat{T}_k(r)| \; ||\mathbf{u} - \mathbf{u}^{(0)}||_2. \tag{15}$$

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, $\max_{r\in[-\rho,\rho]} |\hat{T}_k(r)| \approx 10^{-6}$. This bound shows that $\tilde{M}$ is almost exactly M but is still a very inexpensive way of inverting this operator.

4 Constraint Preconditioning

As noted in Section 3, the coefficient matrix in (9) is of a saddle-point form. In recent years, the projected conjugate gradient (PPCG) method [10] has become an increasingly popular method for solving saddle-point systems. The method requires the use of a preconditioner that has a very specific structure. If, as in (10), we write the coefficient matrix $\mathcal{A}$ of (9) 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^TAZ and Z^TGZ 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 (10) can be written as

$$x^* = Wx_w^* + Zx_z^*. \tag{16}$$

Substituting (16) into (10) and premultiplying the resulting system by $\begin{bmatrix} W^T & 0 \\ Z^T & 0 \\ 0 & I \end{bmatrix}$, we obtain the linear system

$$\begin{bmatrix} W^TAW & WAZ & WB^T \\ Z^TAW & Z^TAZ & 0 \\ BW & 0 & 0 \end{bmatrix} \begin{bmatrix} x_w^* \\ x_z^* \\ y \end{bmatrix} = \begin{bmatrix} W^T\mathbf{c} \\ Z^T\mathbf{c} \\ \mathbf{d} \end{bmatrix}.$$

Therefore, we may compute x_w^* by solving

$$BWx_w^* = \mathbf{d},$$

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

$$A_{zz}x_z^* = c_z,$$

where

$$\begin{aligned} A_{zz} &= Z^TAZ, \\ c_z &= Z^T\left(\mathbf{c} - AAX_w^*\right). \end{aligned}$$

If a preconditioner of the form Z^TAZ is used, then Gould *et al* [10] suggest terminating the iteration when the easily computable value $\|x_k - x_z^*\|_{Z^TGZ}$ has sufficiently decreased. They also show that the PCG algorithm may be rewritten without the need for Z at all: this results in the PPCG algorithm, Algorithm 4.1.

Algorithm 4.1. *Choose an initial point x satisfying $Bx = d$ and compute $r = Ax - c$. Solve* $\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$ *and set $p = -g$, $y = v$, $r \leftarrow r - B^Ty$. Repeat the following steps until a convergence test is satisfied:*

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

If y^* is required, then one extra step must be carried out to compute it. However, in our case, y^* corresponds to the Lagrange multipliers which we are not interested in calculating. Note, in Algorithm 4.1, $\|x_k - x_z^*\|_{Z^TGZ} = r^Tg$ (see [10]) and, hence, we can still efficiently calculate Gould *et al*'s suggested measure for termination.

The following theorem gives the main properties of the preconditioned matrix $\mathcal{P}^{-1}\mathcal{A}$: the proof can be found in [16].

Theorem 4.2. *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*

$$Z^TAZx_z = \lambda Z^TGZx_z. \tag{17}$$

Additionally, if G is nonsingular, then the eigenvalues defined by (17) 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 [16].

Clearly, for our problem (9), 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 4.2, 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 [24]. 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,

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

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 $z = \begin{bmatrix} z_1^T & z_2^T & z_3^T \end{bmatrix}^T$ and $r = \begin{bmatrix} r_1^T & r_2^T & r_3^T \end{bmatrix}^T$, we may solve systems of the form $\mathcal{P}_{C1} z = r$ by carrying out the following steps:

- Solve
$$Mz_3 = -r_1, \tag{18}$$
- Solve
$$2\beta K^T M^{-1} K z_2 = r_2 - K^T z_3, \tag{19}$$
- Solve
$$Mz_1 = Kz_2 - r_3. \tag{20}$$

As noted in Section 3, systems of the form (18) and (20) may be solved efficiently because M is a mass matrix. We will discuss the efficient (approximate) solution of (19) at the end of this section.

Proposition 4.3. *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*

$$\left(\frac{1}{2\beta}\left(K^T M^{-1} K\right)^{-1} M + I\right) x = \lambda x. \tag{21}$$

Proof. From Theorem 4.2, $\mathcal{P}_{C1}^{-1}\mathcal{A}$ has

- $2n$ eigenvalues at 1; and
- the remaining eigenvalues satisfy

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

This is equivalent to the generalized eigenvalue problem (21). □

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) using quadrilateral $\mathbf{Q_1}$ finite elements and, for this choice, one can prove the following.

Corollary 4.4. *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 h.

Proof. From Proposition 4.3, $\lambda = 1$ 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 3.3 we obtain the desired result. □

Therefore, as we refine the mesh, the bounds in Corollary 4.4 will not get worse. This suggests that this will be an optimal preconditioner for (9). 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 (19). As in Section 3, instead of exactly carrying out solves with K, we may approximate K by a matrix $\tilde{K}$. If our approximation is good enough, then the spectral bounds will be close to those in Corollary 4.4. 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

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

Here, again, $\tilde{K}$ 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` [5], 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 (15) 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 multipications with these matrices.

5 Results

We illustrate our methods with four different examples. For each example we compare the time to solve the system using 'backslash' in MATLAB, MINRES with preconditioners $\mathcal{P}_{\rm D2}$ and $\mathcal{P}_{\rm D3}$, 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. 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. 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. In all our examples we use $\beta = 10^{-2}$.

Example 5.1. *Let $\Omega = [0,1]^m$, where $m = 2, 3$, and consider the problem*

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

$$\text{s.t.} \qquad -\nabla^2 u = f \qquad \text{in } \Omega \tag{22}$$

$$u = \hat{u}|_{\partial\Omega} \qquad \text{on } \partial\Omega \tag{23}$$

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,

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

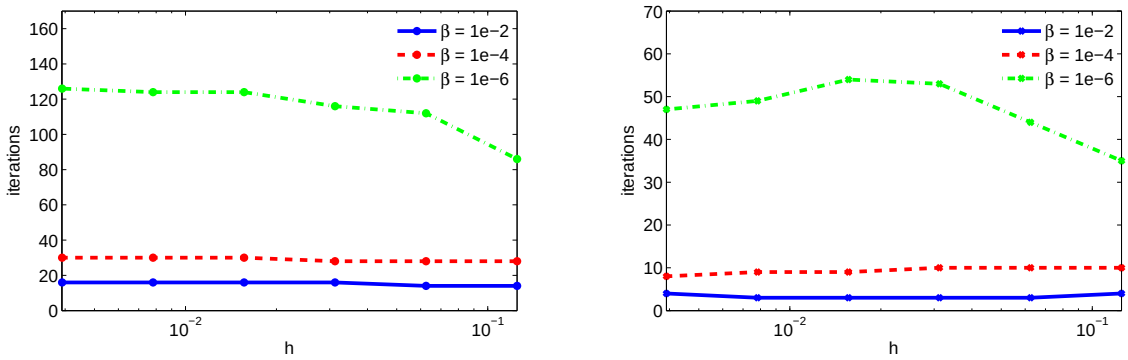
i.e. $\hat{u}$ is bi- or tri-quadratic (depending on whether $m = 2$ or 3) with a peak of unit height at the origin and is zero outside $[0,\frac{1}{2}]^m$.

In Tables 1 and 2, we consider the 2D version of Example 5.1 with tolerances 10^{-6} and 10^{-12}, respectively. We observe that the number or iterations required by our iterative methods grow only slightly as we refine our mesh size with the MINRES preconditioners, and not at all when using PPCG with $\mathcal{P}_{C2}$. 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 implemention it makes infeasible for smaller values of h.

Here the geometric multigrid preconditioner ($\mathcal{P}_{D3}$) converges in a number of iterations less than or equal to the corresponing AMG preconditioner ($\mathcal{P}_{D2}$), but also that the AMG preconditioner is significantly the faster of the two. 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 and 4, we consider the 3D version of Example 5.1 with tolerances 10^{-6} and 10^{-12}, respectively. Again we see much the same behaviour – here MATLAB's backslash and `ldl` methods run out of memory at $h = 2^{-5}$. Here the geometric multigrid version of the MINRES preconditioner is the most efficient, both in terms of the number of iterations and the time taken. Again, the timings are close to scaling linearly with the problem size.

In Figure 2, we compare the number of iterations carried out by both the MINRES method with preconditioner $\mathcal{P}_{D3}$ (Figure 2(a)) and the PPCG method with preconditioner $\mathcal{P}_{C2}$ (Figure 2(b)) for the tolerance 10^{-12} and different values of β. As expected, for fixed β, the number of iterations does not grow significantly as h decreases – any growth, as in MINRES for $\beta = 10^{-6}$, settles to a constant. For fixed values of h, decreasing β results in an increasing number of iterations.



(a) MINRES (b) PPCG

Figure 2: Number of iterations for MINRES with $\mathcal{P}_{D3}$ and PPCG with $\mathcal{P}_{C2}$ to converge for $\beta = 10^{-2},\ 10^{-4}$ and 10^{-6}

Example 5.2. *Again, let* $\Omega = [0,1]^m$*, where* $m = 2, 3$*, and consider the problem*

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

$$\text{s.t.} \qquad -\nabla^2 u = f \qquad \text{in } \Omega \tag{24}$$

$$u = 0 \qquad \text{on } \partial\Omega \tag{25}$$

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* $[\frac{1}{2}, \frac{1}{2}]$.

Tables 5 to 8 show our findings for Example 5.2 which is another Dirichlet control problem – this time with a different cost functional. Here we see identical behaviour to that in Example 5.1, which demonstrates that our method is not dependent on $\hat{u}$.

Example 5.3. *Let* $\Omega = [0,1]^2$ *and consider the problem*

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

$$\text{s.t.} \qquad -\nabla^2 u = f \qquad \text{in } \Omega \tag{26}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega \tag{27}$$

where

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

Example 5.3 is an example of a distributed control problem with a Neumann boundary condition. In this case the stiffness matrix K is singular. We comment that for simple forward solution of the Neumann problem, a singular multigrid cycle is possible [12, Chapter 12], whereas in the control problem we require a definite preconditioner, hence a singular multigrid cycle is not usable. This is not a difficulty, as we simply pin one of the nodes to remove the singularity, which 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 9 and 10 show 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 have both grown. However, the overall behaviour is simliar - we still get mesh size independent convergence and nearly linear complexity. Interestingly, the geometric multigrid based preconditioner, $\mathcal{P}_{D3}$, performs worse here relative to the other examples – presumably as a result of the way we pin the node, which AMG is better suited to cope with.

Example 5.4. *Let* $\Omega = [0,1]^2$ *and consider the problem*

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

$$\text{s.t.} \qquad -\nabla^2 u = f \qquad \text{in } \Omega \tag{28}$$

$$u = \hat{u}|_{\partial\Omega} \text{ on } \partial\Omega_1 \qquad \text{and} \qquad \frac{\partial u}{\partial n} = 0 \text{ on } \partial\Omega_2 \tag{29}$$

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 5.4 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 11 and 12 lie somewhere in between those of Examples 5.1 and 5.2 and those of Example 5.3.

6 Conclusion

We have presented optimal preconditioners for distributed control problems. We have demonstrated that our preconditioners work effectively with regularization parameter $\beta = 10^{-2}$, and although the approximations become less valid as β approaches zero they still give mesh size independent convergence down to $\beta = 10^{-6}$. Only a handful of papers in the literature consider the saddle-point structure of the matrices when solving problems of this type, and we believe that using this stucture is a good way of creating efficient algorithms. A large amount of work that has been done on solving saddle point systems: e.g. see the survey paper by Benzi, Golub and Liesen [2].

We have presented two classes of preconditioners for the simplest case of PDE, namely the Poisson equation, but our methods are more general than that. For any PDE, if there is a preconditioner available for the forward problem one could use that as $\tilde{K}$ and should still get good convergence. We have included results using preconditioners based both on algebraic and geometric multigrid routines.

There are some β-independent methods in the literature, in particular the recent method by Schöberl and Zulehner [22], which also uses block preconditioners, gives both h and β inedependent convergence. However, their method is specific to this PDE problem, and so is not as general as the methods we present here.

Although we have presented the distributed control problem, changing the problem to, for example, a boundary control problem, or the addition of bound constraints, will require the solution of a matrix with a similar block structure to (9) and we therefore anticipate developing our ideas further to give optimal preconditioners for these other classes of problems.

References

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

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

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

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

[5] 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.

[6] 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.

[7] 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.

[8] M. Engel and M. Griebel. Flow Simulation on Moving Boundary-Fitted Grids and Application to Fluid-Structure Interaction Problems. *International Journal for Numerical Methods in Fluids*, 50(4):50, 2006.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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 5.1 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-6} 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	0.02 (7)	0.13 (7)	0.01 (2)	0.004 (6)
2^{-3}	147	0.002	0.03 (9)	0.16 (9)	0.02 (2)	0.02 (8)
2^{-4}	675	0.01	0.05 (9)	0.21 (9)	0.03 (2)	0.14 (8)
2^{-5}	2883	0.08	0.14 (9)	0.41 (9)	0.06 (1)	0.85 (7)
2^{-6}	11907	0.46	0.61 (9)	1.29 (9)	0.29 (1)	5.94 (7)
2^{-7}	48387	3.10	2.61 (9)	5.09 (9)	1.92 (2)	36.1 (7)
2^{-8}	195075	15.5	15.0 (11)	23.6 (9)	8.79 (2)	190 (7)
2^{-9}	783363	—	75.6 (11)	136 (9)	39.2 (2)	—

Table 2: Comparison of times and iterations to solve Example 5.1 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-12} 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	0.03 (12)	0.15 (12)	0.03 (4)	0.004 (6)
2^{-3}	147	0.002	0.05 (14)	0.20 (14)	0.03 (4)	0.03 (16)
2^{-4}	675	0.01	0.08 (14)	0.28 (14)	0.04 (3)	0.25 (17)
2^{-5}	2883	0.08	0.21 (14)	0.63 (16)	0.10 (3)	1.77 (17)
2^{-6}	11907	0.46	1.02 (16)	2.13 (16)	0.53 (3)	12.23 (17)
2^{-7}	48387	2.28	4.40 (16)	8.60 (16)	2.45 (3)	74.78 (16)
2^{-8}	195075	15.5	23.5 (18)	40.2 (16)	13.7 (4)	357 (16)
2^{-9}	783363	—	130 (20)	189 (16)	60.9 (4)	—

Table 3: Comparison of times and iterations to solve Example 5.1 in 3D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-6} 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}	81	0.001	0.02 (7)	0.14 (8)	0.01 (2)	0.01 (6)
2^{-3}	1029	0.013	0.13 (9)	0.26 (8)	0.06 (2)	0.56 (8)
2^{-4}	10125	25.1	1.89 (8)	1.69 (8)	0.96 (2)	25.7 (8)
2^{-5}	89373	—	22.1 (8)	15.9 (8)	11.1 (2)	—

Table 4: Comparison of times and iterations to solve Example 5.1 in 3D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-12} 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}	81	0.001	0.04 (13)	0.15 (11)	0.02 (5)	0.004 (6)
2^{-3}	1029	0.013	0.21 (15)	0.34 (13)	0.12 (6)	1.13 (16)
2^{-4}	10125	25.1	3.42 (16)	2.75 (14)	1.68 (5)	57.1 (17)
2^{-5}	89373	—	39.4 (16)	28.3 (15)	19.6 (4)	—

Table 5: Comparison of times and iterations to solve Example 5.2 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-6} 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	0.02 (7)	0.13 (7)	0.01 (2)	0.003 (3)
2^{-3}	147	0.002	0.03 (7)	0.15(7)	0.02 (2)	0.01 (3)
2^{-4}	675	0.01	0.05(7)	0.19 (7)	0.03 (2)	0.05 (2)
2^{-5}	2883	0.06	0.15 (9)	0.52 (9)	0.08 (2)	0.36 (2)
2^{-6}	11907	0.36	0.79 (9)	1.65 (9)	0.38 (2)	1.83 (1)
2^{-7}	48387	2.27	3.08 (9)	7.18 (9)	1.68 (2)	13.4 (1)
2^{-8}	195075	15.5	12.7 (9)	13.6 (9)	7.88 (2)	79.3 (1)
2^{-9}	783363	—	70.7 (11)	113 (9)	50.5 (3)	—

Table 6: Comparison of times and iterations to solve Example 5.2 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-12} 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	0.04 (12)	0.14 (8)	0.02 (3)	0.003 (3)
2^{-3}	147	0.002	0.05 (14)	0.19 (12)	0.02 (3)	0.02 (9)
2^{-4}	675	0.01	0.09 (14)	0.28 (14)	0.04 (3)	0.09 (6)
2^{-5}	2883	0.06	0.21 (14)	0.70 (14)	0.10 (3)	0.50 (4)
2^{-6}	11907	0.37	1.07 (16)	3.12 (16)	0.50 (3)	2.95 (3)
2^{-7}	48387	2.27	4.41 (16)	12.3 (16)	2.20 (3)	16.8 (2)
2^{-8}	195075	15.5	23.9 (18)	40.3 (16)	9.96 (3)	97.9 (2)
2^{-9}	783363	—	132 (20)	189 (16)	62.5 (4)	—

Table 7: Comparison of times and iterations to solve Example 5.2 in 3D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-6} 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}	81	0.001	0.03 (8)	0.05 (8)	0.01 (2)	0.01 (4)
2^{-3}	1029	0.013	0.12 (8)	0.16 (8)	0.06 (2)	0.30 (4)
2^{-4}	10125	24.7	1.88 (8)	1.60 (8)	0.95 (2)	13.1 (3)
2^{-5}	89373	—	22.1 (8)	15.8 (8)	11.1 (2)	—

Table 8: Comparison of times and iterations to solve Example 5.2 in 3D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-12} 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}	81	0.001	0.04 (13)	0.07 (11)	0.02 (5)	0.01 (4)
2^{-3}	1029	0.013	0.22 (15)	0.24 (13)	0.09 (4)	0.65 (12)
2^{-4}	10125	24.7	3.17 (15)	2.64 (14)	1.65 (5)	21.4 (7)
2^{-5}	89373	—	37.2 (15)	28.3 (15)	16.5 (4)	—

Table 9: Comparison of times and iterations to solve Example 5.3 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-6} 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}	72	0.0007	0.04 (15)	0.16 (11)	0.02 (3)	0.005 (5)
2^{-3}	240	0.003	0.05 (14)	0.19 (11)	0.02 (3)	0.02 (3)
2^{-4}	864	0.01	0.10 (15)	0.28 (12)	0.04 (3)	0.07 (2)
2^{-5}	3264	0.08	0.25 (15)	0.75 (17)	0.12 (3)	0.40 (2)
2^{-6}	12672	0.55	1.03 (15)	2.43 (17)	0.62 (3)	2.06 (1)
2^{-7}	49920	3.99	4.52 (15)	10.1 (17)	2.72 (3)	12.7 (1)
2^{-8}	198144	28.2	22.3 (17)	47.6 (19)	11.5 (3)	84.4 (1)
2^{-9}	789504	—	89.2 (17)	210 (21)	46.8 (3)	—

Table 10: Comparison of times and iterations to solve Example 5.3 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-12} 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}	72	0.0007	0.06 (20)	0.22 (19)	0.04 (6)	0.008 (11)
2^{-3}	240	0.003	0.09 (22)	0.27 (20)	0.04 (6)	0.04 (12)
2^{-4}	864	0.01	0.15 (24)	0.45 (23)	0.07 (6)	0.19 (10)
2^{-5}	3264	0.08	0.41 (24)	1.07 (26)	0.21 (6)	1.11 (9)
2^{-6}	12672	0.55	1.59 (24)	3.65 (26)	0.97 (6)	5.80 (7)
2^{-7}	49920	3.99	7.11 (24)	16.31 (28)	4.55 (6)	30.7 (6)
2^{-8}	198144	28.2	35.8 (28)	78.7 (32)	19.2 (6)	161 (5)
2^{-9}	789503	—	142 (28)	315 (32)	76.2 (6)	—

Table 11: Comparison of times and iterations to solve Example 5.4 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-6} 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}	48	0.0004	0.03 (9)	0.15 (9)	0.02 (3)	0.005 (8)
2^{-3}	192	0.002	0.04 (11)	0.17 (9)	0.02 (3)	0.03 (8)
2^{-4}	768	0.01	0.07 (11)	0.26 (11)	0.04 (3)	0.14 (8)
2^{-5}	3072	0.07	0.18 (11)	0.52 (11)	0.08 (2)	0.81 (7)
2^{-6}	12288	0.43	0.76 (11)	1.68 (11)	0.33 (1)	5.53 (7)
2^{-7}	49152	2.38	3.53 (11)	7.39 (11)	2.12 (2)	35.0 (7)
2^{-8}	196608	15.3	15.3 (11)	32.1 (11)	11.6 (3)	199 (7)
2^{-9}	786432	—	70.4 (13)	114 (11)	46.0 (3)	—

Table 12: Comparison of times and iterations to solve Example 5.4 in 2D for different mesh sizes (h) (with $3n$ unknowns) to a tolerance of 10^{-12} for MATLAB's backslash method, MINRES with preconditioners $\mathcal{P}_{\mathrm{D}2}$, $\mathcal{P}_{\mathrm{D}3}$, 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	0.0004	0.05 (16)	0.17 (14)	0.03 (4)	0.006 (10)
2^{-3}	192	0.002	0.06 (16)	0.22 (16)	0.04 (5)	0.05 (17)
2^{-4}	768	0.01	0.11 (18)	0.33 (16)	0.05 (4)	0.27 (17)
2^{-5}	3072	0.07	0.28 (18)	0.76 (18)	0.14 (4)	1.72 (17)
2^{-6}	12288	0.43	1.18 (18)	2.82 (18)	0.73 (4)	11.4 (17)
2^{-7}	49152	2.38	6.19 (20)	10.66 (18)	3.33 (4)	67.4 (16)
2^{-8}	196608	15.3	29.0 (22)	56.0 (20)	16.9 (5)	370 (16)
2^{-9}	786432	—	124 (24)	200 (20)	66.5 (5)	—