Effective matrix-free preconditioning for the augmented immersed interface method[☆]

Jianlin Xia[a], Zhilin Li[b], Xin Ye[a]

[a] *Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA. E-mail: xiaj@math.purdue.edu, ye83@purdue.edu.*

[b] *Center for Research in Scientific Computation & Department of Mathematics, North Carolina State University, Raleigh, NC 27695, USA. E-mail: zhilin@math.ncsu.edu.*

Abstract

We present effective and efficient matrix-free preconditioning techniques for the augmented immersed interface method (AIIM). AIIM has been developed recently and is shown to be very effective for interface problems and problems on irregular domains. GMRES is often used to solve for the augmented variable(s) associated with a Schur complement $\mathbf{A}$ in AIIM that is defined along the interface or the irregular boundary. The efficiency of AIIM relies on how quickly the system for $\mathbf{A}$ can be solved. For some applications, there are substantial difficulties involved, such as the slow convergence of GMRES (particularly for free boundary and moving interface problems), and the inconvenience in finding a preconditioner (due to the situation that only the products of $\mathbf{A}$ and vectors are available). Here, we propose matrix-free structured preconditioning techniques for AIIM via adaptive randomized sampling, using only the products of $\mathbf{A}$ and vectors to construct a hierarchically semiseparable matrix approximation to $\mathbf{A}$. Several improvements over existing schemes are shown so as to enhance the efficiency and also avoid potential instability. The significance of the preconditioners includes: (1) they do not require the entries of $\mathbf{A}$ or the multiplication of $\mathbf{A}^T$ with vectors; (2) constructing the preconditioners needs only $O(\log N)$ matrix-vector products and $O(N)$ storage, where N is the size of $\mathbf{A}$; (3) applying the preconditioners needs only $O(N)$ flops; (4) they are very flexible and do not require any *a priori* knowledge of the structure of $\mathbf{A}$. The preconditioners are observed to significantly accelerate the convergence of GMRES, with heuristical justifications of the effectiveness. Comprehensive tests on several important applications are provided, such as Navier-Stokes equations on irregular domains with traction boundary conditions, interface problems in incompressible flows, mixed boundary problems, and free boundary problems. The preconditioning techniques are also useful for several other problems and methods.

Key words: augmented immersed interface method, Schur complement system, GMRES, matrix-free preconditioning, hierarchically semiseparable structure, adaptive randomized compression

---

[☆]The research of Jianlin Xia was supported in part by an NSF CAREER Award DMS-1255416 and an NSF grant DMS-1115572. The research of Zhilin Li was partially supported by an NIH grant 5R01GM96195-2, an NSF grant DMS-1522768, and an NCSU Research Innovation Seed Fund.

1. Introduction

In recent years, the augmented immersed interface method (AIIM) has been shown to be very effective for the solution of many interface problems and problems on irregular domains. The method is first developed for elliptic interface problems with discontinuous and piecewise constant coefficients [19]. Later, the idea is extended to moving interface problems on irregular domains [26] and incompressible Stokes equations with discontinuous viscosities [22]. We refer the readers to [21] for more information about AIIM.

Augmented strategies can be naturally used to design efficient and accurate algorithms based on existing fast solvers. As an example, for incompressible Stokes equations with a discontinuous viscosity across the interface, the discontinuous pressure and velocity can be decoupled by augmented strategies. The immersed interface method can then be applied conveniently with a fast Poisson solver in the iterative solution of the Schur complement system.

In augmented strategies, a large linear system is formed for the approximate solution $\mathbf{u}$ to the original problem together with an augmented variable $\mathbf{g}$ (which may be a vector) of co-dimension one. Eliminating the block corresponding to $\mathbf{u}$ from the coefficient matrix yields a Schur complement system for $\mathbf{g}$, which is often much smaller compared with $\mathbf{u}$. Finding $\mathbf{g}$ can then make it convenient to solve for $\mathbf{u}$.

Thus, it is critical to quickly solve the Schur complement system, which can be done via direct or iterative solvers. Direct solvers can be used for some applications when the Schur complement matrix $\mathbf{A}$ is a constant matrix, for example, for a fixed interface or boundary. If $\mathbf{A}$ is not a constant matrix, typically for free boundary and moving interface problems, efficient iterative solvers may be preferred. Iterative solvers such as GMRES [37] do not require the explicit formation of the Schur complement matrix $\mathbf{A}$, and are thus often used. In addition, the solution often needs just modest accuracies, and iterative methods make it convenient to control the accuracy and the cost. Iterative solvers only need the product of $\mathbf{A}$ and vectors. This is usually done as follows. First, assume $\mathbf{g}$ is available and solve the original problem for an approximate solution $\mathbf{u}$. Next, use the approximate solution to compute the residual and thus the matrix-vector product.

AIIM can be considered as a generalized boundary integral method without explicitly using Green functions. The augmented variable can be considered as a source term. If the system behaves like an integral equation of the first kind, then GMRES converges slowly in general as described in several applications in this paper. For these applications, effective preconditioners are then needed.

Our work here is initially motivated by the application of AIIM to Navier-Stokes equations with a traction boundary condition. Explicit numerical tests indicate that, for some applications, the condition number of the Schur complement matrix $\mathbf{A}$ is of size $O(1)$, and $\|\mathbf{A}^T\mathbf{A} - \mathbf{A}\mathbf{A}^T\|_2$ is also very small. Nevertheless, GMRES still either takes too many steps to converge or simply stalls. Finding a preconditioner that can greatly improve the convergence of GMRES and is easy to apply becomes crucial for augmented methods.

However, there are some substantial difficulties in constructing suitable preconditioners. In fact, it may be difficult to even obtain a simple preconditioner such as the diagonal of $\mathbf{A}$. The reasons are: (1) $\mathbf{A}$ is generally dense and the entries are not known explicitly in augmented methods, as mentioned above; (2) we can quickly multiply $\mathbf{A}$ and vectors, but not $\mathbf{A}^T$ and vectors. Preliminary attempts have been made, such as extracting few columns of $\mathbf{A}$ or multiplying $\mathbf{A}$ with special vectors, and then converting the results into a block diagonal preconditioner. The preconditioner may work for a particular problem or right-hand side, but often fails. Thus, it is necessary to design reliable *matrix-free* preconditioning techniques based on only matrix-vector products. Previously, some matrix-free preconditioners are designed for certain sparse problems and specific applications [2, 4, 7, 10, 31]. The ideas are to extract partial approximations or to compute incomplete factorizations via matrix-vector multiplications.

The objective of this work is to construct effective and efficient *structured matrix-free* preconditioners solely based on the products of $\mathbf{A}$ and vectors. This is achieved by taking advantage of some new preconditioning techniques introduced in recent years, such as rank structured preconditioning [5, 8, 11, 14, 18, 47] and randomized preconditioning [33, 34].

The idea of rank structured preconditioning is to obtain a preconditioner via the truncation of the singular values of appropriate off-diagonal blocks. If the off-diagonal singular values decay quickly (the problem is then said to have a *low-rank property*), it is known that this truncation strategy can be used to develop fast approximate direct solvers. On the other hand, if the off-diagonal singular values are aggressively truncated regardless of their decay rate, structured preconditioners can be obtained. The effectiveness is studied for some cases in [18, 47]. Among the most frequently used rank structures is the hierarchically semiseparable (HSS) form [6, 46]. Unlike standard dense matrix operations, structured methods in terms of compact HSS forms can achieve significantly better efficiency. In fact, the factorization and solution of an HSS matrix need only about $O(N)$ flops and $O(N)$ storage, where N is the matrix size.

In some latest developments, randomized sampling is combined with rank structures to obtain enhanced flexibility [28, 32, 42, 48]. That is, the construction of rank structures (e.g., HSS) may potentially use only matrix-vector products instead of the original matrix itself. The methods in [32, 48] use both matrix-vector products and selected entries of the matrix. The one in [28] is matrix-free, although requiring slightly more matrix-vector products. In [42], a fully matrix-free and adaptive HSS construction scheme is developed. It uses an adaptive rank detection strategy in [15] to dynamically decide the off-diagonal ranks based on a pre-specified accuracy.

However, all the randomized methods in [28, 32, 42, 48] require the products of both $\mathbf{A}$ and $\mathbf{A}^T$ with vectors if $\mathbf{A}$ is nonsymmetric, and are thus not applicable to AIIM. Here, we seek to precondition $\mathbf{A}$ with an improved adaptive matrix-free scheme, using only the products of $\mathbf{A}$ and vectors. Due to the special features of AIIM as mentioned above, we construct a nearly symmetric HSS approximation H to $\mathbf{A}$ via randomized sampling. In the construction, $\mathbf{A}$ replaces $\mathbf{A}^T$ for the multiplication of $\mathbf{A}^T$ and vectors. Thus, the column basis of an off-diagonal block of $\mathbf{A}$ obtained by randomized sampling is used to approximate the row basis of the off-diagonal block at the symmetric position of $\mathbf{A}$. This enables us to find low-rank approximations to all the off-diagonal blocks. We explicitly specify a small ($O(1)$) rank or a low accuracy in the approximation. The off-diagonal approximations are then combined with the matrix-vector products to yield approximations of the diagonal blocks. This is done in a hierarchical fashion so that the overall HSS construction process needs only $O(\log N)$ matrix-vector products.

Several other improvements to the schemes in [28, 42] are made. We replace half of the randomized sampling by deterministic QR factorizations. We also design a strategy to reduce the number of matrix multiplications and avoid the use of pseudoinverses that are both expensive and potentially unstable.

H is then quickly factorized, and the factors are used as a preconditioner. Existing HSS algorithms are simplified to take advantage of the near symmetry, and the factorization and the application of the preconditioner have only $O(N)$ complexity and $O(N)$ storage. Since $\mathbf{A}$ corresponds to the interface, its size N is much smaller than the Poisson solution needed to compute a matrix-vector product. Thus, the preconditioning cost is negligible as compared with the matrix-vector multiplication cost in the iterations. We also provide a simplified preconditioner given by the diagonal blocks of H that is easier to use and sometimes has comparable performance.

The effectiveness of the preconditioners is discussed in terms of several aspects of structured and randomized preconditioning, such as the benefits of low-accuracy rank structured preconditioners in reducing condition numbers. The preconditioners also share some ideas with the additive preconditioning techniques in [33, 34], where random low-rank matrices are added to the original matrix to provide effective preconditioners. In addition, since a low-accuracy HSS approximation tends to preserve well-separated eigenvalues [39], it can bring the eigenvalues together when used as a preconditioner. Although the convergence of GMRES does not necessarily rely on the eigenvalue distribution [12, 38], the eigenvalue redistribution provides an empirical explanation for the preconditioner, as used in practice.

The effectiveness and the efficiency are further demonstrated with a survey of several important applications, such as Navier-Stokes equations on irregular domains with traction boundary conditions, an incompressible interface in incompressible flow, a contact problem of drop spreading, and a mixed boundary problem. For the Schur complement matrix $\mathbf{A}$ in AIIM, GMRES (with restart) generally stalls. Even non-restarted GMRES barely converges unless the number of iterations reaches nearly N. However, after our matrix-free preconditioning, GMRES converges quickly. The convergence is also observed to be relatively insensitive to N and some physical parameters.

We also give a comprehensive test for a free boundary problem that involves multiple stages of GMRES solutions within the iterative solution of a nonlinear equation. The problem involves mixed boundary conditions, and the system behaves like integral equations of both first and second kinds. With our preconditioner, the overall performance GMRES solutions for all the stages is significantly improved.

The presentation is organized as follows. In Section 2, the features of the Schur complements and the motivation for our work are discussed, together with a brief review of the idea of AIIM. In particular, the linear system with the Schur complement matrix $\mathbf{A}$ is explained in detail, including the fast multiplication of $\mathbf{A}$ with vectors and the formation of the right-hand side. Section 3 presents the matrix-free structured preconditioner and the detailed algorithms, and the effectiveness is discussed. Section 4 lists the applications, summarizes the numerical tests, and discusses the generalizations. Some conclusions are drawn in Section 5.

2. Features of the Schur complement systems in AIIM and motivation for the work

AIIM usually has two discretizations. One is for the governing PDE with the assumption that the augmented variable is known. The second is for the augmented equation such as the boundary condition or the interface condition. In this section, as a preparation for our preconditioning techniques, we use the example of solving a Poisson equation on an irregular domain in [19, 21] to demonstrate the idea of AIIM, and show the form of the Schur complement $\mathbf{A}$ and its multiplication with vectors. We then discuss some useful features of the Schur complement system. These features are based on both mathematical and numerical observations, and provide a motivation for the development of our preconditioners.

2.1. Finite difference method for elliptic problems with singular source terms

In this subsection, we review the discretization of the governing PDE as in [19, 21], assuming the augmented variable is known. Let $\mathbf{R} = \{(x, y),\ a < x < b,\ c < y < d\}$ be a rectangular domain. Consider an elliptic interface problem with a specified boundary condition on $\partial\mathbf{R}$:

$$\begin{gathered} \Delta u = f(x,y), \quad (x,y) \in \mathbf{R} - \partial\Omega, \\ [u] = w, \quad [u_n] = v, \end{gathered} \tag{1}$$

where $\partial\Omega = (\hat{x}(s),\ \hat{y}(s))$ is a curve or interface within $\mathbf{R}$ with a parameter s (such as the arc-length), $[u]$ is the jump of the solution across the boundary $\partial\Omega$, n is the unit direction pointing outward of $\partial\Omega$, and $[u_n] = [\nabla u \cdot n]$ is the jump in the normal derivative of u across $\partial\Omega$. If $w \equiv 0$, (1) can be rewritten as Peskin's model [35]

$$\Delta u = f(x,y) + \int_{\partial\Omega} v(s)\delta(x - \hat{x}(s))\delta(y - \hat{y}(s))\, ds.$$

We have a single equation for the entire domain. The second term on the right-hand side involves the two-dimensional Dirac delta function, which is called a singular source term or a source distribution along the curve $\partial\Omega$. If $w \neq 0$, then it is called a double layer, similar to the derivative of the Dirac delta function, which is again called a singular source. To solve the equation above numerically, either the immersed boundary method or the immersed interface method (IIM) can be used. For example, following the standard five-point finite difference discretization on a uniform mesh, we can write both methods as

$$\frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i+1,j} - 4u_{i,j}}{h^2} = f_{ij} + c_{ij},$$

where h is the uniform mesh size, $u_{ij} \approx u(x_i, y_j)$ is the discrete solution, $f_{ij} = f(x_i, y_j)$, and c_{ij} is the discrete delta function in the immersed boundary method or is chosen to achieve the second order accuracy in IIM. A matrix-vector form can be conveniently written for the scheme:

$$A\mathbf{u} = \mathbf{f} + B\mathbf{w} + C\mathbf{v},$$

where $\mathbf{w}$ and $\mathbf{v}$ are the discrete forms of w and v, respectively. The matrices B and C are sparse matrices that are related to the coordinates of grid points in the finite difference stencil and the boundary information including the first and the second order partial derivatives of $\partial\Omega$. Usually, each row of B or C has $3 \sim 9$ nonzero entries, depending on the applications.

Here, we assume that we know w and v on a rectangular domain. For AIIM, one of them is unknown and should be chosen to satisfy a certain kind of interface/boundary conditions for different applications.

2.2. AIIM for Helmoltz/Poisson equations on irregular domains

Now we explain AIIM for the solution of Helmholtz/Poisson equations on an irregular interior or exterior domain Ω. See [16, 17, 26] for more details. Consider an Helmholtz/Poisson equation with a linear boundary condition $q(u, u_n)$ along the boundary $\partial\Omega$:

$$\begin{aligned} \Delta u - \omega u &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ q(u, u_n) &= 0, \quad \mathbf{x} \in \partial\Omega. \end{aligned}$$

In AIIM, Ω is embedded into a rectangle or cube domain $\mathbf{R}$, and $\partial\Omega$ becomes an interface. The PDE and the source term are then extended to the entire domain $\mathbf{R}$ as follows:

$$\begin{aligned} \Delta u - \lambda u &= \begin{cases} f, & \mathbf{x} \in \Omega, \\ 0, & \mathbf{x} \in \mathbf{R} - \Omega, \end{cases} \\ q(u, u_n) &= 0, \quad \mathbf{x} \in \partial\Omega. \end{aligned}$$

$$\begin{cases} [u] &= g, \\ [u_n] &= 0, \end{cases} \quad \text{or} \quad \begin{cases} [u] &= 0, \\ [u_n] &= g, \end{cases} \quad \mathbf{x} \in \partial\Omega.$$

If g is known, then we can find the solution u with a fast Poisson solver. In the discrete sense, this can be represented by a matrix-vector equation

$$A\mathbf{u} = \mathbf{f} - B\mathbf{g}. \tag{2}$$

To solve the original problem, the augmented variable g is determined so that $q(u(g), u_n(g)) = 0$ for $u(g)$ along the boundary/interface $\partial\Omega$. This is the second discretization in AIIM. One strategy of the discretization is to apply least squares interpolations [19, 21] in terms of $\mathbf{u}$ and $\mathbf{g}$ at a set of discrete points along $\partial\Omega$, which leads to a matrix-vector equation

$$C\mathbf{u} + D\mathbf{g} - \mathbf{q} = 0. \tag{3}$$

The residue vector is

$$\mathcal{R}(\mathbf{g}) = C\mathbf{u}(\mathbf{g}) + D\mathbf{g} - \mathbf{q}.$$

$\mathcal{R}(\mathbf{g})$ here has dual meanings. It is not only the regular residual of the linear system (6) below, but is also the measurement of how the boundary condition is satisfied. $\mathbf{u}$ is the solution when $\mathcal{R}(\mathbf{g}) = \mathbf{0}$.

Combine (2) and (3) to get

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{q} \end{pmatrix}.$$

Compute a block LU factorization

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} A & \\ C & I \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ & \mathbf{A} \end{pmatrix}, \tag{4}$$

where $\mathbf{A}$ is the Schur complement

$$\mathbf{A} = D - CA^{-1}B. \tag{5}$$

We can then solve a much smaller system for $\mathbf{g}$:

$$\mathbf{A}\mathbf{g} = \mathbf{b}, \quad \text{with} \quad \mathbf{b} = \mathbf{q} - CA^{-1}\mathbf{f}. \tag{6}$$

Once we find $\mathbf{g}$, then we solve (2) for the solution $\mathbf{u}$. The right-hand side vector $\mathbf{b}$ in (6) can be found with the evaluation of $-\mathcal{R}(\mathbf{g})$ at $\mathbf{g} = \mathbf{0}$, since

$$-\mathcal{R}(\mathbf{0}) = -(C\mathbf{u}(\mathbf{0}) + D\mathbf{0} - \mathbf{q}) = -(CA^{-1}\mathbf{f} - \mathbf{q}) = \mathbf{b}.$$

In iterative solutions of (6), we need to evaluate the products of $\mathbf{A}$ with vectors $\tilde{\mathbf{g}}$. For this purpose, we first solve (2) with $\mathbf{g}$ replaced by $\tilde{\mathbf{g}}$:

$$A\tilde{\mathbf{u}}(\tilde{\mathbf{g}}) = \mathbf{f} - B\tilde{\mathbf{g}}.$$

Then

$$\begin{aligned}\mathbf{A}\tilde{\mathbf{g}} &= D\tilde{\mathbf{g}} - CA^{-1}B\tilde{\mathbf{g}} = D\tilde{\mathbf{g}} - CA^{-1}(\mathbf{f} - A\tilde{\mathbf{u}}(\tilde{\mathbf{g}})) \\ &= D\tilde{\mathbf{g}} - C\mathbf{u}(\mathbf{0}) + C\tilde{\mathbf{u}}(\tilde{\mathbf{g}}) = (C\tilde{\mathbf{u}}(\tilde{\mathbf{g}}) + D\tilde{\mathbf{g}}) - C\mathbf{u}(\mathbf{0}) \\ &= (C\tilde{\mathbf{u}}(\tilde{\mathbf{g}}) + D\tilde{\mathbf{g}} - \mathbf{q}) - (D\mathbf{0} + C\mathbf{u}(\mathbf{0}) - \mathbf{q}) \\ &= \mathcal{R}(\tilde{\mathbf{g}}) - \mathcal{R}(\mathbf{0}).\end{aligned} \tag{7}$$

That is, to compute $\mathbf{A}\tilde{\mathbf{g}}$, we can use an interpolation to get the residual for the boundary condition.

In general, the Schur complement matrix $\mathbf{A}$ is nonsymmetric. Even A may be nonsymmetric, such as in the other applications in this paper. $\mathbf{A}$ is generally a dense matrix since A^{-1} is. In the boundary integral method, $\mathbf{A}$ is close to the discretization of the second kind integral equation. In the examples presented in this paper, the matrices A, B, C, and D are not explicitly formed. The matrices C and D are sparse and rely on the interpolation scheme used to approximate the boundary condition. There are about $3 \sim 16$ entries each row, depending on the geometry of the boundary and its neighboring grid points for different applications. C and D are determined from the interpolation scheme to approximate the boundary condition. Hence, we generally do not have C^T and D^T available if the matrices are not formed. Thus in practice, $\mathbf{A}$ is not explicitly available, and the multiplication of $\mathbf{A}^T$ by vectors is not convenient.

2.3. *Features of the Schur complement systems and challenges in GMRES solutions*

As mentioned above, the Schur complement matrices $\mathbf{A}$ in AIIM such as (5) are usually dense and not explicitly formed. On the other hand, $\mathbf{A}$ can be multiplied by vectors quickly with the aid of fast solvers (e.g., Poisson solvers). Thus, iterative methods such as GMRES are usually used to solve the Schur complement system

$$\mathbf{A}\mathbf{g} = \mathbf{b}. \tag{8}$$

For the problems we consider, the following features or challenges are often observed.

- For many Schur complement systems resulting from AIIM, GMRES without preconditioning has difficulty in converging. By saying this, we mean that the restarted GMRES method stalls or the non-restarted GMRES method converges only when the number of iterations reaches nearly the size N of $\mathbf{A}$. Sometimes, this is due to the ill conditioning of $\mathbf{A}$. However, for various cases here, this happens even if $\mathbf{A}$ is well conditioned. Often, the eigenvalues of $\mathbf{A}$ are scattered around the origin, which is empirically observed to affect the convergence of GMRES. The physical background for the slow convergence is as follows. AIIM can be considered as a generalized boundary integral method without explicitly using Green functions. The augmented variable can be considered as a source term. Thus, if the discrete system corresponds to an integral equation of the second kind, then GMRES can converge quickly. (We refer the readers to [49] for the relation between an augmented approach and the boundary integral method.) One such an example is to solve a Poisson equation on an irregular domain with different boundary conditions. If the system behaves like an integral equation of the first kind, then GMRES converges slowly in general. One such example is to solve a Poisson equation on an irregular domain with both Dirichlet and Neumann boundary conditions defined along part of the boundary. Thus, an effective preconditioner is crucial for the convergence of GMRES.

- $\mathbf{A}$ is usually dense and it is costly to form it. (For example, A^{-1} is involved in (5) for a large sparse matrix A.) In fact, it is not convenient to even extract its diagonal. Even if we find the diagonal, it may have zero entries and cannot be used as a preconditioner in a straightforward way.
- $\mathbf{A}\tilde{\mathbf{g}}$ can be conveniently computed for a vector $\tilde{\mathbf{g}}$. Thus, we may extract few columns of $\mathbf{A}$ or multiply $\mathbf{A}$ by certain special vectors (e.g., vector of ones), so as to construct diagonal or block diagonal preconditioners. However, the preconditioners may be close to singular or may perform poorly.
- $\mathbf{A}^T\tilde{\mathbf{g}}$ cannot be conveniently computed for a vector $\tilde{\mathbf{g}}$. (See the end of the previous subsection.) Thus, even the recent matrix-free direct solution techniques in [28, 42] cannot be used to get a preconditioner, since they require the multiplication of both $\mathbf{A}$ and $\mathbf{A}^T$ by vectors to get an approximation to $\mathbf{A}$.
- In some cases, $\mathbf{A}$ is close to be normal or even symmetric.
- The singular values of the off-diagonal blocks of $\mathbf{A}$ have reasonable decay. In some cases, the decay is very fast so that such blocks have small numerical ranks.

As an example, we consider a 680×680 Schur complement $\mathbf{A}$ arising from AIIM for solving the Navier-Stokes equation with a traction boundary condition in Section 4.1.1. The mesh size is 240×240. Note that the 2-norm condition number of the matrix is $\kappa_2(\mathbf{A}) = 10.09$, and $\|\mathbf{A} - \mathbf{A}^T\|_2 = 0.02$, $\|\mathbf{A}^T\mathbf{A} - \mathbf{A}\mathbf{A}^T\|_2 = 1.98 \times 10^{-5}$. However, due to the traction boundary condition, restarted GMRES (with 50 inner iterations) applied to (8) fails to converge, as shown in Figure 1.

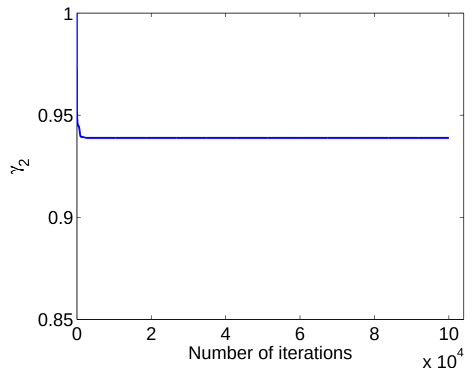


Figure 1: Convergence of restarted GMRES without preconditioning for (8) from AIIM for solving the Navier-Stokes equations with a traction boundary condition in Section 4.1.1, where $N = 680$, the mesh size is 240×240, and $\gamma_2 = \frac{\|\mathbf{Ag}-\mathbf{b}\|_2}{\|\mathbf{b}\|_2}$ is the relative residual.

Thus, an effective preconditioner is needed. The diagonal of $\mathbf{A}$ contains zero entries and cannot be conveniently used as a preconditioner, even if we could extract it. We will seek to find a structured preconditioner. In fact, the off-diagonal blocks of $\mathbf{A}$ have relatively small numerical ranks. In Figure 2, we show the singular values of an $\frac{N}{2} \times \frac{N}{2}$ off-diagonal block of $\mathbf{A}$. Clearly, the block only has few large singular values. Thus, we can use a low-rank form to approximate this block to a reasonable accuracy. Overall, we can approximate $\mathbf{A}$ by a rank structured (e.g., HSS) matrix that can serve as an effective preconditioner. However, this would require either $\mathbf{A}$ explicitly or the multiplication of both $\mathbf{A}$ and $\mathbf{A}^T$ by vectors. In the next section, we address these issues.

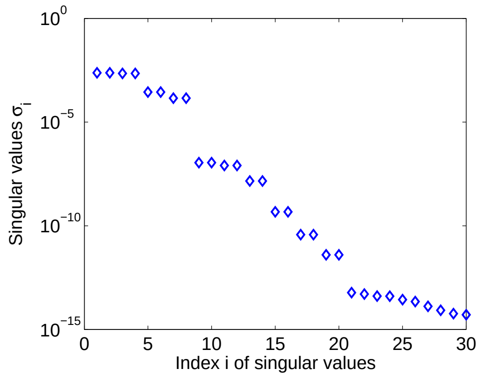


Figure 2: The first 30 singular values of $\mathbf{A}(1 : \frac{N}{2}, \frac{N}{2} + 1 : N)$ for the Schur complement $\mathbf{A}$ from AIIM for solving the Navier-Stokes equations with a traction boundary condition in Section 4.1.1, where $N = 680$ and the mesh size is 240×240.

3. Matrix-free preconditioning techniques for AIIM

The challenges and the tests in the previous section provide a motivation for this work. We show a matrix-free scheme that improves those in [28, 48] and produces a nearly symmetric approximation H to $\mathbf{A}$ in a structured form. H will be quickly factorized and used as a structured preconditioner.

3.1. Rank structures

Our preconditioning techniques employ rank structures, or more specifically, HSS representations [6, 46]. An HSS representation provides a convenient format to organize the off-diagonal blocks of a matrix H by low-rank forms. H is partitioned hierarchically, so that the off-diagonal blocks at all the hierarchical levels have low-rank forms and also share certain common bases. A comprehensive summary of HSS structures and algorithms can be found in [43]. Here, we briefly review its definition, with the aid of the following notation:

- $H|_{s\times t}$ is a submatrix of H formed by its entries corresponding to the row index set s and column index set t;
- $H|_s$ is a submatrix of H formed by its rows corresponding to the index set s;
- $H|_{:\times t}$ is a submatrix of H formed by its columns corresponding to the index set t.

A general HSS form looks like $D_k \equiv H|_{\mathcal{I}\times\mathcal{I}}$, where $\mathcal{I}$ is the index set $\{1, 2, \ldots, N\}$, and D_k is defined recursively following a full binary tree T with k nodes labeled as $i = 1, 2, \ldots, k$ in a postorder. Each node i of T is associated with an index set t_i such that

$$t_k = \mathcal{I}, \quad t_i = t_{c_1} \cup t_{c_2}, \quad t_{c_1} \cap t_{c_2} = \emptyset,$$

where c_1 and c_2 are the children of a non-leaf node i and $c_1 < c_2$ (c_1 ordered before c_2). Then for each such i, recursively define

$$D_i = \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1} V_{c_2}^T \\ U_{c_2} B_{c_2} V_{c_1}^T & D_{c_2} \end{pmatrix}, \quad U_i = \begin{pmatrix} U_{c_1} R_{c_1} \\ U_{c_2} R_{c_2} \end{pmatrix}, \quad V_i = \begin{pmatrix} V_{c_1} W_{c_1} \\ V_{c_2} W_{c_2} \end{pmatrix},$$

where the matrices D_i, U_i, etc. are called the *generators* that define the HSS form of H. Also, if we associate each node i of T with the corresponding generators, T is called an *HSS tree*. T can be used to conveniently organize the storage of H and perform HSS operations. The off-diagonal blocks we are interested in are $H|_{t_i \times (\mathcal{I} \backslash t_i)}$ and $H|_{(\mathcal{I} \backslash t_i) \times t_i}$, called *HSS block rows and columns*, respectively. U_i gives a column basis for $H|_{t_i \times (\mathcal{I} \backslash t_i)}$, and V_i gives a column basis for $(H|_{(\mathcal{I} \backslash t_i) \times t_i})^T$. For convenience, assume U_i and V_i have orthonormal columns. In standard HSS computations, the generators are usually dense. However, particular HSS construction algorithms may yield generators that have additional internal structures (see, e.g., [44, 48]).

Later, we use par(i) and sib(i) to denote the parent and the sibling of i, respectively. Also we say c_1 is a left node and c_2 is a right one.

3.2. Matrix-free structured preconditioning via randomized sampling

HSS preconditioning for symmetric positive definite problems have been discussed in [47]. Here, our matrices are generally nonsymmetric and indefinite. The primary idea of our preconditioning techniques is to construct a nearly-symmetric HSS approximation H to $\mathbf{A}$ with an improved matrix-free HSS construction scheme via only the products of $\mathbf{A}$ and vectors. The major components are as follows.

1. For an off-diagonal block of $\mathbf{A}$, use an adaptive randomized sampling method [15, 42] to find an approximate column basis, and a deterministic method to find an approximate row basis.
2. In a top-down traversal of the HSS tree, use the products of $\mathbf{A}$ and vectors to obtain hierarchically low-rank approximations to certain off-diagonal blocks. Treat $\mathbf{A}$ as a symmetric matrix to approximate the other off-diagonal blocks. We avoid using pseudoinverses and some matrix multiplications in [28, 42].
3. This further enables us to approximate the diagonal blocks, and we then obtain a nearly symmetric HSS approximation H to $\mathbf{A}$.
4. Compute a structured ULV-type factorization [6, 46] of H. Use the factors as a preconditioner in a ULV solution scheme.

The detailed improvements over existing similar schemes are elaborated in the following subsections.

3.2.1. Improved randomized compression

Firstly, we explain briefly the adaptive randomized sampling ideas, and show our improvement to the randomized compression. For an $M \times N$ matrix Φ of rank or numerical rank r, we seek to find a low-rank approximation of the form

$$\Phi \approx UBV^T.$$

We multiply Φ and random vectors and use adaptive randomized sampling to find U as in [15, 40]. Then unlike the methods in [15, 28, 40, 42], we do not multiply Φ^T with random vectors when finding V. Furthermore, we do not use pseudoinverses to find B. These are detailed as follows.

Initially, let X be an $M \times \tilde{r}$ Gaussian random matrix, where $\tilde{r}$ is a conservative estimate of r. Compute the product

$$Y = \Phi X,$$

and a QR factorization

$$Y = U\tilde{Y}. \tag{9}$$

U is expected to provide a column basis matrix for Φ when $\tilde{r}$ is close to r. To quickly estimate how well $U(U^T\Phi)$ approximates Φ, the following bound can be used with high probability [15, 40]:

$$\|\Phi - U(U^T\Phi)\|_2 \le \eta \sqrt{\frac{2}{\pi}} \max_{j=\tilde{r}-d+1,\ldots,\tilde{r}} \left\|(I - UU^T)Y|_{:\times j}\right\|_2, \tag{10}$$

where d is a small integer and η is a real number, and d and η determine the probability. If the desired accuracy is not reached, more random vectors are used. When this stops, we obtain the approximate basis matrix U and the rank estimate.

In existing randomized sampling methods, it then usually multiplies Φ^T with random vectors to find V in a similar way. That is, randomized sampling are used twice to extract both the row and the column basis. Here, instead, we compute

$$\tilde{\Phi} = U^T\Phi, \tag{11}$$

and then compute an RQ factorization

$$\tilde{\Phi} = BV^T. \tag{12}$$

That is, we use randomized sampling only once, but use the deterministic orthogonal way (11)–(12) to find V. This is potentially beneficial for the approximation. Furthermore, (12) provides both B and V without the need of pseudoinverses in [28, 42]. The RQ factorization is generally much faster and much more stable. This idea is similar to a deterministic compression strategy for HSS construction in [46], and a parallel implementation is later discussed in [30].

3.2.2. Nearly-symmetric matrix-free HSS construction with improvements

We then apply the improved randomized compression to the HSS blocks of $\mathbf{A}$:

$$\Phi = \mathbf{A}|_{t_i \times (\mathcal{I}\backslash t_i)} \quad \text{or} \quad \mathbf{A}|_{(\mathcal{I}\backslash t_i)\times t_i}, \tag{13}$$

The compression is done hierarchically for all the HSS blocks. Previous attempts to find accurate HSS approximations to problems with low-rank off-diagonal blocks are made in [28, 32, 42, 48], where the products of both $\mathbf{A}$ and $\mathbf{A}^T$ with random vectors are needed. As compared with the original matrix-free HSS constructions in [28, 42], the following improvements and modifications are made:

- A nearly-symmetric HSS approximation is constructed. That is, for a node i of the HSS tree and $j = \text{sib}(i)$, $V_i \equiv U_i$, $W_i \equiv R_i$, $B_j = B_i^T$, but the D generators are nonsymmetric.

- Randomized sampling is only used to find U_i for the left nodes i. To find U_i for the right nodes i, the deterministic way (11)–(12) is used instead. This reduces the number of matrix multiplications, and avoids expensive and potentially unstable pseudoinverses.
- Specifically for the purpose of preconditioning, a low approximation accuracy and small $\tilde{r}$ and d are used in (10).

The details are as follows. Assume that T is the desired HSS tree. As in [28, 42], the nodes i of T are visited in a top-down way, so as to construct the column bases hierarchically for the HSS blocks $\mathbf{A}|_{t_i\times(\mathcal{I}\backslash t_i)}$. For convenience, we use $\tilde{T}_l$ and $\hat{T}_l$ to denote the sets of left and right nodes at a level l of T, respectively. Also, let the root of T be at level 0 and the leaves be at the largest level $l_{\max}$. For $1 \le l \le l_{\max}$, define

$$\tilde{\mathbf{t}}_l = \bigcup_{i\in\tilde{T}_l} t_i, \quad \hat{\mathbf{t}}_l = \bigcup_{j\in\hat{T}_l} t_j.$$

Let X be a Gaussian random matrix whose column size is decided via adaptive randomized sampling applied to $\mathbf{A}|_{t_i\times t_j}$ for each $i \in \tilde{T}_l$. Define $\tilde{X}$ so that

$$\tilde{X}|_{\tilde{\mathbf{t}}_l} = X|_{\tilde{\mathbf{t}}_l}, \quad \tilde{X}|_{\hat{\mathbf{t}}_l} = 0. \tag{14}$$

Compute

$$\tilde{Y} = \mathbf{A}\tilde{X} - \tilde{Z},$$

where $\tilde{Z}$ is the product of the partially computed HSS form (due to recursion at upper levels) and $\tilde{X}$, denoted

$$\tilde{Z} = \mathsf{hssmv}(\mathbf{A}, \tilde{X}, l-1).$$

($\tilde{Z}$ is 0 if $l = 1$.) Then compute a QR factorization

$$\tilde{Y}|_{t_i} = \bar{U}_i \bar{S}_i,$$

where $\bar{U}_i$ is a column basis matrix for $\mathbf{A}|_{t_i\times t_j}$ with $j = \mathrm{sib}(i)$.

To find a row basis matrix for $\mathbf{A}|_{t_i\times t_j}$, unlike the methods in [28, 42] that still uses randomized sampling, we use a deterministic way. Define a matrix $\hat{X}$, which is a zero matrix except for each $i \in \tilde{T}_l$,

$$\hat{X}|_{t_i} = \bar{U}_i.$$

Compute

$$\hat{Y} = \mathbf{A}\hat{X} - \mathsf{hssmv}(\mathbf{A}, \hat{X}, l-1).$$

Then for each right node j at level l, compute a QR factorization

$$\hat{Y}|_{t_j} = \bar{U}_j \bar{B}_j,$$

where $\bar{U}_j$ is a column basis matrix for $(\mathbf{A}|_{t_i\times t_j})^T$.

We are then ready to find the generators. If $l = 1$, simply set $U_i = \bar{U}_i$, $B_i = \bar{B}_j^T$. Otherwise, after $j = \mathrm{sib}(i)$ is also visited, let $p = \mathrm{par}(i)$ and partition U_p as $\begin{pmatrix} U_{p;1} \\ U_{p;2} \end{pmatrix}$ so that $U_{p;1}$ has the same row size as $\bar{U}_i$ (assuming i is a left node). Then compute QR factorizations

$$\begin{pmatrix} \bar{U}_i & U_{p;1} \end{pmatrix} = U_i \begin{pmatrix} S_i & R_i \end{pmatrix}, \quad \begin{pmatrix} \bar{U}_i & U_{p;2} \end{pmatrix} = U_j \begin{pmatrix} S_j & R_j \end{pmatrix}.$$

Also, set

$$B_i = S_j \bar{B}_j S_i^T.$$

In comparison, multiple pseudoinverses and a lot more matrix multiplications are needed in [28]. An improved version is given in [42], but still needs two more matrix multiplications and a pseudoinverse on top of two randomized sampling stages. Another strategy to avoid the pseudoinverse is later given in [30], and can potentially better reveal the hierarchical structures. However, it needs additional randomized sampling and matrix-vector multiplications. Here, since our purpose is preconditioning, the strategy above is sufficient and preferable.

This process is repeated for the nodes of T in a top-down traversal. After the leaf level is traversed, approximations to all the HSS blocks are obtained. We can then approximate all the diagonal blocks $\mathbf{A}|_{t_i\times t_i}$ for the leaves i by subtracting the products of appropriate off-diagonal blocks of $\mathbf{A}$ and $\mathbf{I} = (I, I, \ldots, I)^T$ from $\mathbf{AI}$ [28]. (Some additional zero columns may be needed for certain block rows of $\mathbf{I}$.) More specifically,

$$\mathbf{A}|_{t_i\times t_i} \approx H|_{t_i\times t_i} \equiv \mathbf{AI} - \mathsf{hssmv}(\mathbf{A}, \mathbf{I}, l_{\max}).$$

By now, we have obtained an HSS approximation H to $\mathbf{A}$, where the off-diagonal blocks have a symmetric pattern, or

$$H|_{t_i\times t_j} = (H|_{t_j\times t_i})^T.$$

The diagonal blocks $D_i \equiv H|_{t_i\times t_i}$ are nonsymmetric. The accuracy of this HSS approximation may be low, but it suffices for our preconditioning purpose.

3.2.3. Nearly-symmetric HSS factorization

At this point, we can quickly factorize the HSS matrix H, and the factors are used for preconditioning. Since H is nearly symmetric, we use the fast ULV factorization in [46], with some simplifications to take full advantage of the off-diagonal symmetric pattern and with some modifications to accommodate the nonsymmetric diagonal blocks. The basic idea includes the following steps.

- Introduce zeros into $H|_{t_i\times(\mathcal{I}\backslash t_i)}$ by expanding U_i into an orthogonal matrix $\tilde{U}_i$ and multiplying $\tilde{U}_i$ to $H|_{t_i}$. This just needs to modify D_i as

$$D_i \leftarrow \tilde{U}_i D_i \tilde{U}_i^T.$$

- Partition D_i into a block 2×2 form $\begin{pmatrix} D_{i;1,1} & D_{i;1,2} \\ D_{i;2,1} & D_{i;2,2} \end{pmatrix}$, so that $D_{i;1,1}$ a square matrix with size equal to the column size of U_i. Partially LU factorize D_i:

$$D_i \equiv \begin{pmatrix} D_{i;1,1} & D_{i;1,2} \\ D_{i;2,1} & D_{i;2,2} \end{pmatrix} = \begin{pmatrix} L_{i;1,1} & \\ L_{i.2,1} & I \end{pmatrix} \begin{pmatrix} G_{i;1,1} & G_{i;1,2} \\ & G_{i;2,2} \end{pmatrix},$$

  where $D_{i;1,1} = L_{i;1,1}G_{i;1,1}$ is the LU factorization of $D_{i;1,1}$, and $G_{i;2,2} = D_{i;2,2} - L_{i.2,1}G_{i;1,2}$.

- After these steps for two sibling nodes i and j of T are finished, merge the remaining blocks. Here, this is to simply set

$$D_p \leftarrow \begin{pmatrix} G_{i;2,2} & B_i \\ B_i^T & G_{j;2,2} \end{pmatrix}.$$

Note that no actual operations are performed. Then we remove i and j from T to obtain a smaller HSS form, called a reduced HSS matrix [45].

- Repeat the above steps on the reduced HSS matrix.

After the factorization, the ULV factors are a sequence of orthogonal and lower and upper triangular matrices and are used as a structured preconditioner. The ULV solution procedure for the preconditioning is similar to that in [46] and is skipped.

3.2.4. Algorithm and variation

The details are shown in Algorithm 1, which includes the HSS approximation and the factorization for constructing the preconditioner.

To make the preconditioner easier to use, we may also use a simplified variation by forming a block diagonal matrix $\mathbf{D}$ from the D_i generators:

$$\mathbf{D} = \operatorname{diag}(D_i | i\text{: leaves of } T).$$

Then factorize $\mathbf{D}$ and use the triangular factors as the preconditioner.

To summarize, we have two types of preconditioners:

- Preconditioner I: Structured preconditioner given by the ULV factors of the HSS approximation H to $\mathbf{A}$;
- Preconditioner II: Block-diagonal preconditioner given by the LU factors of the block diagonal matrix $\mathbf{D}$ formed by the D_i generators of H. This preconditioner is easier to apply, although it may lead to slightly slower convergence.

For convenience, we may simply say that H or $\mathbf{D}$ is the preconditioner.

3.3. Efficiency and effectiveness

The HSS construction costs $O(r^2N)$ flops plus the cost to multiply $\mathbf{A}$ with $O(r\log N)$ vectors, where r is the maximum numerical rank of the HSS blocks. The factorization and the preconditioning costs are $O(r^2N)$ and $O(rN)$ flops, respectively. We may also treat the diagonal blocks as symmetric ones so as to further save the costs. The storage for the preconditioner is $O(rN)$.

In the preconditioning, we use a low accuracy so that r is a very small integer. The preconditioning cost at each GMRES iteration is then $O(N)$. This cost is negligible as compared with the cost of multiplying $\mathbf{A}$ and a vector. The reason is that $\mathbf{A}$ corresponds to the interface and has a much smaller size than the entire problem. See, e.g., (4)–(5), where the multiplication of $\mathbf{A}$ and a vector needs to solve a Poisson problem represented by A. The matrix A has a size much larger than N.

Just like many other preconditioning techniques, a full analytical justification of the effectiveness of the preconditioner is not yet available. Especially, the complex nature of the rank structures makes the analysis a nontrivial issue. However, several aspects of rank structured approximation and randomized preconditioning are closely related and give useful heuristical explanations.

1. HSS representations recursively capture the algebraic structure of the discretization with the off-diagonal blocks corresponding to the interactions of subdomains. A low-accuracy off-diagonal approximation represents essential basic information within the subdomain interaction.

Algorithm 1 Constructing the matrix-free preconditioner

```
1: procedure mfprec
       Input: HSS partition, HSS tree $T$, compression tolerance (and/or rank bound $r$), and
              mat-vec (Schur complement matrix-vector multiplication routine as in (7))
       Output: HSS factors as a preconditioner
2:     $\tilde{X} \leftarrow 0, \quad \hat{X} \leftarrow 0$
3:     for level $l = 1, 2, \ldots, l_{\max}$ do    ▷ Improved nearly-symmetric matrix-free HSS construction
4:         Construct $\tilde{X}$ as in (14) via adaptive randomized sampling
5:         $\tilde{Y} \leftarrow$ mat-vec$(\mathbf{A}, \tilde{X})$ − hssmv$(\mathbf{A}, \tilde{X}, l-1)$    ▷ hssmv returns $0$ if $l = 1$
6:         for each left node $i$ at level $l$ do
7:             $\tilde{Y}|_{t_i} = \bar{U}_i S_i$    ▷ QR factorization
8:             $\hat{X}|_{t_i} \leftarrow \bar{U}_i$
9:         end for
10:        $\hat{Y} \leftarrow$ mat-vec$(\mathbf{A}, \hat{X})$ − hssmv$(\mathbf{A}, \hat{X}, l-1)$
11:        for each right node $j$ at level $l$ do
12:            $\hat{Y}|_{t_j} = \bar{U}_j \bar{B}_j$    ▷ QR factorization
13:            $i \leftarrow \mathrm{sib}(j)$
14:            if $l = 1$ then    ▷ i is a child of the root
15:                $U_i \leftarrow \bar{U}_i, \quad U_j \leftarrow \bar{U}_j, \quad B_i \leftarrow \bar{B}_j^T$    ▷ $U, B$ generators
16:            else
17:                $U_{\mathrm{par}(i)} = \begin{pmatrix} U_{\mathrm{par}(i);1} \\ U_{\mathrm{par}(i);2} \end{pmatrix}$    ▷ Partition so that $U_{\mathrm{par}(i);1}$ and $\bar{U}_i$ have the same row size
18:                $\begin{pmatrix} \bar{U}_i & U_{\mathrm{par}(i);1} \end{pmatrix} = U_i \begin{pmatrix} S_i & R_i \end{pmatrix}, \quad \begin{pmatrix} \bar{U}_j & U_{\mathrm{par}(i);2} \end{pmatrix} = U_j \begin{pmatrix} S_j & R_j \end{pmatrix}$
                                                                   ▷ QR factorizations for $U, R$ generators
19:                $B_i \leftarrow S_j \bar{B}_j S_i^T$    ▷ $B$ generator
20:            end if
21:        end for
22:    end for
23:    $\tilde{Y} \leftarrow \mathbf{AI}$ − hssmv$(\mathbf{A}, \mathbf{I}, l_{\max})$
24:    for each leaf $i$ do
25:        $D_i \leftarrow \tilde{Y}|_{t_i}$    ▷ $D$ generator
26:    end for
27:    for node $i = 1, 2, \ldots, k$ do    ▷ ULV factorization to generate the preconditioner
28:        $\tilde{U}_i \leftarrow U_i$    ▷ Extension of $U_i$ to orthogonal $\tilde{U}_i$; implicit zero introduction into $A|_{t_i \times (\mathcal{I} \backslash t_i)}$
29:        $D_i \leftarrow \tilde{U}_i D_i \tilde{U}_i^T$    ▷ Diagonal block update
30:        $D_i = \begin{pmatrix} L_{i;1,1} & \\ L_{i.2,1} & I \end{pmatrix} \begin{pmatrix} G_{i;1,1} & G_{i;1,2} \\ & G_{i;2,2} \end{pmatrix}$    ▷ Partial LU factorization
31:        if $i$ is a nonleaf node then
32:            $D_i \leftarrow \begin{pmatrix} G_{c_1;2,2} & B_{c_1} \\ B_{c_1}^T & G_{c_2;2,2} \end{pmatrix}$    ▷ $c_1, c_2$: children of $i$
33:        end if
34:    end for
35: end procedure
```

2. Keeping few largest off-diagonal singular values in rank structured approximation tends to have an effect of roughly preserving certain eigenvalues of the original matrix [39, 42]. In particular, if some eigenvalues are well separated from the others, then a low-accuracy HSS approximation can give a much more accurate approximation of these eigenvalues. This structured perturbation analysis is shown in [39]. Therefore, when the HSS approximation is used as a preconditioner, it can potentially help to bring the eigenvalues closer. Although this does not necessarily guarantee the fast convergence of GMRES [12, 38], it gives an empirical way to look into the behavior of the preconditioner.
3. Even if the problem may not have the low-rank property or the off-diagonal singular values only slowly decay, a structured approximation as a preconditioner can significantly accelerate the decay of the condition number κ [47], where the preconditioner is obtained with different numbers of off-diagonal singular values kept in the approximation.
4. In our preconditioner, multiplications of $\mathbf{A}$ by random matrices are used to approximate the off-diagonal blocks by low-rank forms, which are further used to approximate the diagonal blocks via matrix addition/subtraction. This is then similar to the idea of additive preconditioning [33, 34], where random low-rank matrices are added to the original matrix, which is shown to yield a well-conditioned preconditioner, and also improves the condition number of the original matrix.

4. Preconditioning AIIM for different applications and generalizations

In this section, we show several important applications where our preconditioning techniques for AIIM can be applied, particularly for flow problems. Numerical results are given to demonstrate the effectiveness and efficiency. We focus on the structured Preconditioner I given at the end of Section 3.2, and also briefly show the performance of Preconditioner II.

4.1. Preconditioning individual Schur complements in various applications

In this subsection, we show the preconditioning of some individual Schur complements in several applications. In our tests, different types of right-hand sides $\mathbf{b}$ are tried, such as random vectors, products of $\mathbf{A}$ and given vectors, and those actually arising in the applications. Similar performance is observed. Thus for convenience in comparing different methods, we report the results with $\mathbf{b}$ to be the products of $\mathbf{A}$ and vectors of all ones. We point out that our preconditioners are not limited to any specific type of right-hand sides. For convenience, the following notation is used:

- $\kappa_2(\mathbf{A})$: 2-norm condition number of $\mathbf{A}$;
- $n_{\mathrm{mv}}^{\mathrm{pre}}$: number of matrix-vector multiplications for computing the preconditioner;
- n_{it}: number of GMRES iterations;
- $\gamma_2 = \frac{\|\mathbf{Ag}-\mathbf{b}\|_2}{\|\mathbf{b}\|_2}$: relative residual.

Before presenting the details of the individual problems, we give the basic properties of the matrices in Table 1, and summarize the convergence results in Table 2. The matrices include both well-conditioned and ill-conditioned ones. We can observe that, GMRES has difficulty to converge for all the cases. In fact, restarted GMRES fails to converge for all the cases except one (where it takes 8794 steps to converge for $N = 194$). Even non-restarted GMRES needs large numbers of iterations to converge, which are often close to N. On the other hand, preconditioned GMRES with our structured preconditioner gives quick convergence for all the cases. Both $n_{\mathrm{mv}}^{\mathrm{pre}}$ and n_{it} are relatively insensitive to the increase of N.

Table 1: Properties of **A**, including the corresponding mesh size for the entire domain.

Problem	Mesh	N	$\kappa_2(\mathbf{A})$	$\|\|\mathbf{A}-\mathbf{A}^T\|\|_2$	$\frac{\|\|\mathbf{A}-\mathbf{A}^T\|\|_2}{\|\|\mathbf{A}\|\|_2}$	$\|\|\mathbf{A}^T\mathbf{A}-\mathbf{A}\mathbf{A}^T\|\|_2$	$\frac{\|\|\mathbf{A}^T\mathbf{A}-\mathbf{A}\mathbf{A}^T\|\|_2}{\|\|\mathbf{A}^T\mathbf{A}\|\|_2}$
`Traction`	60×60	176	9.62	0.02	1.43	2.36×10^{-5}	0.10
	120×120	344	9.48	0.02	1.40	2.23×10^{-5}	0.09
	240×240	680	10.09	0.02	1.41	1.98×10^{-5}	0.09
	480×480	1360	13.71	0.02	1.42	1.66×10^{-5}	0.08
`Inextensible`	128×128	128	$1.55e5$	4.08	0.69	9.33	0.27
	256×256	256	$1.34e7$	8.05	0.70	39.05	0.29
	512×512	512	$2.31e6$	18.65	0.51	176.85	0.13
`Contact`	128×128	182	$6.79e2$	1.01	1.02	0.86	0.88
	256×256	362	$9.56e2$	1.02	1.03	0.86	0.87
	512×512	726	$4.76e3$	1.06	0.99	0.89	0.78
`Mix_Irregular`	64×64	194	$3.16e3$	1.03	0.98	0.91	0.82
	128×128	274	$3.46e3$	1.26	0.97	1.52	0.89
	256×256	514	$1.71e4$	2.21	0.99	4.83	0.97
	512×512	834	$2.54e5$	3.74	1.00	13.91	0.99
	1024×1024	1314	$8.59e6$	6.91	1.00	47.58	1.00

Remark 1. Since **A** corresponds to the interface for the augmented variable, the size N of **A** is usually not very large. However, the number of variables in the original problem is much larger. See the mesh size in Table 1. Even if **A** may have a small size, the multiplication of **A** by a vector needs the solution of a much larger equation for A, e.g., with a Poisson solver.

Remark 2. For time dependent problems and multiple right-hand sides, we may use $n_{\mathrm{mv}}^{\mathrm{pre}}$ matrix-vector multiplications in a precomputation to find a preconditioner, and then apply the preconditioner to multiple time steps and right-hand sides. Thus, it sometimes makes sense to allow $n_{\mathrm{mv}}^{\mathrm{pre}}$ to be slightly larger so as to reduce n_{it}. This would be beneficial in reducing the total number of matrix-vector products.

Remark 3. The measurements in Table 1 only give a partial way to look into the symmetry/normality of **A**, and do not necessarily tell the actual symmetry/normality. In practice, we may not know if **A** is actually close to symmetric or not. For some of the examples, the eigenvalues of **A** are scattered around (say, in a disk) and are not close to the real axis. In general, the structure of the Schur complement matrix depends on the application and the representation of the interface. For our test examples, it is hard to tell whether the matrix is eventually close to symmetric or not, but the preconditioner works well.

Remark 4. The reasons why we are not reporting results for restarted GMRES are as follows. First, the numbers of iterations in our tests are usually small. Next, if restart is used, the conver-

Table 2: Summary of the convergence of GMRES without preconditioning and with our structured Preconditioner I, where τ is around $0.1 \sim 0.8$ in constructing the preconditioner.

Problem	GMRES				Preconditioned GMRES (Non-restarted)		
	Restarted		Non-restarted				
	n_{it} (outer; inner)	γ_2	n_{it}	γ_2	$n_{\text{mv}}^{\text{pre}}$	n_{it}	γ_2
`Traction`	91750 (1835; 50)	Fail	175	$1.43e-4$	63	12	$3.73e-7$
	93650 (1873; 50)	Fail	343	$1.34e-5$	88	11	$5.83e-7$
	9500 (190; 50)	Fail	672	$3.11e-6$	85	12	$5.14e-7$
	3300 (66; 50)	Fail	1331	$1.71e-6$	106	13	$9.01e-7$
`Inextensible`	55750 (1115; 50)	Fail	127	$1.63e-4$	73	26	$7.90e-7$
	30200 (604; 50)	Fail	250	$8.91e-7$	92	42	$7.96e-8$
	100000 (2000; 50)	Fail	492	$4.63e-7$	110	68	$2.94e-7$
`Contact`	49250 (985; 50)	Fail	181	$6.98e-6$	71	21	$3.47e-7$
	81500 (1630; 50)	Fail	361	$5.20e-7$	83	30	$5.47e-7$
	3150 (63; 50)	Fail	725	$1.71e-6$	99	48	$8.01e-7$
`Mix_Irregular`	8794 (176; 50)	$1.00e-6$	118	$7.03e-7$	68	22	$7.82e-7$
	14150 (283; 50)	Fail	156	$8.14e-7$	79	39	$2.47e-7$
	95200 (1904; 50)	Fail	328	$9.48e-7$	81	64	$7.61e-7$
	7300 (146; 50)	Fail	633	$9.18e-7$	102	84	$9.60e-7$
	33950 (679; 50)	Fail	1045	$9.36e-7$	105	98	$9.09e-7$

gence is slower than without, depending on the number of inner iterations. However, in AIIM with preconditioning, it seems preferable to store a little more intermediate iterates (as in non-restarted GMRES) so as to accelerate the convergence. This is because the sizes of the original discretized PDEs are much larger than the sizes of the Schur complement matrices $\mathbf{A}$. It is usually cheap to store some small intermediate iterates in GMRES, but is costly to compute matrix-vector products. Therefore, reducing the total number of matrix-vector products is more crucial (than saving a small amount of storage for the GMRES iterates). Of course, for large-scale practical computations, we may choose to balance the convergence and the storage via restart with appropriate numbers of inner iterations.

We then explain the individual applications in Tables 1–2 and show additional specific tests in the following subsections.

4.1.1. Navier-Stokes equations on irregular domains with open and traction boundary conditions

This application is one of our motivations to develop the preconditioner. A detailed description of the problem and its solution with AIIM can be found in [20].

Let $\mathbf{R}$ be a rectangular domain with an inclusion Ω. Consider the Navier-Stokes equation

$$\rho\left(\frac{\partial u}{\partial t} + (u\cdot\nabla)u\right) + \nabla p = \mu\Delta u + g, \quad \nabla\cdot u = 0, \qquad (15)$$
$$\mathbf{x} \in \mathbf{R}\setminus\Omega,$$

where ρ, μ, u, p, and $g(x, y, t)$ are the fluid density, the viscosity, the fluid velocity, the pressure, and an external forcing term, respectively. There is a traction boundary condition [29] along the interior boundary $\partial\Omega$:

$$n^T \cdot \mu(\nabla u + \nabla u^T) \cdot n = p - \hat{p} - \gamma\xi, \quad \eta^T \cdot \mu(\nabla u + \nabla u^T) \cdot n = 0, \quad \mathbf{x} \in \partial\Omega, \tag{16}$$

where ξ is the curvature, γ is the coefficient of the surface tension, η is the unit tangential direction of the interface, and $\hat{p}$ is the pressure of the air.

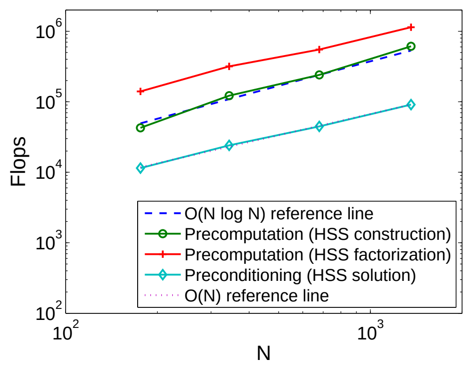
During the numerical solution, the procedure to advance from a time step t^k to the next one t^{k+1} includes two steps [20]. The first is to solve (15) with (16) for the velocity using AIIM. The second is to solve the momentum equation for the pressure p. The augmented variable is $q^{k+1} \equiv \frac{\partial u^{k+1}}{\partial n}$, which is determined so that u^{k+1} satisfies (16) with an approximation to p^{k+1}.

In a specific example, set the boundary $\partial\Omega$ to be a particular curve such as a circle defined by a function $\varphi(x, y)$. Let $\varphi_{i,j}$ be the discrete case of $\varphi(x, y)$. A grid point $\mathbf{x}_{ij}$ is irregular if

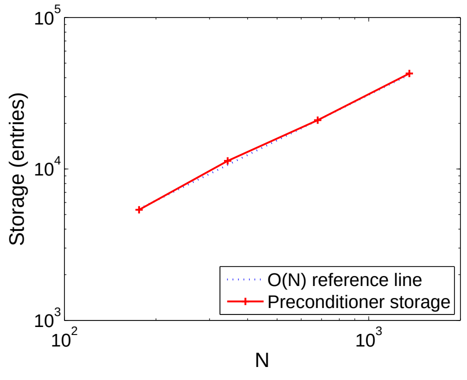
$$\max\left\{\varphi_{i-1,j}, \varphi_{i+1,j}, \varphi_{ij}, \varphi_{i,j-1}, \varphi_{i,j+1}\right\} \cdot \min\left\{\varphi_{i-1,j}, \varphi_{i+1,j}, \varphi_{ij}, \varphi_{i,j-1}, \varphi_{i,j+1}\right\} \le 0.$$

The Schur complement system (8) for the augmented variable is defined at the orthogonal projections of such irregular $\mathbf{x}_{ij}$ on $\partial\Omega$ from the outside. If the time step size $\Delta t = t^{k+1} - t^k$ is fixed, then so is $\mathbf{A}$.

(8) is solved by GMRES with our preconditioning techniques. Preconditioned GMRES quickly converges. See the results in the row of '`Traction`' in Table 2, where we use time $t = 0$ and $\mu = 0.02$. Note that the number of matrix-vector products for constructing the preconditioner and the number of iterations increase very slowly with N. More specifically, Figure 3 shows the costs (excluding the matrix-vector multiplication which is problem dependent) and the storage. For varying N, we use a low accuracy $\tau = 0.4$ and a small off-diagonal rank. Then the construction of the preconditioner (including the HSS construction and the ULV factorization) and the application of the preconditioner (the ULV solution) both cost about $O(N)$ flops and need about $O(N)$ storage. Such costs are negligible as compared with even one matrix-vector product with $\mathbf{A}$, which costs about $O(N^2 \log N)$.



(i) Costs of the algorithms



(ii) Storage for the preconditioner

Figure 3: Costs for precomputing or constructing the preconditioner (including HSS construction and ULV factorization) and applying the preconditioner (HSS solution), and the storage for the preconditioner.

In particular, for the example with $N = 680$ in Figure 1 in Section 2.3, the convergence of preconditioned GMRES is significantly faster than the standard non-restarted GMRES method.

(Note that restarted GMRES simply stalls.) See Figure 4. It is also observed that most of the eigenvalues of the preconditioned matrix cluster around 1.

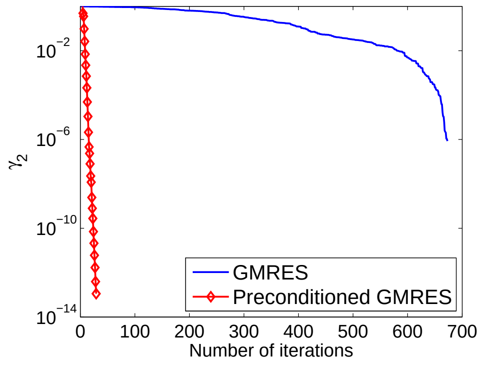


Figure 4: Convergence of non-restarted GMRES without preconditioning and with our structured preconditioning for the 680×680 Schur complement **A** (corresponding to Figure 1) from a 240×240 mesh.

We also test the problem at time $t = 0.5$, and the results are given in Table 3. The convergence is very close to that for $t = 0$ in Table 2.

Table 3: Convergence of preconditioned GMRES for the problem '`Traction`' at time $t = 0.5$.

Mesh	$n_{\text{mv}}^{\text{pre}}$	n_{it}	γ_2
60×60	63	13	$4.29e-7$
120×120	88	12	$6.81e-7$
240×240	85	11	$6.37e-7$
480×480	106	13	$6.53e-7$

Furthermore, for the example with $N = 680$, we also test a wide range of μ values. See Table 4. Clearly, at both $t = 0$ and 0.5, the convergence preconditioned GMRES is similar for different μ. This illustrates the insensitivity of the convergence to different time steps and physical parameters.

Table 4: Convergence of preconditioned GMRES for the problem '`Traction`' at times $t = 0$ and 0.5 with different μ values.

μ	$t = 0$			$t = 0.5$		
	$n_{\text{mv}}^{\text{pre}}$	n_{it}	γ_2	$n_{\text{mv}}^{\text{pre}}$	n_{it}	γ_2
2	90	14	$5.87e-7$	90	14	$5.26e-7$
0.2	88	13	$4.04e-7$	88	14	$6.13e-7$
0.02	85	12	$5.14e-7$	85	11	$6.37e-7$
0.002	85	13	$3.63e-7$	85	12	$6.58e-7$

4.1.2. An extensible interface in an incompressible flow

Then we consider an inverse interface problem, which has a moving interface $\partial\Omega(t)$ in a shear flow within a rectangular domain **R** [24]. The problem is still modeled by the Navier-Stokes equation:

$$\frac{\partial u}{\partial t} + u \cdot \nabla u + \nabla p = \mu \, \Delta u + f(\mathbf{x}, t), \quad \nabla \cdot u = 0, \\ \mathbf{x} \in \mathbf{R},$$

where $f(\mathbf{x}, t)$ is the force term. Let η be the tangential direction of $\partial\Omega(t)$. The force term is

$$f(\mathbf{x}, t) = \int_{\partial\Omega(t)} \frac{\partial}{\partial s} \left(\sigma(s, t) \, \eta(s, t) \right) \delta(\mathbf{x} - \omega(s, t)) \, ds,$$

where $\omega(s, t)$ is a parametric representation of $\partial\Omega(t)$, and $\sigma(s, t)$ is the surface tension. The force term $f(\mathbf{x}, t)$ is also part of the unknown. We need to find $f(\mathbf{x}, t)$ such that the incompressible condition along the tangential direction is also satisfied. $\sigma(s, t)$ is chosen as the augmented variable so that

$$(\nabla_s \cdot u)_{\partial\Omega} = \left. \frac{\partial u}{\partial \eta} \cdot \eta \right|_{\partial\Omega} = 0.$$

That is, both the length of $\partial\Omega(t)$ and the enclosed area remain constant.

For this problem, the number of GMRES iterations is dramatically reduced with our preconditioner. See the results in the '`Inextensible`' row in Table 2, where the tests are done at time $t = 0$ with $\mu = 20$.

4.1.3. A contact problem of drop spreading

AIIM for modeling a moving contact line problem where a liquid drop spreads can be found in [23]. Just like in the previous problems, the fluid domain Ω with the free boundary $\partial\Omega$ is extended into a rectangular domain $\mathbf{R}$. The augmented variable is also the jump of the normal derivative of the velocity along $\partial\Omega$. The performance of GMRES with our preconditioner is given in the '`Contact`' row in Table 2, where the time is $t = 0$.

4.1.4. A boundary value problem with mixed boundary conditions on different parts of the boundary

Another important problem where an effective preconditioner is needed in AIIM is the Poisson equation with different types of boundary conditions on different parts of the boundary. As an example, consider the domain Ω enclosed by a half circle $x^2 + y^2 \le 1$, $x \ge 0$ and the line segment $x = 0$, $-1 \le y \le 1$. Assume we have the following mixed boundary conditions:

$$\begin{aligned} u(x, y) &= g_1, \quad x^2 + y^2 = 1, \quad x \ge 0 \quad \text{(Dirichlet)}, \\ \frac{\partial u}{\partial n} &= g_2, \quad x = 0, \quad -1 \le y \le 1 \quad \text{(Neumann)}. \end{aligned}$$

Ω is extended to the rectangular domain $\mathbf{R} = [-2, \ 2]^2$, and AIIM is applied to the problem as explained in Section 2. The augmented variable is set to be $[\frac{\partial u}{\partial n}]$ along the half circle and $[u]$ along the line segment $x = 0$, $-1 \le y \le 1$. See the line of '`Mix_Irregular`' in Table 2 for the performance of our preconditioner.

In particular, we also try Preconditioner II given at the end of Section 3.2. See Table 5. We observe satisfactory convergence results too, though slightly slower than those in Table 2.

Table 5: Convergence of GMRES with our Preconditioner II for the problem '`Mix_Irregular`' in Table 1.

Mesh	n_{it}	γ_2
64×64	24	$6.59e-7$
128×128	38	$9.12e-7$
256×256	66	$4.44e-7$
512×512	97	$8.34e-7$
1024×1024	111	$7.88e-7$

4.2. Comprehensive simulation in terms of a free boundary problem

We then show a comprehensive test in terms of a free boundary problem [25], including all the stages of an entire simulation.

This problem is to determine the position of the crack of certain minimizers. It is a mixed boundary value problem where we need to find a potential function $u(x,y)$ and a free boundary $\Gamma_1(x,y)$ such that

$$\Delta u = 0 \quad \text{in} \quad \Omega, \tag{17}$$

$$u|_{\Gamma_2} = w, \tag{18}$$

$$\left.\frac{\partial u}{\partial \nu}\right|_{\Gamma_1} = 0, \quad \frac{\pi}{2}c = \left[\left|\nabla u^+\right|^2 - \left|\nabla u^-\right|^2\right]_{\Gamma_1}, \tag{19}$$

where ν is the unit normal of Γ_1 and c is the curvature of Γ_1. The other boundary Γ_2 is fixed, and the Dirichlet boundary condition is defined on Γ_2. Note that the parts of free boundary Γ_1 below and above the x-axis are antisymmetric, and the free boundary has three fixed points: the top ($y > 0$) corner, the bottom ($y > 0$) corner, and the origin ($(0,0)$). Figure 5 gives an illustration. Additional plots of the final shape of the boundary can be found in [25].

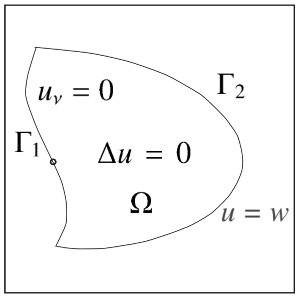


Figure 5: An illustration of the free boundary problem (17)–(19) in [25], where Γ_1 is a free boundary and Γ_2 is fixed.

As usual, we use an iterative scheme to solve this free boundary problem. The free boundary Γ_1 can be written as a perturbation to the line $(\hat{x}_0(t), \hat{y}_0(t)) = (a_1 t, a_2 t)$ connecting the origin and the top corner, where (a_1, a_2) is the direction of the line. Denote the free boundary above the x-axis as $(\hat{x}(t), \hat{y}(t)) = (t, g(t))$ with $g(0) = 0$, $g(d) = 0$, where d is the number such that $(\hat{x}_0(d), \hat{y}_0(d))$ is the fixed top corner.

From the analysis in [25], $g(t)$ is the solution to the following non-linear equation which depends on $u(x,y)$:

$$\frac{(t^2+g^2(t))g''(t)}{(1+g'^2(t))^{\frac{3}{2}}}+\frac{tg'(t)-g(t)}{\left((t^2+g^2)(1+g'^2(t))\right)^{\frac{1}{2}}}=G(t), \tag{20}$$

where $t \in (0,d)$, and

$$G(t)=\frac{1}{\pi}\left(|\partial_\tau u|^2(t,g(t))-|\partial_\tau u|^2(-t,-g(t))\right).$$

Here, $\partial_\tau u$ is the tangential derivative of $u(x,y)$ along the free boundary Γ_1. A much simplified iterative method is proposed in [25] to solve the non-linear equation (20):

$$\frac{(t^2+g_k^2(t))}{(1+g_k'^2(t))^{\frac{3}{2}}}g_{k+1}''(t)+\frac{tg_k'(t)-g_k(t)}{\left((t^2+g_k^2)(1+g_k'^2(t))\right)^{\frac{1}{2}}}=G_k(t), \tag{21}$$

where

$$g_{k+1}(0)=g_{k+1}(d)=0,$$
$$G_k(t)=\frac{1}{\pi}\left(|\partial_\tau u_k|^2(t,g_k(t))-|\partial_\tau u_k|^2(-t,-g_k(t))\right),$$

and $(t,g_k(t))$ defines the boundary of the domain for (17)–(19) whose solution is u_k. This is much simpler than the Newton iterative method and has fast convergence.

At each stage k of the iteration (21), we use the augmented method described in Section 2 to solve the Poisson equation on Ω with Dirichlet and Neumann boundary conditions on different parts of the boundary. The augmented variable is the jump in the normal derivative. It is well known that the GMRES converges very slowly since the system is similar to one using the boundary integral method with integral equations of both first and second kinds.

In the simulation, we show the performance of GMRES as well as the preconditioned one. The initial guess of the free boundary is the line connecting the top-left corner and the origin. The Dirichlet boundary condition is

$$w(x,y)=\frac{\operatorname{sgn}(y)}{2}\left((x^2+y^2)^{\frac{1}{2}}-x\right)^{\frac{1}{2}}-0.01. \tag{22}$$

Since multiple stages/iterations (21) are involved, the outcome of the previous stage GMRES iteration is used as the initial estimate of the next stage GMRES iteration. Thus, unlike the failure in the tests in the previous subsection, here, GMRES converges to modest accuracy after a certain number of steps.

Nevertheless, the preconditioned GMRES method with our preconditioner converges much faster. In Table 6, we show the total number of iterations n_{it} for GMRES at all stages, as well as the total number of matrix-vector multiplications $n_{\text{mv}}^{\text{pre}}+n_{\text{it}}$ for the preconditioned GMRES method. We construct the preconditioner once and use it for all the iterations. The stopping criterion is when the difference in a certain measurement between two consecutive stages k and $k+1$ is smaller than 10^{-6}. The tolerance for the inner GMRES iterations is also 10^{-6}. We run the tests in Fortran on a 2.5 GHz Intel Core i7 Macbook Pro with 16 GB of memory. Both the total number of GMRES iterations and the total CPU time have been greatly reduced.

The numbers of GMRES and preconditioned GMRES iterations at each stage k for the iteration (21) is also given in Table 7. With preconditioned GMRES, the solution is not only faster, but also more reliable. In fact, with just GMRES, even for the small 40×40 mesh it takes an unusually large

Table 6: Comparison of the entire simulation for the free boundary problem (17)–(19) with GMRES and preconditioned GMRES for different mesh sizes.

Mesh size	N	With GMRES		With preconditioned GMRES	
		Total n_{it}	Total CPU time (sec)	Total $n_{\text{mv}}^{\text{pre}} + n_{\text{it}}$	Total CPU time (sec)
40×40	96	*1241*	$3.74e0$	150	$5.06e-1$
80×80	114	700	$2.90e0$	164	$7.75e-1$
160×160	150	*8801*	$7.40e1$	180	$1.57e0$
320×320	222	821	$2.08e1$	183	$4.77e0$
640×640	366	979	$9.25e1$	211	$2.10e1$
1280×1280	652	951	$4.07e2$	227	$9.20e1$
2560×2560	1224	937	$1.69e3$	257	$4.75e2$

number of iterations. For the 160×160 mesh, it even fails to converge to the desired accuracy. However, with preconditioned GMRES, both the number of iterations (21) and the numbers of interior GMRES iterations are significantly reduced.

Table 7: Comparison of the numbers of iterations at all the stages k for the iteration (21).

Mesh	With GMRES	With preconditioned GMRES
40×40	$77, 79, 83, 73, 77, 81, 78, 77, \ldots, 77$ (9 times)	$24, 24, 24, 24, 24$
80×80	$70, 70, 70, 70, 70, 70, 70, 70, 70, 70$	$13, 13, 13, 13, 13, 13, 13, 13$
160×160	$89, 88, \ldots, 88$ (99 times) — divergence	$13, 17, 17, 17, 17, 17$
320×320	$83, 82, 82, 82, 82, 82, 82, 82, 82, 82$	$19, 20, 20, 20, 20, 20$
640×640	$97, 98, 98, 98, 98, 98, 98, 98, 98, 98$	$17, 23, 23, 23, 23$
1280×1280	$96, 95, 95, 95, 95, 95, 95, 95, 95, 95$	$24, 34, 34, 34$
2560×2560	$97, 94, 94, 94, 93, 93, 93, 93, 93, 93$	$25, 33, 33$

4.3. Generalizations

Other than AIIM, the preconditioning techniques can also be useful in other applications and methods such as saddle point problems, domain decomposition, hybrid solutions. See, e.g., [3, 9, 36]. In these cases, often a large problem is first solved by direct methods, and a much smaller Schur complement system corresponding to a small subdomain or interface is solved with iterative ones.

The matrix-free preconditioning techniques here are also useful for more general problems where it is difficult to form the matrix $\mathbf{A}$ explicitly or to evaluate the product of $\mathbf{A}^T$ with vectors. This also includes sparse problems such as the GeneRank problems in [41] which are nonsymmetric but have symmetric nonzero patterns or involve other types of symmetry.

5. Conclusions

AIIM has significant benefits for the fast and accurate solutions of some interface problems and problems defined on irregular domains. The efficient application of AIIM relies on the fast solution of the Schur complement system $\mathbf{Ag} = \mathbf{b}$. Since the products of $\mathbf{A}$ with vectors can be quickly evaluated, but not the entries of $\mathbf{A}$ or the products of $\mathbf{A}^T$ with vectors, we propose matrix-free preconditioning techniques to accelerate the convergence of GMRES. Rank structured techniques are combined with adaptive randomized sampling. Several improvements to existing randomized and structured algorithms are made. Various advantages of the preconditioner are demonstrated, including the flexibility, efficiency, and effectiveness, which are supported by comprehensive tests on many difficult situations. In our future work, we would like to analytically study the effectiveness of the preconditioner, at least for certain simple cases (e.g., with few blocks). We also plan to develop a parallel implementation for large-scale tests.

Acknowledgements

We are grateful to the referees for the valuable suggestions. Also thank Xiao Liu and Yuanzhe Xi for helping with the code. The research of Jianlin Xia was supported in part by an NSF CAREER Award DMS-1255416 and an NSF grant DMS-1115572. The research of Zhilin Li was partially supported by an NIH grant 5R01GM96195-2, an NSF grant DMS-1522768, and CNSF grants 11371199 and 11471166, and BK201411443.

References

[1] J. B. Bell, P. Colella, H. M. Glaz, A second-order projection method for the incompressible Navier-Stokes equations, J. Comput. Phys. 85 (1989) 257–283.

[2] S. Bellavia, J. Gondzio, B. Morini, A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems, SIAM J. Sci. Comput. 35 (2013) A192–A211.

[3] M. Benzi, G. H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numerica, 14 (2005) 1–137.

[4] J. Cullum, M. Tůma, Matrix-free preconditioning using partial matrix estimation, BIT Numer. Math. 46 (2006) 711–729.

[5] S. Le Borne, L. Grasedyck, $\mathcal{H}$-matrix preconditioners in convection-dominated problems, SIAM. J. Matrix Anal. Appl. 27 (2006) 1172–1183.

[6] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, A fast ULV decomposition solver for hierarchically semiseparable representations, SIAM J. Matrix Anal. Appl. 28 (2006) 603–622.

[7] J. Duintjer Tebbens, M. Tůma, Preconditioner updates for solving sequences of linear systems in matrix-free environment, Numer. Linear Algebra Appl. 17 (2010) 997–1019.

[8] B. Engquist, L. Ying, Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation, Pure Appl. Math. LXIV (2011) 0697–0735.

[9] L. Giraud, A. Haidar, Parallel algebraic hybrid solvers for large 3D convection-diffusion problems, Numer. Algor. 51 (2009) 151–177.

[10] J. Gondzio, Matrix-free interior point method, Comput. Optim. Appl. 51 (2012) 457–480.

[11] L. Grasedyck, R. Kriemann, S. Le Borne, Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems, Comput. Visual Sci. 11 (2008) 273–291.

[12] A. Greenbaum, V. Pták, Z. Strakoš, Any nonincreasing convergence curve is possible for GMRES, SIAM J. Matrix Anal. Appl. 17 (1996) 465–469.

[13] M. Gu, S. C. Eisenstat, Efficient algorithms for computing a strong-rank revealing QR factorization, SIAM J. Sci. Comput. 17 (1996) 848–869.

[14] M. Gu, X. S. Li, P. S. Vassilevski, Direction-preserving and Schur-monotonic semiseparable approximations of symmetric positive definite matrices, SIAM. J. Matrix Anal. Appl. 31 (2010) 2650–2664.

[15] N. Halko, P.G. Martinsson, J. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Review 53 (2011) 217–288.

[16] T. Hou, Z. Li, S. Osher, H. Zhao, A hybrid method for moving interface problems with application to the Hele-Shaw flow, J. Comput. Phys. 134 (1997) 236–252.

[17] J. Hunter, Z. Li, H. Zhao, Autophobic spreading of drops, J. Comput. Phys. 183 (2002) 335–366.

[18] R. Li, Y. Saad, Divide and conquer low-rank preconditioners for symmetric matrices, SIAM J. Sci. Comput. 35 (2013) A2069–A2095.

[19] Z. Li, A fast iterative algorithm for elliptic interface problems, SIAM J. Numer. Anal. 35 (1998) 230–254.

[20] Z. Li, Q. Cai, H. Zhao, R. Luo, A semi-implicit augmented IIM for Navier-Stokes equations with open and traction boundary conditions, submitted, 2013

[21] Z. Li, K. Ito, The Immersed Interface Method – Numerical Solutions of PDEs Involving Interfaces and Irregular Domains, SIAM Frontier Series in Applied mathematics, FR33 (2006)

[22] Z. Li, K. Ito, M-C. Lai, An augmented approach for Stokes equations with a discontinuous viscosity and singular forces, Comput. Fluids 36 (2007) 622–635.

[23] Z. Li, M-C. Lai, G. He, H. Zhao, An augmented method for free boundary problems with moving contact lines, Comput. Fluids 39 (2010) 1033–1040.

[24] Z. Li, M-C. Lai, New finite difference methods based on IIM for inextensible interfaces in incompressible flows, ESIAM of Applied Mathematics, 1 (2011) 155–171.

[25] Z. Li and H. Mikayelyan, Fine numerical analysis of the crack-tip position for a Mumford-Shah minimizer, submitted.

[26] Z. Li, H. Zhao, H. Gao, A numerical study of electro-migration voiding by evolving level set functions on a fixed cartesian grid, J. Comput. Phys. 152 (1999) 281–304.

[27] E. Liberty, F. Woolfe, P. G. Martinsson, V. Rokhlin, M. Tygert, Randomized algorithms for the low-rank approximation of matrices, Proc. Natl. Acad. Sci. USA, 104 (2007) 20167–20172.

[28] L. Lin, J. Lu, L. Ying, Fast construction of hierarchical matrix representation from matrix-vector multiplication, J. Comput. Phys. 230 (2011) 4071–4087.

[29] J. Liu, Open and traction boundary conditions for the incompressible Navier-Stokes equations, J. Comput. Phys. 228 (2009) 7250–7267.

[30] X. Liu, J. Xia, M. V. de Hoop, Parallel randomized and matrix-free direct solvers for large structured dense linear systems, SIAM J. Sci. Comput., submitted, 2015.

[31] L. Lukšan, Jan Vlček, Efficient tridiagonal preconditioner for the matrix-free truncated New-

ton method, Appl. Math. Comput. 235 (2014) 394–407.

[32] P. G. Martinsson, A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix, SIAM. J. Matrix Anal. Appl. 32 (2011) 1251–1274.

[33] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive preconditioning for matrix computations, Linear Algebra Appl. 432 (2010) 1070–1089.

[34] V. Y. Pan, G. Qian, Randomized preprocessing of homogeneous linear systems of equations, Linear Algebra Appl. 432 (2010) 3272–3318.

[35] C. S. Peskin, The immersed boundary method, Acta Numerica 11 (2002) 479–517.

[36] E. Polizzi, A. H. Sameh, A parallel hybrid banded system solver: the SPIKE algorithm, Parallel Comput. 32 (2006) 177–194.

[37] Y. Saad, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.

[38] E. Vecharynski, J. Langou, Any admissible cycle-convergence behavior is possible for restarted GMRES at its initial cycles, Num. Lin. Algebr. Appl. 18 (2011) 499–511.

[39] J. Vogel, J. Xia, S. Cauley, V. Balakrishnan, Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions, SIAM J. Sci. Comput., submitted, 2015.

[40] F. Woolfe, E. Liberty, V. Rokhlin, M. Tygert, A fast randomized algorithm for approximation of matrices, Appl. Comput. Harmon. Anal. 25 (2008) 335–366.

[41] G. Wu, W. Xu, Y. Zhang, Y. Wei, A preconditioned conjugate gradient algorithm for GeneRank with application to microarray data mining, Data Min. Knowl. Disc. 26 (2013) 27–56.

[42] Y. Xi, J. Xia, R. Chan, A fast randomized eigensolver with structured LDL factorization update, SIAM J. Matrix Anal. Appl. 35 (2014) 974–996.

[43] J. Xia, On the complexity of some hierarchical structured matrix algorithms, SIAM J. Matrix Anal. Appl. 33 (2012) 388–410.

[44] J. Xia, Randomized sparse direct solvers, SIAM J. Matrix Anal. Appl., SIAM J. Matrix Anal. Appl. 34 (2013) 197–227.

[45] J. Xia, Efficient structured multifrontal factorization for general large sparse matrices, SIAM J. Sci. Comput. 35 (2013) A832–A860.

[46] J. Xia, S. Chandrasekaran, M. Gu, X. S. Li, Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl. 17 (2010) 953–976.

[47] J. Xia, M. Gu, Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices, SIAM J. Matrix Anal. Appl. 31 (2010) 2899–2920.

[48] J. Xia, Y. Xi, M. Gu, A superfast structured solver for Toeplitz linear systems via randomized sampling, SIAM J. Matrix Anal. Appl. 33 (2012) 837–858.

[49] W.-J. Ying and C. S. Henriquez, A kernel-free boundary integral method for elliptic boundary value problems, J. Comput. Phy. 227 (2007) 1046–1074.