Journal of Computational and Applied Mathematics 343 (2018) 240–249

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Simple efficient solvers for certain ill-conditioned systems of linear equations, including $H(\mathrm{div})$ problems

JaEun Ku [a], Lothar Reichel [b,*]

[a] Department of Mathematics, Oklahoma State University, 401 Mathematical Sciences, Stillwater, OK 74078, USA
[b] Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA

ARTICLE INFO

Article history:
Received 28 July 2017
Received in revised form 6 December 2017

Keywords:
Finite element method
Nearly singular system
Variational problem

ABSTRACT

Simple, accurate, and efficient iterative methods for the solution of a nearly singular variational problem are described. The systems considered arise, e.g., when seeking to determine the flux of second order elliptic partial differential equations. Each (outer) iteration uses a robust and efficient solver, which may be a direct or iterative method, and only a few (outer) iterations are required to obtain an approximate solution. Reduced rank extrapolation may be applied to speed up the convergence.

1. Introduction

This paper describes simple efficient iterative methods for the computation of an accurate approximate solution of variational problems of the form

$$(\nabla\cdot\boldsymbol{\sigma},\nabla\cdot\boldsymbol{\tau})+\delta(\boldsymbol{\sigma},\boldsymbol{\tau})=F(\boldsymbol{\tau})\quad\text{for all }\boldsymbol{\tau}\in H(\mathrm{div}),\tag{1.1}$$

where δ is a constant with $0<\delta\ll 1$, $F(\cdot)$ is a linear functional, and $(\cdot,\cdot)$ denotes the standard inner product in L_2. Variational problems of this kind arise in a variety of applications, including in the context of preconditioning in mixed finite element methods and least-squares finite element methods; see [1,2]. Moreover, the finite element method proposed in [3] for determining the flux $\boldsymbol{\sigma}$ for second order elliptic partial differential equations (PDEs) leads to an equation of the type (1.1). We present more details of this application in Section 2.

The parameter $\delta>0$ in (1.1) provides a connection between the approximation spaces for the solution u of a second order elliptic PDE and the flux $\boldsymbol{\sigma}=-\nabla u$. The flux is the quantity of primary interest in many problems that arise in the sciences and engineering. It is desirable that δ be small in order to be able to determine an accurate approximation of the flux with a modest computational effort. This is explained in Section 2. However, when $\delta>0$ is "tiny", discretization of (1.1) gives rise to a linear system of algebraic equations (LSAEs) that is nearly singular. The near-singularity of the LSAE can make its accurate solution difficult. This is illustrated in Section 5.

The aim of this paper is to provide simple, efficient, and accurate solvers for discretizations of variational problems of the form (1.1) with a small parameter $\delta>0$. A difficulty is that the quality of the computed solution deteriorates as δ decreases to zero, and is poor for small values of δ, when the solution is computed by a direct or iterative method applied to the linear system of equations with a nearly singular matrix obtained by discretizing Eq. (1.1); see Section 5. We present a robust solution method for the LSAE obtained from the discretization of (1.1) that produces accurate solutions independent of $\delta>0$. This method is based on replacing the stiffness matrix by a better conditioned matrix, and solving the LSAE so

* Corresponding author.
E-mail addresses: jku@okstate.edu (J. Ku), reichel@math.kent.edu (L. Reichel).

https://doi.org/10.1016/j.cam.2018.05.002
0377-0427/

obtained by an iterative method. An advantage of our solution approach, when compared with available schemes, such as those described in [1,2], is its simplicity. In particular, our approach uses standard FEM spaces and is easy to implement.

We are interested in solution methods that are well suited for LSAEs of small to moderate size, as well as in solution methods suitable for large-scale LSAEs. All methods presented for the solution of (1.1) are iterative, however, their design depends on the size of the linear system of algebraic equations.

Iterative methods for the solution of singular and nearly singular LSAEs have received considerable attention in the literature; see, e.g., [4–8] and references therein. We will describe a fast iterative solution method that exploits the structure of Eq. (1.1). Our method is easy to implement. Specifically, we use an iterative or direct solver for the LSAE with parameter $\delta = 1$ to determine a solution of the LSAE with $\delta > 0$ "tiny". When $\delta = 1$, the matrix of the LSAE is symmetric positive definite and there are many robust and efficient solvers available for the resulting system; see [9]. Several such systems have to be solved in our iterative method for the solution of the discretized linear system with $\delta > 0$ small obtained from (1.1). Numerical examples reported in Section 5 show the number of iterations to be small and neither sensitive to $\delta > 0$ nor to the mesh size. Moreover, the number of iterations can be reduced by the application of a vector extrapolation method. We will apply reduced rank (vector) extrapolation (RRE). A nice survey of extrapolation methods is provided by Brezinski and Redivo Zaglia [10]. Vector extrapolation methods, and in particular RRE, are discussed in [11–14] as well as in references therein.

The LSAE obtained by discretization of (1.1) with $\delta = 1$ can be solved in a variety of ways. We will solve it by a direct method based on sparse Cholesky factorization of the system matrix when the LSAE is of small to moderate size. Large-scale systems can be solved by a preconditioned conjugate gradient method or a multigrid method. Our solution methods for LSAEs with $\delta > 0$ small takes advantages of these well-developed solvers, and are easy to implement as a result. In this paper, we use MATLAB functions or algebraic multigrid methods developed in [15–18] for solving the LSAE with $\delta = 1$.

This paper is organized as follows. Section 2 describes the background for Eq. (1.1) and presents some preliminary results. In particular, we provide a justifications for choosing $\delta > 0$ small. Section 3 discusses approximation spaces. The basis elements are defined in Section 4, where we also derive the linear system of algebraic equations associated with (1.1). Section 5 describes some numerical examples that illustrate the theory and the fast convergence of the iterative methods proposed. Concluding remarks can be found in Section 6.

2. Problem formulation

We describe an example that gives rise to a variational problem of the form (1.1) and discuss our reason for choosing a small value of δ. Our description is quite brief and we refer interested readers to [3] for further details.

Let $\Omega \in \mathbb{R}^n$, with $n = 2$ or $n = 3$, be a convex polygonal domain with boundary $\partial\Omega$, and let $\|\cdot\|_s$ denote the norm in the Sobolev space $H^s(\Omega)$ and $(\cdot,\cdot)$ the inner product in $L_2(\Omega)$. Introduce the model second order PDE

$$-\nabla\cdot\mathcal{A}\nabla u = f \text{ in } \Omega \tag{2.1}$$

with boundary condition

$$u = 0 \text{ on } \partial\Omega,$$

where $\mathcal{A}$ is a symmetric positive definite matrix and $f \in L_2(\Omega)$. For ease of exposition, we let $\mathcal{A}$ be the identity matrix.

In many applications, the flux $\boldsymbol{\sigma} = -\nabla u$ is the quantity of primary interest. To approximate the flux accurately, one often transforms the second order equation (2.1) into a system of first order equations,

$$\boldsymbol{\sigma} + \nabla u = 0 \text{ in } \Omega, \tag{2.2}$$

$$\nabla\cdot\boldsymbol{\sigma} = f \text{ in } \Omega. \tag{2.3}$$

This system can be discretized and solved by a mixed finite element method or by a least-squares finite element method. These methods produce accurate approximations of the flux. However, a drawback of both these methods is that they approximate both u and $\boldsymbol{\sigma}$ simultaneously. This results in an unnecessarily large problem size and makes the computational effort required to solve the resulting LSAE unnecessarily large when u is not required. To overcome this difficulty, a new hybrid finite element method has been proposed in [3]. This hybrid method is based on the following formulation. Taking an inner product with $\boldsymbol{\tau} \in H(\mathrm{div})$ in (2.2) and multiplying by δ, taking an inner product with $\nabla\cdot\boldsymbol{\tau} \in L_2(\Omega)$ in (2.3), and adding the equations so obtained yields

$$(\nabla\cdot\boldsymbol{\sigma}, \nabla\cdot\boldsymbol{\tau}) + \delta(\boldsymbol{\sigma} + \nabla u, \boldsymbol{\tau}) = (f, \nabla\cdot\boldsymbol{\tau}) \text{ for all } \boldsymbol{\tau} \in H(\mathrm{div}).$$

Using integration by part, we obtain $(\nabla u, \boldsymbol{\tau}) = -(u, \nabla\cdot\boldsymbol{\tau})$ and the above equation can be expressed as

$$(\nabla\cdot\boldsymbol{\sigma}, \nabla\cdot\boldsymbol{\tau}) + \delta(\boldsymbol{\sigma}, \boldsymbol{\tau}) = (f + \delta u, \nabla\cdot\boldsymbol{\tau}) \text{ for all } \boldsymbol{\tau} \in H(\mathrm{div}). \tag{2.4}$$

This is a variational problem of the form (1.1) with $F(\boldsymbol{\tau}) = (f + \delta u, \nabla\cdot\boldsymbol{\tau})$.

However, since the primary variable u is not known, we cannot use (2.4) to compute the flux $\boldsymbol{\sigma}$. We therefore first compute an approximation u_H of u on a coarse mesh (with mesh size H). Then we compute an approximation $\boldsymbol{\sigma}_h$ of $\boldsymbol{\sigma}$ on a fine mesh (with mesh size h) by solving

$$(\nabla \cdot \boldsymbol{\sigma}_h, \nabla \cdot \boldsymbol{\tau}_h) + \delta(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) = (f + \delta u_H, \nabla \cdot \boldsymbol{\tau}_h) \tag{2.5}$$

for all $\boldsymbol{\tau}_h$ in some discrete space.

The following error estimates are shown in [3, Theorem 4.4] for any mixed-type finite element approximation spaces such as Raviart-Thomas (RT) spaces or Brezzi-Douglas-Marini (BDM) spaces; see [19,20,22]. The estimates shed lights on the roles δ, h, and H, and on how their sizes should be chosen.

Theorem 2.1. *Let $\boldsymbol{\sigma}_h$ satisfy (2.5) and assume that $\|u - u_H\|_1 \leq CH\|u\|_2$ for some constant $C > 0$ independent of H. Then*

$$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\| \leq Ch\|\boldsymbol{\sigma}\|_1 + C\sqrt{\delta}H^2\|u\|_2 \text{ for } RT_0 \text{ spaces}$$

and

$$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\| \leq Ch^2\|\boldsymbol{\sigma}\|_2 + C\sqrt{\delta}H^2\|u\|_2 \text{ for } BDM_1 \text{ spaces}.$$

Remark 2.2. The crude approximation u_H can be computed by using one of many well-known numerical methods, such as a standard Galerkin method. Due to the fact that the parameter δ is small, the cost of computing u_H typically is negligible as explained in the following subsection. □

2.1. The role of the parameter δ

The error estimates of Theorem 2.1 suggest that the ratio between h and H should be

$$h = \sqrt{\delta}H^2 \text{ for } RT_0 \text{ spaces and } h^2 = \sqrt{\delta}H^2 \text{ for } BDM_1 \text{ spaces}.$$

Choosing a small value of δ makes it possible to use a small fine mesh size h in conjunction with a large coarse mesh size H. This implies that the cost of the computations on the coarse mesh is negligible compared to the cost of the computations on the fine mesh when $\delta > 0$ is sufficiently small. We are therefore interested in using a small value of δ. On the other hand, a small δ results in a LSAE with a nearly singular matrix, and this may cause numerical difficulties if not handled properly. In Section 4, we describe simple, efficient, and reliable solvers for these LSAEs with a near-singular matrix.

2.2. Two linear operators

Define the linear operators A and A_δ by

$$(A\boldsymbol{\sigma}, \boldsymbol{\tau}) = (\nabla \cdot \boldsymbol{\sigma}, \nabla \cdot \boldsymbol{\tau}) + (\boldsymbol{\sigma}, \boldsymbol{\tau}), \tag{2.6}$$

$$(A_\delta\boldsymbol{\sigma}, \boldsymbol{\tau}) = (\nabla \cdot \boldsymbol{\sigma}, \nabla \cdot \boldsymbol{\tau}) + \delta(\boldsymbol{\sigma}, \boldsymbol{\tau}). \tag{2.7}$$

The operator A is symmetric and positive definite. Its smallest eigenvalue is larger than or equal to 1. Hence, all eigenvalues of A^{-1} lie between 0 and 1. The operator A_δ can be expressed as

$$A_\delta = A + (\delta - 1)I.$$

This shows that A_δ also is symmetric and positive definite for $\delta > 0$ with its eigenvalues bounded below by δ.

3. Approximation spaces

To compute an approximate solution of (1.1), we first discretize the domain Ω by triangles. Let $\mathcal{T}_h$ be a quasi-uniform family of triangulations of Ω, where $h > 0$ is a parameter representative of the diameter of the triangles; see [21]. Fig. 3.1 shows a simple example. The triangles of $\mathcal{T}_h$ are denoted by T.

We will use the Raviart–Thomas finite element spaces to approximate $\boldsymbol{\sigma} \in H(\text{div})$. For each nonnegative integer r, the Raviart–Thomas space of index r is given by

$$\mathbf{V}_h = \{\mathbf{v} \in H(\text{div}) : \mathbf{v}|_T \in P_r(T) + (x, y)P_r(T) \text{ for all } T \in \mathcal{T}_h\}.$$

Here $P_r(T)$ denotes the set of polynomial functions of degree at most r on T. Note that the function $\mathbf{v} \in \mathbf{V}_h$ is continuous across the edges so that $\mathbf{V}_h \subset H(\text{div})$, i.e.,

$$[\mathbf{v}]_E \cdot \nu_E = 0 \text{ for all interior edges } E,$$

where $[\mathbf{v}]_E = \mathbf{v}|_{T_+} - \mathbf{v}|_{T_-}$ denotes the jump of $\mathbf{v}$ across the edge $E = T_+ \bigcup T_-$ and ν_E is a unit normal vector to E. The space $\mathbf{V}_h$ has the approximation property

$$\inf_{\boldsymbol{\tau}_h \in \mathbf{V}_h} \|\boldsymbol{\sigma} - \boldsymbol{\tau}_h\| \leq Ch^{r+1}\|\boldsymbol{\sigma}\|_{r+1};$$

see [23] and references therein for a more detailed discussion on the properties and implementation of RT spaces.

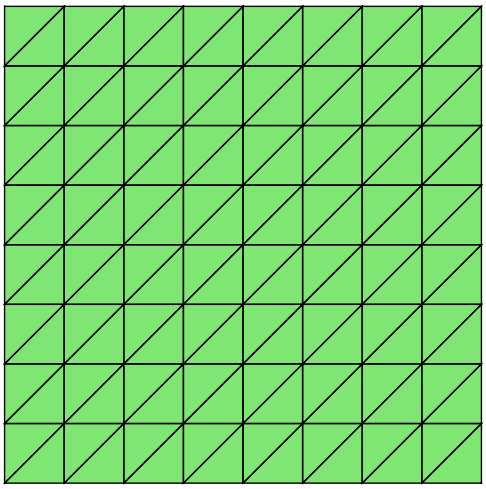

Fig. 3.1. Uniform mesh of mesh size $h = 1/2^3$. There are 208 edges, which equals the degrees of freedom.

3.1. Decomposition of $\mathbf{V}_h$

Recall that $P_r(T)$ denotes the set of polynomials of degree at most r on T. Introduce the space

$$W_h = \{s \in H^1 : s|_T \in P_{r+1}(T)\} \tag{3.1}$$

of continuous piece-wise polynomials of degree $r+1$ with a derivative, as well as the space

$$S_h = \{q \in L_2 : q|_T \in P_r(T)\}$$

of (possibly discontinuous) piece-wise polynomials of degree r on T. Define the discrete gradient operator $\mathbf{grad}_h : S_h \to \mathbf{V}_h$ by

$$(\mathbf{grad}_h q, \mathbf{v}) = -(q, \nabla \cdot \mathbf{v}) \text{ for all } \mathbf{v} \in \mathbf{V}_h. \tag{3.2}$$

Then we have the following (discrete) Helmholtz decomposition (cf. [24]),

$$\mathbf{V}_h = \mathbf{grad}_h S_h \oplus \mathbf{curl}\, W_h.$$

This decomposition is orthogonal with respect to both the L_2 and $H(\mathrm{div})$ inner products. Using the orthogonality, one can easily show that the two summand spaces $\mathbf{grad}_h S_h$ and $\mathbf{curl}\, W_h$ are invariant under A and A_δ.

We also define a local L_2-projection $P_h : L_2(\Omega) \to S_h$ as follows: Let for $f \in L_2(\Omega)$,

$$(P_h f, q_h) = (f, q_h) \text{ for all } q_h \in S_h. \tag{3.3}$$

We refer to [1] for a more detailed presentation of the material in this subsection.

3.2. An iterative method

Using (2.7), we can express Eq. (2.5) as

$$(A_\delta \boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) = (f + \delta u_H, \nabla \cdot \boldsymbol{\tau}_h) \text{ for all } \boldsymbol{\tau} \in \mathbf{V}_h,$$

where we note that $\nabla \cdot \boldsymbol{\tau}_h \in S_h$. Application of the L_2 projection operator P_h defined in (3.3) to the above equation yields

$$(A_\delta \boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) = (P_h(f + \delta u_H), \nabla \cdot \boldsymbol{\tau}_h) = (-\mathbf{grad}_h(P_h(f + \delta u_H)), \boldsymbol{\tau}_h) \tag{3.4}$$

for all $\boldsymbol{\tau} \in \mathbf{V}_h$. Thus, we seek to solve

$$A_\delta \boldsymbol{\sigma}_h = -\mathbf{grad}_h(P_h(f + \delta u_H)).$$

Using (2.6) and (2.7), we obtain

$$A_\delta \boldsymbol{\sigma}_h = A\boldsymbol{\sigma}_h + (\delta - 1)\boldsymbol{\sigma}_h = -\mathbf{grad}_h(P_h(f + \delta u_H))$$

and it follows that

$$\boldsymbol{\sigma}_h = (1-\delta)A^{-1}\boldsymbol{\sigma}_h + A^{-1}(-\mathbf{grad}_h(P_h(f+\delta u_H))). \tag{3.5}$$

This expression suggests the iterative scheme

$$\boldsymbol{\sigma}_h^{n+1} = (1-\delta)A^{-1}\boldsymbol{\sigma}_h^{n} + A^{-1}(-\mathbf{grad}_h(f+\delta u_H)). \tag{3.6}$$

Theorem 3.1. *Let the sequence* $\{\boldsymbol{\sigma}_h^n\}_{n=0}^{\infty}$ *be defined by* (3.6). *Then*

$$\boldsymbol{\sigma}_h^n \to \boldsymbol{\sigma}_h \text{ as } n \to \infty.$$

Proof. It suffices to show that

$$(1-\delta)\|A^{-1}\| < 1. \tag{3.7}$$

Since the operator A is symmetric and positive definite with all eigenvalues larger than or equal to one, it follows that $\|A^{-1}\| \leq 1$. The inequality (3.7) now follows from the fact that $\delta > 0$. □

Remark 3.2. The above proof of convergence is based on the observation that $(1-\delta)\|A^{-1}\| \leq 1-\delta < 1$. However, the actual rate of convergence is much faster than indicated by the proof. This is due to the fact that $\mathbf{grad}_h S_h$ is invariant under A and A_δ, and $-\mathbf{grad}_h(P_h(f+\delta u_H)) \in \mathbf{grad}_h S_h$. Thus, the iterative procedure (3.6) only works in the subspace $\mathbf{grad}_h S_h$. We therefore may write this iterative procedure as

$$\boldsymbol{\sigma}_h^{n+1} = (1-\delta)A^{-1}_{|_{\mathbf{grad}_h S_h}}\boldsymbol{\sigma}_h^{n} + A^{-1}_{|_{\mathbf{grad}_h S_h}}(-\mathbf{grad}_h(f+\delta u_H)).$$

The eigenvalues of A corresponding to eigenfunctions in $\mathbf{grad}_h S_h$ are strictly larger than unity, while the eigenvalues of A corresponding to eigenfunctions in $\mathbf{curl}\, W_h$ are one. It follows that the convergence rate is bounded by the quantity $(1-\delta)\|A^{-1}_{|_{\mathbf{grad}_h S_h}}\|$, which is much smaller than unity. The iterations (3.6) therefore converge rapidly; see Tables 5.1 and 5.3 below. □

4. The linear system of algebraic equations

For simplicity, we will approximate $\boldsymbol{\sigma}$ using elements in the Raviart–Thomas space $\mathbf{V}_h$ of the lowest order $r = 0$. Let $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N$ be a (edge) basis for $\mathbf{V}_h$, i.e.,

$$\mathbf{V}_h = \text{span}\{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N\}, \tag{4.1}$$

where each function $\boldsymbol{\psi}_i$ is supported by two elements (triangles) having the edge E_i as a common side. The following definition of the basis functions is given in [23, Definition 4.3] and is helpful for the implementation of the functions.

Definition 4.1. Given an edge E, there are either two elements T_+ and T_- in $\mathcal{T}_h$ with the joint edge $E = \partial T_+ \bigcap \partial T_-$ or exactly one element T_+ in $\mathcal{T}_h$ with $E \subset \partial T_+$. Then, if $T_\pm = conv\{E \bigcup T_\pm\}$ for the vertex $P_\pm$ opposite to E of $T_\pm$, we have

$$\boldsymbol{\psi}_E(x) = \begin{cases} \pm\dfrac{|E|}{2|T|}(\mathbf{x} - P_\pm) & \text{for } \mathbf{x} \in T_\pm, \\ 0 & \text{otherwise.} \end{cases}$$

Any function $\mathbf{x} \in \mathbf{V}_h$ can be represented as

$$\mathbf{x} = x_1\boldsymbol{\psi}_1 + x_2\boldsymbol{\psi}_2 + \cdots + x_N\boldsymbol{\psi}_N.$$

Let $\vec{\mathbf{x}}$ be a vector representation of $\mathbf{x}$, i.e., $\vec{\mathbf{x}} = [x_1, \ldots, x_N]^T$. Now, using $\mathbf{V}_h$ as approximation space, the approximation $\boldsymbol{\sigma}_h$ of $\boldsymbol{\sigma}$ is defined as follows:

$$(\nabla\cdot\boldsymbol{\sigma}_h, \nabla\cdot\boldsymbol{\tau}_h) + \delta(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) = (q, \nabla\cdot\boldsymbol{\tau}_h)\forall\boldsymbol{\tau}_h \in \mathbf{V}_h, \tag{4.2}$$

where $q = f + \delta u_H^G$. Since $\boldsymbol{\sigma}_h$ can be represented as

$$\boldsymbol{\sigma}_h = \sigma_1\boldsymbol{\psi}_1 + \sigma_2\boldsymbol{\psi}_2 + \cdots \sigma_n\boldsymbol{\psi}_N,$$

and taking $\boldsymbol{\tau}_h = \boldsymbol{\psi}_i$, for $i = 1, 2, \ldots, N$ in (4.2), we obtain the linear system of equations:

$$\begin{aligned}
(d_{11}+\delta b_{11})\sigma_1 + (d_{12}+\delta b_{12})\sigma_2 + \cdots + (d_{1N}+\delta b_{1N})\sigma_N &= (q, \nabla\cdot\boldsymbol{\psi}_1) \\
(d_{21}+\delta b_{21})\sigma_1 + (d_{22}+\delta b_{22})\sigma_2 + \cdots + (d_{2N}+\delta b_{2N})\sigma_N &= (q, \nabla\cdot\boldsymbol{\psi}_2) \\
&\cdots \\
(d_{N1}+\delta b_{N1})\sigma_1 + (d_{N2}+\delta b_{N2})\sigma_2 + \cdots + (d_{NN}+\delta b_{NN})\sigma_N &= (q, \nabla\cdot\boldsymbol{\psi}_N),
\end{aligned} \tag{4.3}$$

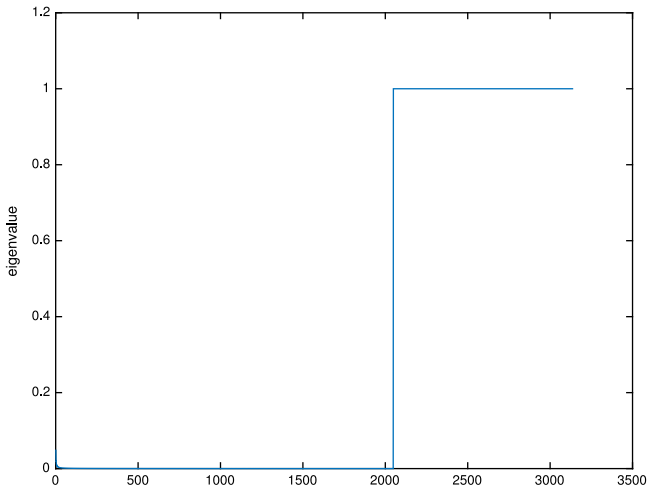

Fig. 4.1. Distribution of eigenvalues of the matrix $S^{-1}B$ for a uniform mesh on the unit square with mesh size $h = \frac{1}{2^5}$. The matrix is of order 3136.

where the (i, j)th elements of b_{ij} and d_{ij} are given by

$$b_{ij} = (\boldsymbol{\psi}_j, \boldsymbol{\psi}_i) \text{ and } d_{ij} = (\nabla \cdot \boldsymbol{\psi}_j, \nabla \cdot \boldsymbol{\psi}_i).$$

Let the $N \times N$ matrices B and D be defined by

$$B = [b_{ij}] \quad \text{and} \quad D = [d_{ij}],$$

and $\vec{\mathbf{q}} = [q_1, q_2, \ldots, q_N]$, where $q_i = (q, \nabla \cdot \boldsymbol{\psi}_i)$ for $i = 1, 2, \ldots, N$. Then, using (4.3), the approximate solution $\boldsymbol{\sigma}_h$ is obtained by solving the following system of linear equations:

$$(D + \delta B)\vec{\boldsymbol{\sigma}}_h = \vec{\mathbf{q}}. \tag{4.4}$$

4.1. The iterative method revisited

The matrix $D + \delta B$ in the linear system of algebraic equations (4.4) is nearly singular for small positive values of δ. As mentioned above, it is desirable to let δ be small so that the flux can be computed with high accuracy and moderate effort. To overcome the difficulty of solving the near-singular linear system of algebraic equation (4.4), we proceed as follows. First, set $S = D + B$. Then the matrix S is symmetric positive definite and well-conditioned. We write the system (4.4) as

$$(D + \delta B)\vec{\boldsymbol{\sigma}} = (D + B)\vec{\boldsymbol{\sigma}} + (\delta - 1)B\vec{\boldsymbol{\sigma}} = S\vec{\boldsymbol{\sigma}} + (\delta - 1)B\vec{\boldsymbol{\sigma}} = \vec{\mathbf{q}}.$$

Hence, the solution of (4.4) satisfies

$$\vec{\boldsymbol{\sigma}} = (1 - \delta)S^{-1}B\vec{\boldsymbol{\sigma}} + S^{-1}\vec{\mathbf{q}}.$$

This equation suggests the iterative scheme

$$\vec{\boldsymbol{\sigma}}_{n+1} = M\vec{\boldsymbol{\sigma}}_n + \vec{\mathbf{g}}, \tag{4.5}$$

where $M = (1 - \delta)S^{-1}B$ and $\vec{\mathbf{g}} = S^{-1}\vec{\mathbf{q}}$. Eq. (4.5) is the matrix representation of (3.6) with respect to the basis (4.1). Convergence is established by Theorem 3.1. We comment on the computations required to carry out the iterations below.

Remark 4.2. The solution $\boldsymbol{\sigma}_h$ of (3.5) lives in $\mathbf{grad}_h S_h$ and this subspace is invariant under the operator A. All eigenvalues of the matrix representation of A^{-1}, i.e. of $S^{-1}B$, are in the interval $(0, 1]$. The eigenvalues of A^{-1} restricted to $\mathbf{grad}_h S_h$ are smaller than unity, typically much smaller, while the eigenvalues of the restriction of A^{-1} to $\mathbf{curl}\, W_h$ are one. The matrix M only acts on $\mathbf{grad}_h S_h$. Therefore, the iterations (4.5) converge quickly; see also Remark 3.2. The distribution of the eigenvalues of $S^{-1}B$ for a typical situation is displayed in Fig. 4.1. □

We conclude this section with some comments on the computational effort required for each iteration (4.5). When the matrix S is of small to moderate size, we can determine its (sparse) Cholesky factorization. Each iteration then requires a backward and forward solve using the Cholesky factors. This is quite inexpensive. Large-scale problems, for which it is not attractive to compute the Cholesky factorization of S, can be solved by an iterative method, such as a preconditioned conjugate gradient method or a multigrid method. It may be attractive to use an incomplete Cholesky factorization as a preconditioner for the conjugate gradient method.

Table 5.1
Performance for different δ values with $h = \frac{1}{128}$.

δ	h^2	h^4	h^5	h^6	h^8	h^{10}
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^C\|$	0.0012	0.0013	0.0449	–	–	–
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^M\|$	0.0012	0.0012	0.0476	2.0919e+04	2.0752e+04	2.0752e+04
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^D\|$	0.0012	0.0012	0.0724	4.0891	4.3579	4.3579
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^I\|$	0.0012	0.0012	0.0012	0.0012	0.0012	0.0012
# of iterations	8	8	8	8	8	8

Table 5.2
Difference between smallest eigenvalue of operator A restricted to the range of the gradient and smallest eigenvalue of the corresponding restriction of the matrix A.

h	$1/2^4$	$1/2^5$	$1/2^6$
$\lvert\lambda_{\min} - \lambda_{\min,h}\rvert$	$4.83 \cdot 10^{-5}$	$1.23 \cdot 10^{-5}$	$3.07 \cdot 10^{-6}$

5. Numerical experiments

This section provides numerical examples that show the performance of the iterative method (4.5) for nearly singular systems (4.4) to obtain approximation solution $\boldsymbol{\sigma}_h$ for the solution of $\boldsymbol{\sigma}$ of the following model problem:

$$(\nabla \cdot \boldsymbol{\sigma}, \nabla \cdot \boldsymbol{\tau}) + \delta(\boldsymbol{\sigma}, \boldsymbol{\tau}) = (q, \nabla \cdot \boldsymbol{\tau}), \text{ on } \Omega. \tag{5.1}$$

We use the lowest-order Raviart–Thomas space RT_0 in our computations and apply the iterative method (4.5) to determine an approximation of the solution $\boldsymbol{\sigma}$.

We terminate the iterations (4.5) when the relative difference between two consecutive iterates is less than 10^{-10}, i.e. when

$$\frac{\|\vec{\boldsymbol{\sigma}}_n - \vec{\boldsymbol{\sigma}}_{n+1}\|}{\|\vec{\boldsymbol{\sigma}}_n\|} \leq 10^{-10}. \tag{5.2}$$

Here $\|\cdot\|$ denotes the Euclidean vector norm.

We compare the iterative method (4.5) to several other solution methods that are easy to implement, including direct solution of (4.4) using the MATLAB backslash command (\), which computes a Cholesky or LU factorization of the matrix $D + \delta B$ in (4.4), and using the MATLAB command 'chol', which computes the Cholesky factorization of the same matrix. Since the matrix $D + \delta B$ is symmetric positive definite, the backslash command should compute the Cholesky factorization and both solution methods should give the same computed solution. However, as we will see below, they do not. We also compare with the use of an algebraic multigrid method (AMG). Specifically, we apply the AMG solver developed and made available by Notay [16].

5.1. Example on a square

We solve the model problem (5.1) on $\Omega = [0,\ 1] \times [0,\ 1]$ with exact solution $\boldsymbol{\sigma} = [(1-2x)(y-y^2), (x-x^2)(1-2y)]^T$. To illustrate the performance of the iterative procedure (4.5) when the system (4.4) is nearly singular, we let $\delta = h^2, h^4, \ldots, h^{10}$, where $h = 1/128$ is the mesh size. The number of degrees of freedom is $2h^{-1}(h^{-1}+1)+h^{-2}$, which is 49 408. The matrix M in (4.5) therefore is of size $49\,408 \times 49\,408$, which is small enough to allow the computation of its sparse Cholesky factorization in MATLAB.

In Table 5.1, $\boldsymbol{\sigma}$ denotes the exact solution of the discrete problem, $\boldsymbol{\sigma}_h^I$ is the solution determined by the iterative method (4.5), $\boldsymbol{\sigma}_h^D$ stands for the solution computed by the MATLAB direct solver \, and $\boldsymbol{\sigma}_h^M$ and $\boldsymbol{\sigma}_h^C$ denote the solution obtained by the AMG method and Cholesky factorization, respectively. The table shows the iterative method (4.5) to produce approximate solutions that are close to the exact solution, while the other methods are seen to yield poor accuracy when δ is small. When the MATLAB function 'chol' for the Cholesky factorization of the matrix $D + \delta B$ in (4.4) is applied for small values of $\delta > 0$, it does not recognize the matrix as positive definite and does not provide a solution. This happens for $\delta \leq h^6$.

Table 5.1 shows the iterative method (4.5) to require 8 iterations to achieve the desired accuracy for all $\delta = h^2, \ldots, h^{10}$. We recall that the quantity $(1-\delta)\|A^{-1}_{|\mathbf{grad}_h S_h}\|$ bounds the rate of convergence of the iterative method. For our model problem, the smallest eigenvalue of the operator A restricted to the subspace $\mathbf{grad}\, S$ is $\lambda_{\min} = 1 + 2\pi^2$. An associated eigenfunction is $(\cos \pi x \sin \pi y, \sin \pi x \cos \pi y)$. The smallest eigenvalue, $\lambda_{\min,h}$, of the restriction of the matrix A to $\mathbf{grad}_h S_h$ approximates $\lambda_{\min}$. The difference $|\lambda_{\min} - \lambda_{\min,h}|$ decreases as h decreases; see Table 5.2. Thus,

$$(1-\delta)\|A^{-1}_{|\mathbf{grad}_h S_h}\| \sim (1-\delta)\frac{1}{1+2\pi^2} \sim 0.0482$$

Table 5.3
Number of iteration as a function of δ with $h = \frac{1}{128}$.

δ	1	0.99999	0.99	0.9	0.5	0.1
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^l\|$	0.0012	0.0012	0.0012	0.0012	0.0012	0.0012
# of iterations	1	2	4	5	7	8

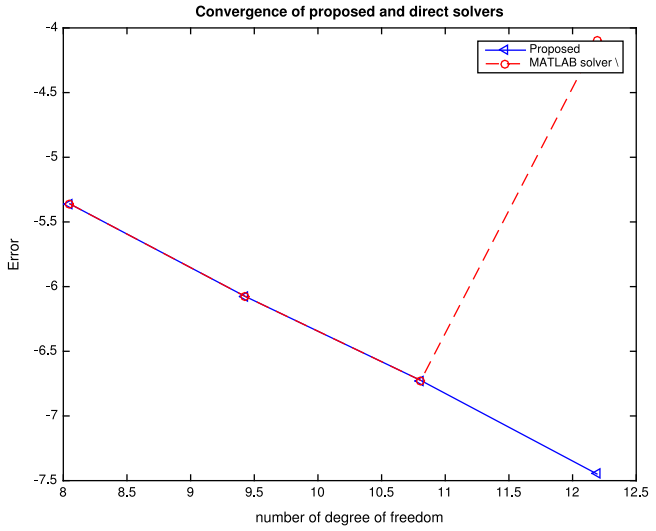

Fig. 5.1. log-log plot of convergence behavior for mesh sizes $h = 1/2^5, \ldots, 1/2^8$ with $\delta = h^4$.

for tiny $\delta > 0$. This explains why 8 iterations are required for all values of h, because $0.0482^8 < 10^{-10} < 0.0482^7$, where 10^{-10} is the stopping criterion (5.2).

Table 5.3 shows how the number of iterations required to achieve the desired accuracy for $\delta = 1, 0.99999, 0.99, 0.5, 0.1$ depends on δ. For all values of δ, the number of iterations to achieve the desired accuracy satisfies

$$((1-\delta)0.0482)^{N(\delta)} < 10^{-10} < ((1-\delta)0.0482)^{N(\delta)-1},$$

where $N(\delta)$ denotes the number of iterations as a function of δ.

Fig. 5.1 shows the convergence behavior of the errors $\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^D\|$ and $\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^l\|$ for $\delta = h^4$ and decreasing $h = 1/2^5, 1/2^6, 1/2^7, 1/2^8$, where $\boldsymbol{\sigma}_h^D$ and $\boldsymbol{\sigma}_h^l$ are the approximate solutions determined by the MATLAB command \ and our iterative procedure, respectively. The degrees of freedom of Fig. 5.1 are 3136, 12416, 49408, and 197120. The direct MATLAB solver is seen not to be able to deal with the nearly singular systems obtained for small h values and produces a solution with a large error, while our iterative method determines accurate approximate solutions. We also considered mesh sizes both larger and smaller than $h = 1/128$. The number of iterations required to satisfy (5.2) for these mesh sizes was the same as for $h = 1/128$.

5.2. Example on a disk

We solve the model problem (5.1) on the unit disk $\Omega = \{(x, y) : x^2 + y^2 \leq 1\}$ with exact solution

$$\boldsymbol{\sigma} = [\frac{x}{\sqrt{x^2+y^2}}, \frac{y}{\sqrt{x^2+y^2}}]^T = -\nabla(1 - \sqrt{x^2+y^2}).$$

For the generation of meshes on the disk, we use the mesh generator presented in [25]. The matrix of the resulting linear system of equations is symmetric positive definite.

We let $\delta = h^2, h^4, \ldots, h^{10}$, where $h = \frac{1}{64}$ is the mesh size, and show the convergence behavior of several solvers in Table 5.4 for different δ-values. Moreover, Fig. 5.2 displays the convergence of our proposed method and the direct MATLAB solver (\) for $h = 1/2^3, \ldots, 1/2^6$ with $\delta = h^6$. Our proposed method shows superior performance.

5.3. Extrapolation

Extrapolation is a popular approach to speed up the convergence of a slowly converging sequence. Vector extrapolation methods with many applications are discussed in [11–14]. The reduced rank extrapolation method (RRE) is a particularly

Table 5.4
Performance for different δ values with $h = \frac{1}{64}$.

δ	1	h^2	h^4	h^6	h^8	h^{10}
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^C\|$	0.0359	0.0361	0.0361	0.0362	–	–
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^M\|$	0.0359	0.1274	0.1279	0.1279	50.4845	80.5983
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^D\|$	0.0359	0.0361	0.0361	0.0361	38.0661	207.4175
$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h^I\|$	0.0359	0.0361	0.0361	0.0361	0.0361	0.0361
# of iterations	1	11	11	11	11	11

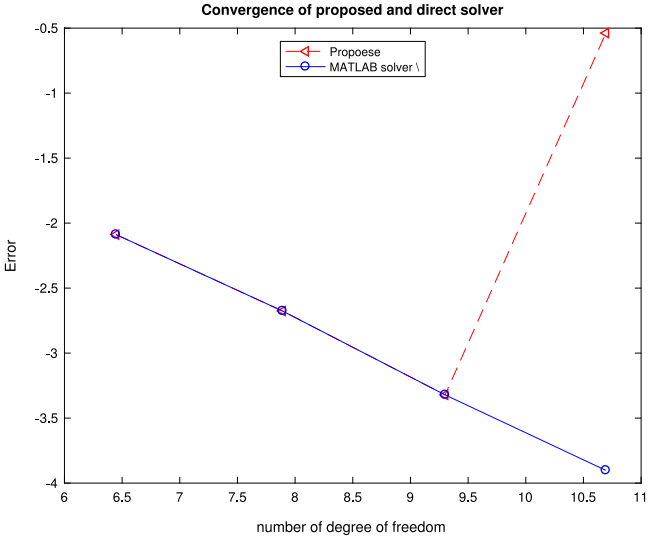

Fig. 5.2. log-log plot of convergence behavior for mesh sizes $h = 1/2^3, \ldots, 1/2^6$ with $\delta = h^6$.

effective vector extrapolation method. We outline this method following [11]. Introduce the first and second differences

$$\Delta\vec{\boldsymbol{\sigma}}_n = \vec{\boldsymbol{\sigma}}_{n+1} - \vec{\boldsymbol{\sigma}}_n, \quad \Delta^2\vec{\boldsymbol{\sigma}}_n = \Delta\vec{\boldsymbol{\sigma}}_{n+1} - \Delta\vec{\boldsymbol{\sigma}}_n, \quad n = 0, 1, 2, \ldots,$$

of the iterates (4.5). The RRE method produces approximations of the form

$$\vec{\mathbf{s}}_{k,q} = \sum_{j=0}^{q} \gamma_j^{(q)} \vec{\boldsymbol{\sigma}}_{k+j} \tag{5.3}$$

of the limit $\vec{\boldsymbol{\sigma}}_\infty$ of the sequence $\vec{\boldsymbol{\sigma}}_0, \vec{\boldsymbol{\sigma}}_1, \vec{\boldsymbol{\sigma}}_2, \ldots$, where

$$\sum_{j=0}^{q} \gamma_j^{(q)} = 1, \qquad \sum_{j=0}^{q} \eta_{ij}^{(k)} \gamma_j^{(q)} = 0, \quad i = 0, 1, \ldots, q-1,$$

with $\eta_{ij}^{(k)} = [\Delta^2\vec{\boldsymbol{\sigma}}_{k+i}, \Delta\vec{\boldsymbol{\sigma}}_{k+j}]$.

Define the matrices $\Delta^i S_{k,q} = [\Delta^i\vec{\boldsymbol{\sigma}}_k, \ldots, \Delta^i\vec{\boldsymbol{\sigma}}_{k+q-1}]$ for $i = 1, 2$. One can express the approximation (5.3) as

$$\vec{\mathbf{s}}_{k,q} = \vec{\boldsymbol{\sigma}}_k - \Delta S_{k,q} \Delta^2 S_{k,q}^{\dagger} \Delta\vec{\boldsymbol{\sigma}}_k,$$

where $\Delta^2 S_{k,q}^{\dagger}$ denotes the Moore–Penrose pseudoinverse of the matrix $\Delta^2 S_{k,q}$; see, e.g., [11] for details. The vectors $\vec{\mathbf{s}}_{k,q}$ can be computed efficiently for several values of k and q by algorithms described in [11,13].

We applied RRE to the iterates $\vec{\boldsymbol{\sigma}}_n$ defined by (4.5) to speed up their convergence towards the limit $\vec{\boldsymbol{\sigma}}_\infty$. Specifically, we applied (5.3) with $k = 0$ and terminated the extrapolation process as soon as an analogue of (5.2) is satisfied for the extrapolated vectors. We found that it was sufficient to carry out 6 iterations (4.5) independently of the size of $\delta > 0$ and for several mesh sizes. Thus, when the vectors $\vec{\boldsymbol{\sigma}}_n$ have many components and are expensive to compute, the application of RRE may be beneficial.

6. Conclusion

The paper describes a simple iterative method that allows fast and accurate solution of linear systems of algebraic equations that arise from the discretization of variational problems of the form (1.1). A computed example illustrates the good performance of the method. A reduction in the number of iterations required to satisfy the stopping criterion (5.2) can be achieved by the application of reduced rank extrapolation.

Acknowledgments

We would like to thank Maya Neytcheva for helpful suggestions. The second author was supported in part by NSF, United States grants DMS-1729509 and DMS-1720259.

References

[1] D.N. Arnold, R.S. Falk, R. Winther, Preconditioning in $H(div)$ and applications, Math. Comp. 66 (1997) 957–984.
[2] R. Hiptmair, J. Xu, Nodal auxiliary space preconditioning in $H(curl)$ and $H(div)$ spaces, SIAM J. Numer. Anal. 45 (2007) 2483–2509.
[3] J. Ku, Y.J. Lee, D. Sheen, A hybrid two-step finite element method for flux approximation: a priori and a posteriori estimates, ESAIM Math. Model. Numer. Anal. (ESAIM: M2AN) 51 (2017) 1303–1316.
[4] P.N. Brown, H.F. Walker, GMRES on (nearly) singular systems, SIAM J. Matrix Anal. Appl. 18 (1997) 37–51.
[5] M. Eiermann, I. Marek, W. Niethammer, On the solution of singular linear systems of algebraic equations by semiiterative methods, Numer. Math. 53 (1988) 265–283.
[6] W.W. Hager, Iterative methods for nearly singular linear systems, SIAM J. Sci. Comput. 22 (2000) 747–766.
[7] K. Hayami, M. Sugihara, A geometric view of Krylov subspace methods on singular systems, Numer. Linear Algebra Appl. 18 (2011) 449–460 and 21 (2014), pp. 701–702.
[8] L. Reichel, Q. Ye, Breakdown-free GMRES for singular systems, SIAM J. Matrix Anal. Appl. 26 (2005) 1001–1021.
[9] S. Zampini, PCBDDC: a class of robust dual-primal methods in PETSc, SIAM J. Sci. Comput. 38 (2016) S282–S306.
[10] C. Brezinski, M. Redivo Zaglia, Extrapolation Methods: Theory and Practice, North-Holland, Amsterdam, 1991.
[11] S. Duminil, H. Sadok, Reduced rank extrapolation applied to electronic structure computations, Electron. Trans. Numer. Anal. 38 (2011) 347–362.
[12] R. El-Moallem, H. Sadok, Vector extrapolation applied to algebraic Riccati equations arising in transport theory, Electron. Trans. Numer. Anal. 40 (2013) 489–506.
[13] W.D. Ford, A. Sidi, Recursive algorithms for vector extrapolation methods, Appl. Numer. Math. 4 (1988) 477–489.
[14] K. Jbilou, H. Sadok, Vector extrapolation methods. Applications and numerical comparison, J. Comput. Appl. Math. 122 (2000) 149–165.
[15] A. Napov, Y. Notay, An algebraic multigrid method with guaranteed convergence rate, SIAM J. Sci. Comput. 34 (2012) A1079–A1109.
[16] Y. Notay, AGMG software and documentation, available at http://homepages.ulb.ac.be/~ynotay/AGMG.
[17] Y. Notay, An aggregation-based algebraic multigrid method, Electron. Trans. Numer. Anal. 37 (2010) 123–146.
[18] Y. Notay, Aggregation-based algebraic multigrid for convection–diffusion equations, SIAM J. Sci. Comput. 34 (2012) A2288–A2316.
[19] P.A. Raviart, J.M. Thomas, A mixed finite element method for second order elliptic problems, in: L.I. Galligani, E. Magenes (Eds.), Mathematical Aspects of the Finite Element Method, in: Lecture Notes in Math., # 606, Springer, New York, 1977, pp. 292–315.
[20] F. Brezzi, J. Douglas Jr., L.D. Marini, Two families of mixed finite elements for second order elliptic problems, Numer. Math. 47 (1985) 217–235.
[21] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods, Springer, New York, 2008.
[22] F. Brezzi, M. Fortin, Mixed and Hybrid Finite Element Methods, Springer, Berlin, 1991.
[23] C. Bahriawati, C. Carstensen, Three MATLAB implementations of the lowest-order Raviart–Thomas MFEM with a posteriori control, Comput. Methods Appl. Math. 5 (2005) 333–361.
[24] F. Brezzi, M. Fortin, R. Stenberg, Error analysis of mixed-interpolated elements for Reissner–Mindlin plates, Math. Models Methods Appl. Sci. 1 (1991) 125–151.
[25] P.O. Persson, G. Strang, Simple mesh generator in MATLAB, available at http://persson.berkeley.edu/distmesh/.