SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1959–1981
© 2007 Society for Industrial and Applied Mathematics

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

AN EFFICIENT SOLVER FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS IN ROTATION FORM*

MICHELE BENZI† AND JIA LIU‡

Abstract. We consider preconditioned iterative methods applied to discretizations of the linearized Navier–Stokes equations in two- and three-dimensional bounded domains. Both unsteady and steady flows are considered. The equations are linearized by Picard iteration. We make use of the rotation form of the momentum equations, which has several advantages from the linear algebra point of view. We focus on a preconditioning technique based on the Hermitian/skew-Hermitian splitting of the resulting nonsymmetric saddle point matrix. We show that this technique can be implemented efficiently when the rotation form is used. We study the performance of the solvers as a function of mesh size, Reynolds number, time step, and algorithm parameters. Our results indicate that fast convergence independent of problem parameters is achieved in many cases. The preconditioner appears to be especially attractive in the case of low viscosity and for unsteady problems.

Key words. fluid mechanics, Navier–Stokes, Krylov methods, preconditioning, rotation form, Oseen problem, Schur complement, Hermitian and skew-Hermitian splitting, generalized Stokes problem

AMS subject classifications. Primary, 65F10, 65N22, 65F50; Secondary, 76D07

DOI. 10.1137/060658825

1. Introduction.

Iterative Krylov subspace algorithms are popular methods for solving large and sparse linear systems arising in many areas of computational science and engineering [29, 34]. It is well known that in order to be effective, iterative methods must be coupled with preconditioning techniques. In the last several years, much progress has been made in the development of preconditioners tailored to the linear systems arising from linearizations and discretizations of incompressible flow problems; see [5, 36] and especially the recent monograph [12]. As a result of this effort, several effective solvers now exist for important problems like the generalized Stokes problem and the Oseen problem. Nevertheless, the performance of even the best solvers tends to deteriorate for small values of the viscosity. Furthermore, most existing methods for the Oseen problem rely on the availability of efficient solvers for convection-dominated convection-diffusion equations, which are nontrivial to develop for complex flow pattern, especially in three dimensions.

In this paper we study a preconditioning technique that is applicable to various incompressible flow problems. The preconditioner is especially well suited for the so-called *rotation form* of the Navier–Stokes equations [14, 21, 27, 37]. We present numerical results on both steady and unsteady problems in two and three space dimensions aimed at assessing the performance of the solver with respect to problem parameters such as mesh size, time step, and viscosity. These experiments indicate that the solver is effective for a wide range of problems. Implementation of the preconditioner is fairly straightforward, and it can take advantage of existing solvers

*Received by the editors May 3, 2006; accepted for publication (in revised form) January 16, 2007; published electronically September 28, 2007. This work was supported by National Science Foundation grants DMS-0207599 and DMS-0511336.
http://www.siam.org/journals/sisc/29-5/65882.html
†Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (benzi@mathcs.emory.edu).
‡Department of Mathematics and Statistics, West Florida University, Pensacola, FL 32514 (jliu@uwf.edu).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

(and software) for scalar elliptic problems. In addition, the preconditioner does not require the solution of convection-diffusion problems.

The paper is organized as follows. In section 2 we recall the standard and rotation forms of the incompressible Navier–Stokes equations in primitive variables. In section 3 we describe the discrete form of the linearized equations and briefly discuss the structure and spectral properties of the coefficient matrices. Section 4 contains a description of the preconditioning techniques used in this paper and a brief overview of related work. Numerical experiments on a selection of steady and unsteady model problems in two and three space dimensions are presented in section 5. Here we also report on the convergence of the Picard linearization for the rotation form. Some conclusions are given in section 6.

2. Incompressible fluid flow problems. We are concerned with the primitive variables formulation of the Navier–Stokes equations for a viscous, incompressible Newtonian fluid in a bounded domain $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ with boundary $\partial\Omega$:

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\, \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\, \mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega \times (0, T]\,, \tag{2.1}$$

$$\operatorname{div} \mathbf{u} = 0 \quad \text{in } \Omega \times [0, T]\,, \tag{2.2}$$

$$\mathcal{B}\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega \times [0, T]\,, \tag{2.3}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0 \quad \text{in } \Omega\,, \tag{2.4}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^d$ is the velocity field, $p = p(\mathbf{x}, t) \in \mathbb{R}$ is the pressure, $\mathbf{f} \in \mathbb{R}^d$ is a known forcing term, $\nu > 0$ is the kinematic viscosity coefficient ($\nu = O(\mathrm{Re}^{-1})$, where Re denotes the Reynolds number), Δ is the (vector) Laplace operator in d dimensions, ∇ is the gradient, div is the divergence, and $\mathcal{B}$ is some type of boundary operator (e.g., a trace operator for Dirichlet boundary conditions). Equation (2.1) will be referred to as the *convection form* of the momentum equation. The pressure field, p, is determined up to an additive constant. To uniquely determine p we may impose some additional condition, such as

$$\int_\Omega p\, d\mathbf{x} = 0\,. \tag{2.5}$$

Discretization in time with a fully implicit method (such as backward Euler, although more sophisticated methods are often used in practice) leads to a sequence of semidiscrete systems of the form

$$\sigma\mathbf{u} - \nu\, \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\, \mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega\,, \tag{2.6}$$

$$\operatorname{div} \mathbf{u} = 0 \quad \text{in } \Omega\,, \tag{2.7}$$

$$\mathcal{B}\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega\,. \tag{2.8}$$

Here $\sigma \geq 0$ is a constant, with $\sigma > 0$ in the unsteady case and $\sigma = 0$ in the steady case. Due to the presence of the inertial term $(\mathbf{u} \cdot \nabla)\, \mathbf{u}$ in the momentum equation (2.6), the Navier–Stokes equations are nonlinear. They can be linearized in various ways. A widely used scheme is Picard (fixed-point) linearization [12], which leads to a sequence of *Oseen problems*, i.e., linear problems of the form

$$\sigma\mathbf{u} - \nu\, \Delta \mathbf{u} + (\mathbf{v} \cdot \nabla)\, \mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega\,, \tag{2.9}$$

$$\operatorname{div} \mathbf{u} = 0 \quad \text{in } \Omega\,, \tag{2.10}$$

$$\mathcal{B}\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega\,. \tag{2.11}$$

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

In the momentum equation (2.9), the vector field $\mathbf{v}$ is the approximation of the solution from the previous Picard iteration. For $\mathbf{v} = \mathbf{0}$, equations (2.9)–(2.11) are known as the *generalized Stokes problem;* if in addition $\sigma = 0$, they reduce to the usual (steady) Stokes problem.

An alternative linearization can be derived based on the identity

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = (\nabla \times \mathbf{u}) \times \mathbf{u} + \frac{1}{2}\nabla(\mathbf{u} \cdot \mathbf{u}); \tag{2.12}$$

see [14, 21]. The linearization consists in approximating the right-hand side of (2.12) with the linearized expression $(\nabla \times \mathbf{v}) \times \mathbf{u} + \frac{1}{2}\nabla(\mathbf{u} \cdot \mathbf{u})$, where $\mathbf{v}$ represents the approximation of the velocity field from the previous Picard iteration. The linearized equations are

$$\sigma\mathbf{u} - \nu\,\Delta\mathbf{u} + \mathbf{w} \times \mathbf{u} + \nabla P = \mathbf{f} \quad \text{in } \Omega\,, \tag{2.13}$$

$$\operatorname{div}\mathbf{u} = 0 \quad \text{in } \Omega\,, \tag{2.14}$$

$$\mathcal{B}\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega\,. \tag{2.15}$$

In (2.13) we have set $P = p + \frac{1}{2}\mathbf{u} \cdot \mathbf{u}$ (the *Bernoulli pressure*). Moreover, in the two-dimensional (2D) case

$$(\mathbf{w}\times) = \begin{pmatrix} 0 & w \\ -w & 0 \end{pmatrix},$$

with $w = \nabla \times \mathbf{v} = -\frac{\partial v_1}{\partial x_2} + \frac{\partial v_2}{\partial x_1}$, where $\mathbf{v}$ is the approximation of the solution from the previous Picard iteration. Similarly, for three-dimensional (3D) problems

$$(\mathbf{w}\times) = \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{pmatrix},$$

where w_i denotes the ith component of the vector $\nabla \times \mathbf{v}$. Notice that no first-order terms in the velocities appear in (2.13); on the other hand, the velocities in the d scalar equations comprising (2.13) are now coupled due to the presence of the term $\mathbf{w} \times \mathbf{u}$. The disappearance of the convective terms suggests that the *rotation form* (2.13) of the momentum equations may be advantageous over the standard form (2.9) from the linear solution point of view; see also [21]. For a discussion of the excellent conservation properties of the linearization based on the rotation form, see [22]; another relevant reference is [27].

3. The discrete equations. In this paper we assume $\Omega = [0,1]^d$, and we use a standard marker-and-cell (MAC) scheme [15] with a uniform grid to discretize either the momentum form of the linearized equations (2.9)–(2.11) or the rotation form (2.13)–(2.15). However, most of what follows holds for more complicated geometries and for a variety of spatial discretization schemes, and in particular for finite elements; see [12].

The discrete system has the form

$$\begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \qquad \text{or} \qquad \mathcal{A}\,u = b\,, \tag{3.1}$$

where now $\mathbf{u}$, p, and $\mathbf{f}$ are finite-dimensional counterparts of the functions denoted by the same symbols. System (3.1) is often referred to as a generalized saddle point

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

problem; see [5]. When pressure stabilization is used (e.g., to circumvent the Babuška–Brezzi condition), a nonzero (2,2) block will generally be present in $\mathcal{A}$. Here we assume that a div-stable method (like MAC) is being used, so that no stabilization is needed. However, much of what follows admits a straightforward extension to the case where the (2,2) block of $\mathcal{A}$ consists of a symmetric negative semidefinite matrix.

As is well known, for small values of the viscosity ν velocity stabilization becomes necessary in order to prevent spurious (nonphysical) oscillations in the discrete solution corresponding to all but the finest meshes. For the time being, we sidestep the issue and assume that a sufficiently fine mesh is used. Note that the same assumption is made (for Galerkin FEM discretizations) in [12, p. 327]. We return to this topic in the section on numerical experiments.

The B block in (3.1) represents a discrete divergence operator. In the 2D case, $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$, where B_1 is a discretization of $\frac{\partial}{\partial x}$ and B_2 is a discretization of $\frac{\partial}{\partial y}$. The form of the (1,1) block A depends on whether the convection form or the rotation form of the momentum equation is used. For the convection form we have, in the 2D case,

$$A = \begin{bmatrix} \sigma I + \nu L_1 + S_1 & O \\ O & \sigma I + \nu L_2 + S_2 \end{bmatrix},$$

where L_1, L_2 are discrete (negative) Laplacians and S_1, S_2 represent discretizations of the components of the convection term $(\mathbf{v} \cdot \nabla)\,\mathbf{u}$. In this paper we assume that the convective terms are skew-symmetric: $S_1 = -S_1^T$, $S_2 = -S_2^T$. This assumption is satisfied, for example, if the boundary conditions are of Dirichlet type only, or if Neumann boundary conditions are present on the outflow and characteristic boundaries only; see [12] for a detailed discussion of this issue, especially pp. 120–121 and 151. In the 3D case, there will be three diagonal blocks of the same form rather than two.

If the rotation form is used we will have, in the 2D case,

$$A = \begin{bmatrix} \sigma I + \nu L_1 & D \\ -D & \sigma I + \nu L_2 \end{bmatrix},$$

where the matrix D represents multiplication by the discretized function $w = \nabla \times \mathbf{v}$. For MAC discretization, D is a just a diagonal matrix containing on the main diagonal the values of w on the grid edges. In the 3D case,

$$A = \begin{bmatrix} \sigma I + \nu L_1 & -D_3 & D_2 \\ D_3 & \sigma I + \nu L_2 & -D_1 \\ -D_2 & D_1 & \sigma I + \nu L_3 \end{bmatrix},$$

where now D_i is the matrix that corresponds to multiplication by w_i, the ith component of $\nabla \times \mathbf{v}$. For finite element discretizations, the I and D matrices will be replaced by appropriate (weighted) mass matrices which may no longer be diagonal.

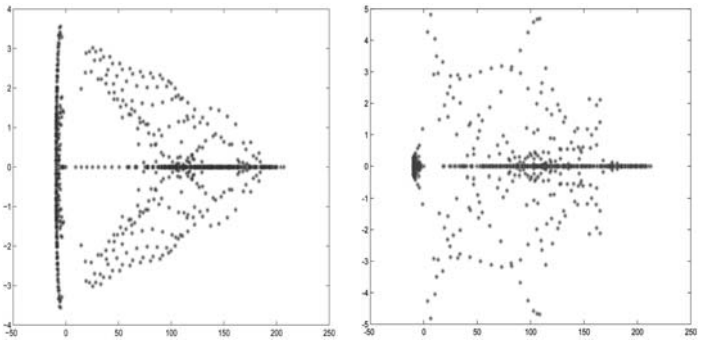
We observe that for both the convection and the rotation form we can write $A = H + K$, where (2D case)

$$H = \frac{1}{2}(A + A^T) = \begin{bmatrix} \sigma I + \nu L_1 & O \\ O & \sigma I + \nu L_2 \end{bmatrix}$$

is symmetric positive definite for all $\sigma \geq 0$. The matrix $K = \frac{1}{2}(A - A^T)$ is skew-symmetric, and its structure will depend on the form of the momentum equation. In the case of the (generalized) Stokes problem, $K = O$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php



(a) Convection form (b) Rotation form

FIG. 3.1. *Eigenvalues for 2D linearized Navier–Stokes problem using MAC discretization (indefinite form).*

With either form of the momentum equation the saddle point matrix $\mathcal{A}$ is highly indefinite, in the sense that it has many eigenvalues on either side of the imaginary axis. Figure 3.1(a) displays the eigenvalues of the discrete steady Oseen operator (convection form, $\nu = 0.01$) obtained from a MAC discretization on $\Omega = [0,1]\times[0,1]$ on a 16×16 grid. As the wind function, we used $\mathbf{v} = (8x(x-1)(1-2y), 8(2x-1)y(y-1))$. Figure 3.1(b) displays the eigenvalues of the discrete operator corresponding to the rotation form on the same grid, with $\nu = 0.01$ and $w = 16x(x-1) + 16y(y-1) = \nabla \times \mathbf{v}$. As can be seen, both matrices have many eigenvalues with negative real part. Additional algebraic properties of saddle point matrices are discussed in [5, sect. 3].

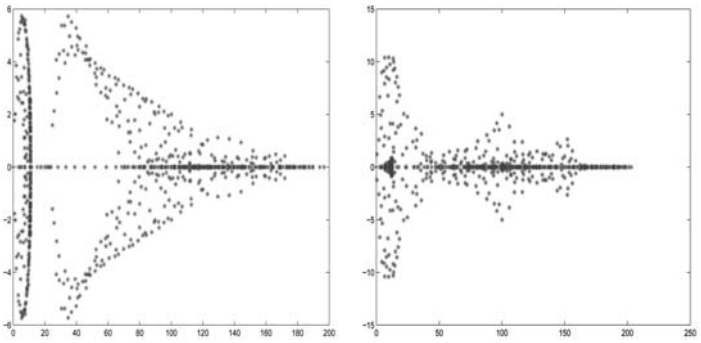
Eigenvalue distributions such as those shown in Figures 3.1(a)–3.1(b) are generally unfavorable for solution by Krylov subspace methods, and, indeed, without preconditioning, Krylov subspace methods converge very slowly (or not at all) when applied to the corresponding linear systems. It has been observed by several authors (see [5, sect. 3.4]) that a simple transformation can be used to obtain an equivalent linear system with a coefficient matrix whose spectrum is entirely contained in the half-plane $\Re(z) > 0$. (Here we use $\Re(z)$ and $\Im(z)$ to denote the real and imaginary parts of $z \in \mathbb{C}$.) Indeed, we can rewrite the saddle point system in the equivalent form

$$\begin{bmatrix} A & B^T \\ -B & O \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \qquad \text{or} \qquad \hat{\mathcal{A}}\, u = \hat{b}\,. \tag{3.2}$$

Figures 3.2(a)–(b) display the eigenvalues of the matrix $\hat{\mathcal{A}}$ corresponding to the same linearized problems as before (convection and rotation form, respectively). As can be seen, all the eigenvalues lie in the right half-plane. Matrices with this property are said to be *positive stable*. A good preconditioner should preserve this property while at the same time clustering the eigenvalues away from the origin. Ideally, most of the eigenvalues should be in the cluster, and the cluster itself should be bounded away from zero independently of the mesh size h and the viscosity ν. A few outliers do not typically constitute a problem. In general, ν-independence is harder to achieve than h-independence; see [12].

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php



(a) Convection form (b) Rotation form

FIG. 3.2. *Eigenvalues for 2D linearized Navier–Stokes problem using MAC discretization (positive definite form).*

4. Preconditioner description. In this section we introduce and motivate our proposed preconditioning strategy for the linearized Navier–Stokes equations in rotation form, including inexact variants. We also (briefly) mention related work by other authors.

4.1. HSS preconditioner and its variants. The Hermitian/skew-Hermitian splitting (HSS) preconditioner is based on the nonsymmetric formulation (3.2). Letting $H \equiv \frac{1}{2}(A + A^T)$ and $K \equiv \frac{1}{2}(A - A^T)$, we have the following splitting of $\hat{\mathcal{A}}$ into its symmetric and skew-symmetric parts:

$$\hat{\mathcal{A}} = \begin{bmatrix} A & B^T \\ -B & O \end{bmatrix} = \begin{bmatrix} H & O \\ O & O \end{bmatrix} + \begin{bmatrix} K & B^T \\ -B & O \end{bmatrix} = \mathcal{H} + \mathcal{K}\,. \tag{4.1}$$

Note that $\mathcal{H}$, the symmetric part of $\hat{\mathcal{A}}$, is symmetric positive semidefinite since H is. Let $\alpha > 0$ be a parameter. Similar in spirit to the classical alternating-direction implicit (ADI) method, we consider the following two splittings of $\hat{\mathcal{A}}$:

$$\hat{\mathcal{A}} = (\mathcal{H} + \alpha\mathcal{I}) - (\alpha\mathcal{I} - \mathcal{K}) \qquad \text{and} \qquad \hat{\mathcal{A}} = (\mathcal{K} + \alpha\mathcal{I}) - (\alpha\mathcal{I} - \mathcal{H})\,.$$

Here $\mathcal{I}$ denotes the identity matrix of order $n + m$. Note that

$$\mathcal{H} + \alpha\mathcal{I} = \begin{bmatrix} H + \alpha I_n & O \\ O & \alpha I_m \end{bmatrix}$$

is symmetric positive definite with eigenvalues bounded below by $\alpha > 0$. Similarly,

$$\mathcal{K} + \alpha\mathcal{I} = \begin{bmatrix} K + \alpha I_n & B^T \\ -B & \alpha I_m \end{bmatrix}$$

is nonsingular and has positive definite symmetric part. Its eigenvalues are of the form $\alpha \pm \mathrm{i}\mu$ with $\mu \in \mathbb{R}$, where $\mathrm{i} = \sqrt{-1}$.

Alternating between these two splittings leads to the stationary HSS iteration:

$$\begin{cases} (\mathcal{H} + \alpha\mathcal{I})\, u_{k+\frac{1}{2}} = (\alpha\mathcal{I} - \mathcal{K})\, u_k + \hat{b}, \\ (\mathcal{K} + \alpha\mathcal{I})\, u_{k+1} = (\alpha\mathcal{I} - \mathcal{H})\, u_{k+\frac{1}{2}} + \hat{b} \end{cases} \tag{4.2}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

($k = 0, 1, \dots$). Here $\hat{b}$ denotes the right-hand side of (3.2); the initial guess u_0 is chosen arbitrarily. Elimination of $u_{k+\frac{1}{2}}$ from (4.2) leads to a stationary (fixed-point) iteration of the form

$$u_{k+1} = \mathcal{T}_\alpha u_k + c\,, \quad k = 0, 1, \dots,$$

where $\mathcal{T}_\alpha = (\mathcal{K} + \alpha\mathcal{I})^{-1}(\alpha\mathcal{I} - \mathcal{H})(\mathcal{H} + \alpha\mathcal{I})^{-1}(\alpha\mathcal{I} - \mathcal{K})$ is the iteration matrix and c is a certain vector. The iteration converges for arbitrary initial guesses u_0 and right-hand sides $\hat{b}$ to the solution $u_* = \hat{\mathcal{A}}^{-1}\hat{b}$ if and only if $\varrho(\mathcal{T}_\alpha) < 1$, where $\varrho(\mathcal{T}_\alpha)$ denotes the spectral radius of $\mathcal{T}_\alpha$. This stationary iteration was first introduced in [2], where convergence was proved for systems of the form $Ax = b$ with A *positive real* (i.e., the symmetric part of A is positive definite). In the same paper, the authors studied the optimal choice of the parameter α and the effect of inexact solves in (4.2). The extension of this method to saddle point systems (for which the symmetric part is usually singular) was first given in [4]; in this paper it was also observed that the rate of convergence of the stationary HSS iteration is often very slow (even with the optimal choice of the parameter α) and that the method should be used as a preconditioner for a Krylov subspace method rather than as a solver.

It follows from the general theory in [7] that there is a unique splitting $\hat{\mathcal{A}} = \mathcal{P} - \mathcal{Q}$ with $\mathcal{P}$ nonsingular such that the iteration matrix $\mathcal{T}_\alpha$ is the matrix induced by that splitting, i.e., $\mathcal{T}_\alpha = \mathcal{P}^{-1}\mathcal{Q} = \mathcal{I} - \mathcal{P}^{-1}\hat{\mathcal{A}}$. An easy calculation shows that the HSS iteration (4.2) can be written in *correction form* as

$$u_{k+1} = u_k + \mathcal{P}^{-1} r_k, \quad r_k = \hat{b} - \hat{\mathcal{A}} u_k\,,$$

where the preconditioner $\mathcal{P}$ is given by

$$\mathcal{P} \equiv \mathcal{P}_\alpha = \tfrac{1}{2\alpha}(\mathcal{H} + \alpha\mathcal{I})(\mathcal{K} + \alpha\mathcal{I})\,. \tag{4.3}$$

Note that as a preconditioner we can use $\mathcal{P}_\alpha = (\mathcal{H} + \alpha\mathcal{I})(\mathcal{K} + \alpha\mathcal{I})$ instead of the expression given in (4.3), since the factor $\frac{1}{2\alpha}$ has no effect on the preconditioned system. It is just a normalization factor that allows us to conclude that $\varrho(\mathcal{T}_\alpha) < 1$; hence, the eigenvalues of the preconditioned matrix $\mathcal{P}_\alpha^{-1}\hat{\mathcal{A}}$ (or $\hat{\mathcal{A}}\,\mathcal{P}_\alpha^{-1}$, which has the same spectrum) are all contained in the disk $\{z \in \mathbb{C} : |z - 1| < 1\}$. In particular, the spectrum of the preconditioned matrix, like that of $\hat{\mathcal{A}}$, lies entirely in the right half-plane: the preconditioned matrix is positive stable.

Spectral properties of the preconditioned matrix as a function of α have been studied, under different sets of assumptions, in [3, 33]. A Fourier analysis of HSS preconditioning for saddle point formulations of Poisson's equation (including the anisotropic case) was given in [3]. The analysis showed that using a sufficiently small value of α results in h-independent convergence. Furthermore, as $\alpha \to 0+$ the eigenvalues of the preconditioned matrix are all real and fall within two small intervals $(0, \varepsilon_1)$ and $(2 - \varepsilon_2, 2)$, with $\varepsilon_1, \varepsilon_2 > 0$ and $\varepsilon_1, \varepsilon_2 \to 0$ as $\alpha \to 0+$. This clustering result was generalized in [33] to general saddle point systems with $A = A^T$ positive definite using purely algebraic arguments; see also [6]. Numerical experiments show that with an appropriate scaling of the system (such that the nonzero diagonal entries of $\hat{\mathcal{A}}$ are equal to 1), there is a unique value α_* of α for which the number of preconditioned iterations is minimized, and this α_* is usually a small number, between 0 and 1. However, h-independent convergence is not always guaranteed; for example, it does not occur for the *steady* Stokes problem, as we shall see.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

No rigorous analysis exists for the nonsymmetric ($A \neq A^T$) case. Here the only available result is that $\varrho(\mathcal{T}_\alpha) < 1$ for all $\alpha > 0$. Nevertheless, the following informal argument suggests that good performance of the HSS preconditioner applied to linearized Navier–Stokes problems should be expected for sufficiently small ν and α. Let us assume that $A = \nu L + K$, where $\frac{1}{2}(A + A^T) = H = \nu L$ is positive definite and K is skew-symmetric; note that this is certainly the case for the linearizations of the steady Navier–Stokes equations discussed in section 2, where L is a discrete vector Laplacian in $\mathbb{R}^d$, and K contains the skew-symmetric contribution from the convective or rotation terms. We have

$$\mathcal{P}_\alpha = \begin{bmatrix} \nu L + \alpha I & O \\ O & \alpha I \end{bmatrix} \begin{bmatrix} K + \alpha I & B^T \\ -B & \alpha I \end{bmatrix} = \begin{bmatrix} \alpha A + \nu L K + \alpha^2 I & \alpha B^T + \nu L B^T \\ -\alpha B & \alpha^2 I \end{bmatrix}.$$

Therefore, as $\nu \to 0$ the (scaled) preconditioner $\alpha^{-1}\mathcal{P}_\alpha$ approaches the matrix $\hat{\mathcal{A}} + \alpha\mathcal{I}$. For small α this is close to $\hat{\mathcal{A}}$, and we conclude that the HSS preconditioner may be interpreted as an *approximate block factorization* of the coefficient matrix $\hat{\mathcal{A}}$, in which the approximation error $\|\alpha^{-1}\mathcal{P}_\alpha - \hat{\mathcal{A}}\|$ decreases as $\nu, \alpha \to 0+$. (This holds true for both the convection and the rotation form of the linearized Navier–Stokes equations.) In practice, however, the shift α should not be taken too small, as the preconditioner becomes singular for $\alpha = 0$. The choice of α is an important issue which we defer until section 5.

In the unsteady case ($\sigma > 0$ in (2.9)), direct computation of $\alpha^{-1}\mathcal{P}_\alpha - \hat{\mathcal{A}}$ shows that the HSS preconditioner does not yield, as ν and α become smaller, a good approximation, except when σ is very small. However, it is possible to remedy the situation by making a small modification to the preconditioner. Observing that $H = \sigma I_n + \nu L$, we consider the following splitting of $\hat{\mathcal{A}}$:

$$\hat{\mathcal{A}} = \begin{bmatrix} \nu L & O \\ O & O \end{bmatrix} + \begin{bmatrix} K + \sigma I & B^T \\ -B & O \end{bmatrix} = \mathcal{H}' + \mathcal{K}'. \tag{4.4}$$

Note that $\mathcal{H}'$ is still symmetric, but $\mathcal{K}'$ is no longer skew-symmetric unless $\sigma = 0$. Now let $\alpha > 0$ be a shift and consider the preconditioner $\mathcal{P}_\alpha = (\mathcal{H}' + \alpha\mathcal{I})(\mathcal{K}' + \alpha\mathcal{I})$. Expanding the product shows that as $\nu, \alpha \to 0+$, the scaled preconditioner $\alpha^{-1}\mathcal{P}_\alpha$ approaches $\hat{\mathcal{A}}$. It can be shown (see [19]) that for this splitting as well, the spectral radius of the corresponding iteration matrix satisfies $\varrho(\mathcal{T}_\alpha) < 1$ for all $\alpha > 0$ and for all $\sigma \geq 0$; this result generalizes Theorem 3.1 in [4]. Furthermore, for $\sigma > 0$ sufficiently large the preconditioner yields a good approximation even for the generalized (i.e., unsteady) Stokes and Oseen-type problems. These observations are corroborated by the numerical results in section 5.

Since for $\sigma = 0$ the preconditioner based on the splitting (4.4) reduces to the standard HSS preconditioner, from now on we will use the name "HSS preconditioner" for the method based on the splitting (4.4), and we will write $\mathcal{H}$, $\mathcal{K}$ instead of $\mathcal{H}'$, $\mathcal{K}'$.

4.1.1. Application of the preconditioner. Application of the HSS preconditioner within a Krylov subspace method (e.g., GMRES [30]) requires solving a linear system of the form $\mathcal{P}_\alpha z_k = r_k$ at each iteration. This is done by first solving the system

$$(\mathcal{H} + \alpha\mathcal{I})\, w_k = r_k \tag{4.5}$$

for w_k, followed by

$$(\mathcal{K} + \alpha\mathcal{I})\, z_k = w_k\,. \tag{4.6}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

The first system requires solving systems with coefficient matrix $H+\alpha I$. This consists of d $(= 2, 3)$ decoupled subsystems (one for each spatial direction) with symmetric positive definite coefficient matrix, regardless of whether the rotation or the convection form of the equations is used. A number of standard methods can be deployed, including sparse Cholesky factorization, preconditioned conjugate gradient schemes, or multigrid methods, either geometric or algebraic. Iterative methods can be expected to work quite well, since each system consists of a shifted Poisson-type equation. Note that owing to the diagonal scaling used, the addition of a positive shift to the main diagonal of H leads to a condition number

$$\kappa(H+\alpha I) = \frac{\lambda_{\max}(H)+\alpha}{\lambda_{\min}(H)+\alpha} < 1 + \frac{\lambda_{\max}(H)}{\alpha} = O(\alpha^{-1}),$$

since $\lambda_{\max}(H)$ is bounded independently of h. A typical value of α is usually around 0.1 or larger, resulting in well-conditioned systems and consequently fast rates of convergence of iterative methods applied to (4.5).

Solving (4.6) is somewhat more involved. It requires the solution of a linear system of the form

$$\begin{bmatrix} K+\gamma I & B^T \\ -B & \alpha I \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} f_k \\ g_k \end{bmatrix}, \tag{4.7}$$

where we have set $\gamma = \sigma+\alpha$. This system can be solved in several ways. One approach is to eliminate x_k from the second equation using the first one (Schur complement reduction), leading to a smaller (order m) linear system of the form

$$[B(K+\gamma I)^{-1}B^T + \alpha I]y_k = B(K+\gamma I)^{-1}f_k + \alpha g_k. \tag{4.8}$$

Once the solution y_k to (4.8) has been computed, the vector x_k is given by

$$(K+\gamma I)x_k = f_k - B^T y_k\,.$$

When $K = O$, system (4.8) simplifies to

$$(BB^T + \alpha\gamma I)\, y_k = Bf_k + \alpha\gamma g_k, \tag{4.9}$$

and $x_k = \gamma^{-1}(f_k - B^T y_k)$. Note that BB^T is essentially a discrete Laplacian, and that $\gamma > 0$; therefore, system (4.9) can be solved efficiently. When $K \neq O$ the matrix in (4.8) is dense for the convective form of the Navier–Stokes equations but sparse for the rotation form. As this is a crucial point, we discuss it in some detail. Recall that in the 2D case the system matrix for the linearization of the rotation form is

$$\begin{bmatrix} A_1 & D & B_1^T \\ -D & A_2 & B_2^T \\ -B_1 & -B_2 & O \end{bmatrix} = \begin{bmatrix} \nu L_1 & O & O \\ O & \nu L_2 & O \\ O & O & O \end{bmatrix} + \begin{bmatrix} \sigma I & D & B_1^T \\ -D & \sigma I & B_2^T \\ -B_1 & -B_2 & O \end{bmatrix} = \mathcal{H} + \mathcal{K}\,.$$

Here the L_i are discrete Laplace operators with appropriate boundary conditions, and $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$; $D = D^T$ is a matrix that corresponds to multiplication by $w = \nabla \times \mathbf{v}$. For finite difference schemes (like MAC) D is a diagonal matrix; for finite elements, it will be a scaled mass matrix. When D is diagonal, the Schur complement $B(K+\gamma I)^{-1}B^T + \alpha I$ is a sparse matrix and can be formed explicitly. This follows from the fact that

$$(I+\gamma^{-1}K)^{-1} = \begin{bmatrix} I & \gamma^{-1}D \\ -\gamma^{-1}D & I \end{bmatrix}^{-1} = \begin{bmatrix} E_1 & -E_2 \\ E_2 & E_3 \end{bmatrix},$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

where E_1, E_2, and E_3 are diagonal matrices given by $E_1 = I - \gamma^{-2} D(I + \gamma^{-2} D^2)^{-1} D$, $E_2 = \gamma^{-1} D(I + \gamma^{-2} D^2)^{-1}$, and $E_3 = (I + \gamma^{-2} D^2)^{-1}$. This can be verified directly, or also by noting that $I + \gamma^{-1} K = PGP^T$, where G is a block diagonal matrix with two-by-two blocks and P is the odd-even permutation matrix. Thus, $(I + \gamma^{-1} K)^{-1} = PG^{-1}P^T$ has exactly the same sparsity structure as $I + \gamma^{-1} K$.

When D is not diagonal, we can replace it with a (spectrally equivalent) diagonal approximation and still have a sparse Schur complement; since we are constructing a preconditioner, the action of $(\mathcal{K} + \alpha \mathcal{I})^{-1}$ need not be computed exactly. Hence, the Schur complement $B(K + \gamma I)^{-1} B^T + \alpha I$ (or the approximation of it obtained by replacing D with a diagonal matrix) is sparse, and system (4.6) can be efficiently solved via (4.8) by explicitly forming the sparse Schur complement system and solving it by appropriate sparse matrix techniques. Standard options include sparse LU factorization or nonsymmetric Krylov subspace methods with an appropriate choice of the preconditioner. It is also possible to use multigrid methods, since the Schur complement can be interpreted as a discretization of a second-order elliptic operator with variable coefficients. Indeed, the matrix $B(I + \gamma^{-1} K)^{-1} B^T$ can be regarded as a discretization of the differential operator

$$\mathcal{L} = -\mathrm{div}\,(k^{-1} \nabla)\,, \quad \text{where} \quad k = \begin{pmatrix} 1 & \gamma^{-1} w \\ -\gamma^{-1} w & 1 \end{pmatrix}, \quad w = w(\mathbf{x}). \tag{4.10}$$

This operator is not self-adjoint (except in the case $w =$ constant), but has positive definite symmetric part; in other words, it is strongly elliptic. Efficient multigrid methods exist for this type of problem; see [22] for an approach especially well suited for this problem. Clearly, the larger γ is, the easier it is to solve the Schur complement system.

While we have focused our description on the 2D case, the 3D case can be treated along similar lines. The system matrix is now

$$\hat{\mathcal{A}} = \begin{bmatrix} \nu L_1 & O & O & O \\ O & \nu L_2 & O & O \\ O & O & \nu L_3 & O \\ O & O & O & O \end{bmatrix} + \begin{bmatrix} \sigma I & -D_3 & D_2 & B_1^T \\ D_3 & \sigma I & -D_1 & B_2^T \\ -D_2 & D_1 & \sigma I & B_3^T \\ -B_1 & -B_2 & -B_3 & O \end{bmatrix} = \mathcal{H} + \mathcal{K}\,.$$

The Schur complement is now given by

$$\alpha I + B(I + \gamma^{-1} K)^{-1} B^T = \alpha I + B \begin{bmatrix} F_1 & -E_3 & E_2 \\ E_3 & F_2 & -E_1 \\ -E_2 & E_1 & F_3 \end{bmatrix} B^T,$$

where F_1, F_2, F_3, E_1, E_2, and E_3 are easily computed diagonal matrices. We refer to [19] for additional details. Again, the Schur complement system is sparse and can be solved using standard sparse matrix solvers or a multigrid method.

4.1.2. Inexact variants. In practice, the solves (4.5) and (4.6)–(4.8) need not be performed to high accuracy. Rather than "exact" solves, inexact solves can be performed by inner iterative schemes applied to (4.5) and (4.8). Our experience is that the rate of convergence of the outer Krylov subspace iteration is scarcely affected by the use of inexact inner solves, and that a fairly low accuracy in the inner solves is sufficient to retain good rates of convergence of the outer iteration. This is especially important for very large problems, where exact inner solves would not be feasible.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

A distinction has to be made depending on the type of solvers used in the inner iterations. If the inexact inner solves consist of a fixed number of steps of a stationary iterative method (such as multigrid), then the (inexact) HSS preconditioner remains constant from outer iteration to outer iteration and a standard Krylov subspace method, such as GMRES, can be used for the outer iteration. On the other hand, if the inner iterations are terminated when a reduction of the inner residual norms by a prescribed amount has been reached, or if the inner iterations consist of a few steps of a nonstationary method (such as conjugate gradients), then the inexact HSS preconditioner changes from one outer iteration to the next, and a flexible method (such as FGMRES [28]) must be used for the outer iteration.

We reiterate that the linear systems arising from (4.5) are well conditioned and are easy to solve approximately. In particular, approximate solutions of (4.5) can be obtained in $O(n)$ operations using multigrid or preconditioned conjugate gradients. For the Schur complement system (4.9) arising in the case of Stokes and generalized Stokes problems, similar techniques can be used to obtain an approximate solution in $O(m)$ work. For the nonsymmetric (Oseen-type) case, the Schur complement system is the nonsymmetric problem (4.8). In this case one could use GMRES with ILU preconditioning, but multigrid is a viable option here as well; see [22].

The above discussion assumes that the rotation form of the equations is used. For the standard (convective) form of the Navier–Stokes equations, the solution of (4.6), whether exact or inexact, appears to be problematic.

4.2. Related work. Olshanskii has proposed an effective block triangular preconditioner for the rotation form of the (linearized) Navier–Stokes equations; see [21, 22, 23] as well as [24] and [20]. His method is quite robust with respect to both h and ν. The preconditioner has the form

$$\mathcal{P} = \begin{bmatrix} \hat{A} & B^T \\ O & \hat{S} \end{bmatrix}, \tag{4.11}$$

where $\hat{A}$ and $\hat{S}$ are not explicitly computed matrices but implicit approximations to the (1,1) block of the saddle point system and of the Schur complement, respectively. Application of the preconditioner within an outer Krylov subspace iteration requires evaluating the action of $\hat{A}^{-1}$ and of $\hat{S}^{-1}$ on given vectors. For the action of $\hat{A}^{-1}$, a suitable multigrid method has been developed and analyzed in [24]. Note that the (1,1) block A can be interpreted as a coupled system of reaction-diffusion equations for the two (three in three dimensions) components of the velocity field. Clearly, this is a more complicated problem than inverting the uncoupled shifted velocity Laplacians in the HSS approach; see (4.5).

The approximate Schur complement step requires inexact solves for the elliptic problem (4.10) already mentioned in the previous subsection. We refer the reader to [21, 23] for details. The cost of this operation is comparable to the cost of solving the Schur complement system (4.8) in the HSS preconditioner; in the latter method, however, the system to be solved is somewhat easier due to the presence of the shift parameter α, which results in linear systems with eigenvalues that are better separated from the imaginary axis. In particular, for small σ the elliptic problem (4.10) tends to become singular and must be regularized by a suitable diagonal shift in the tensor k appearing in (4.10). In our experience, the performance of the preconditioner can be affected rather strongly by the choice of the shift parameter.

In summary, Olshanskii's preconditioner requires specialized solvers for the subproblems corresponding to both the (1,1) block and the approximate Schur comple-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

ment, and applying this preconditioner within one step of an outer Krylov method is somewhat more expensive than with the HSS preconditioner.

It must be noted that several other preconditioners for the discrete Navier–Stokes equations can be cast as approximate (block) factorizations of the system matrix. Furthermore, many algebraic splitting methods for the unsteady Navier–Stokes equations can be interpreted in terms of approximate block factorizations; this is the case, for instance, for the classical fractional step method and for the popular SIMPLE scheme and its variants; see, e.g., the recent references [9, 18, 26, 31, 35, 36]. These splittings naturally lead to preconditioners for Krylov methods, for both steady and unsteady problems [12, Chapter 8.3.5]. While these preconditioners have been designed for the convection form, some of them may be formally extended to the rotation form of the linearized Navier–Stokes equations; this is the case, for instance, for Elman's so-called BFB^T preconditioner [10]. Regardless of which form of the equations is used, however, these preconditioning schemes do not yield better approximations as the viscosity ν approaches zero; on the contrary, the quality of the approximation tends to deteriorate as $\nu \to 0$, especially for stationary problems. This is an important difference between these earlier approaches and the HSS and Olshanskii-type preconditioners.

Furthermore, most of the block preconditioners for the standard form of the Navier–Stokes equations depend on the availability of effective solvers (exact or approximate) for convection-dominated convection-diffusion equations, which are not easily developed. In contrast, no such solvers are needed when the rotation form of the equations is used.

5. Numerical experiments. In this section we report on several numerical experiments meant to illustrate the behavior of the HSS preconditioner on a wide range of model problems. We consider both Stokes and Oseen-type problems, steady and unsteady, in two and three dimensions. The use of inexact solves is also discussed. Our results include iteration counts, as well as some timings and eigenvalue plots. We further include a few experiments illustrating the convergence of Picard's iteration for the rotation form. All results were computed in MATLAB 7.1.0 on one processor of an AMD Opteron with 32 GB of memory. Many additional experimental results can be found in [19]; the results presented here are but a representative sampling of those in [19].

In all experiments, a symmetric diagonal scaling was applied before forming the preconditioner, so as to have all the nonzero diagonal entries of the saddle point matrix equal to 1. We found that this scaling is beneficial to convergence and makes finding (nearly) optimal values of the shift α easier. Of course, the right-hand side and the solution vector were scaled accordingly. We found, however, that without further preconditioning Krylov subspace solvers converge extremely slowly on all the problems considered here. Right preconditioning was used in all cases.

5.1. Oseen-type equations in rotation form. Here we consider linear systems arising from the discretization of the linearized Navier–Stokes equations in rotation form. For lack of a better name we will refer to the linear problem (2.13)–(2.15) as the *Oseen problem in rotation form.* The computational domain is the unit square $\Omega = [0,1] \times [0,1]$ (or the unit cube $\Omega = [0,1] \times [0,1] \times [0,1]$ for 3D problems). Homogeneous Dirichlet boundary conditions are imposed on the velocities; experiments with different boundary conditions were also performed, with results similar to those reported below. We experimented with different forms of the divergence-free field $\mathbf{v}$ appearing (via $\mathbf{w} = \nabla \times \mathbf{v}$) in (2.13). Here we present results for the choice

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 5.1
Iteration counts for 2D steady Oseen problem with different values of ν (exact solves).

Grid	n	m	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\nu = 0.0001$
16×16	480	256	20	16	14	14
32×32	1,984	1,024	24	25	20	14
64×64	8,064	4,096	31	36	30	22
128×128	33,024	16,384	43	50	50	34
256×256	130,560	65,536	58	66	79	52

$w = 16x(x-1) + 16y(y-1)$ (2D case) and $\mathbf{w} = (-4z(1-2x),\, 4z(2y-1),\, 2y(1-y))$ (3D case). The 2D case corresponds to the choice of $\mathbf{v}$ used in [11]. The equations were discretized with a MAC scheme with a uniform mesh size h. The outer iteration (full GMRES) was stopped when

$$\frac{||r_k||_2}{||r_0||_2} < 10^{-6},$$

where r_k denotes the residual vector at step k. For the results presented in this section, the symmetric positive definite systems (4.5) were solved "exactly" by means of the sparse Cholesky factorization available in MATLAB, in combination with an approximate minimum degree ordering [1] to reduce fill-in. For the sparse, nonsymmetric Schur complement system (4.8) we used the sparse LU solver available in MATLAB with the original (lexicographic) ordering. We found this to be faster than a minimum degree ordering, probably because the need for pivoting makes the fill-reducing ordering ineffective or even harmful. We defer a discussion of the effect of inexact solves to section 5.3.

When working with the usual (convection) form of the Oseen problem, it is well known that care must be taken to avoid potential instabilities in the discrete velocities, either by taking h sufficiently small (compared to ν and $\|\mathbf{v}\|_\infty$), or by adding suitable stabilization terms to the discrete equations. Velocity stabilization is an issue for the rotation form of the equations as well; see [20, 22] for a discussion within the finite element context. For the experiments in this section we ignore the issue of velocity stabilization and assume that a sufficiently fine mesh is used; the same point of view is taken in [12, Chapter 7.3]. Whenever we present results on a sequence of grids, we do so only as a means of investigating the scalability of the linear solver as $h \to 0$, assuming that the finest mesh is sufficiently fine for the computed solution to be meaningful. This has indeed been verified by using test problems with a known solution.

First we consider a 2D steady ($\sigma = 0$) Oseen-type problem. In Table 5.1 we report iteration counts for a sequence of increasingly refined grids and for different values of the viscosity ν. The number of velocity and pressure unknowns (n and m, respectively) is also included. A value of the parameter α close to the optimal one was used in all cases. We found that for steady problems, the best value of α is of the form $\alpha = ch$, where c is independent of h but depends (weakly) on ν. A good rule of thumb is $c = -4 \log_{10} \nu$; that is, $c = 4$ for $\nu = 0.1$, and $c = 16$ for $\nu = 0.0001$. Thus, it is possible to determine a good choice of the shift parameter in this way: first one finds the optimal α by solving a few small problems on a coarse grid with mesh size h; then the optimal value of α for a fine grid with mesh size h/ℓ can be estimated by dividing the value of α by ℓ. We note that for every problem we have solved, there is a unique optimal value of α.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 5.2
Results for 2D unsteady Oseen problem with $\sigma = 40$ and different values of ν (exact solves).

Grid	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\nu = 0.0001$
16×16	16	17	17	16
32×32	16	17	17	17
64×64	22	21	21	17
128×128	30	21	20	16
256×256	38	23	22	16

TABLE 5.3
Iteration count as a function of α for 2D Oseen problem, $\sigma = 40$, $\nu = 0.001$, 256×256 grid.

α	0.01	0.05	0.08	0.10	0.20	0.25	0.50	1.00
Its	65	32	27	25	22	22	28	50

TABLE 5.4
Iteration count for 3D generalized Oseen problem (exact solves, $\alpha = 0.1$).

σ	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\nu = 0.0001$
1	34	31	32	26
10	25	22	22	23
20	24	22	22	23
50	23	22	23	24
100	24	24	24	24

It is clear from the results in Table 5.1 that for this set of (steady) problems, the rate of convergence with HSS preconditioning suffers from some degradation as $h \to 0$; on the other hand, the convergence in all cases is pretty fast and tends to improve (albeit not monotonically) as ν becomes smaller. The robustness of HSS preconditioning with respect to ν has been observed in all the tests we performed, and in our opinion this is enough to justify interest in this preconditioning technique.

In Table 5.2 we present results for a quasi-steady problem with $\sigma = 40$ and $\nu = 0.001$. We can see from this table that HSS preconditioning results in fast convergence in all cases, and that the rate of convergence is virtually h-independent, especially for $\nu \leq 0.01$. Here as in all other unsteady (or quasi-steady) problems that we have tested, the rate of convergence is not overly sensitive to the choice of α, especially for small ν. A good choice is $\alpha \approx 0.5$ for the two coarser grids, and $\alpha \approx 0.25$ for the finer grids. Again, the optimal value of α is a few times larger for $\nu = 0.0001$ than for $\nu = 0.1$. The behavior of the iteration count as a function of α is displayed in Table 5.3 for the case $\nu = 0.001$ on the 256×256 grid.

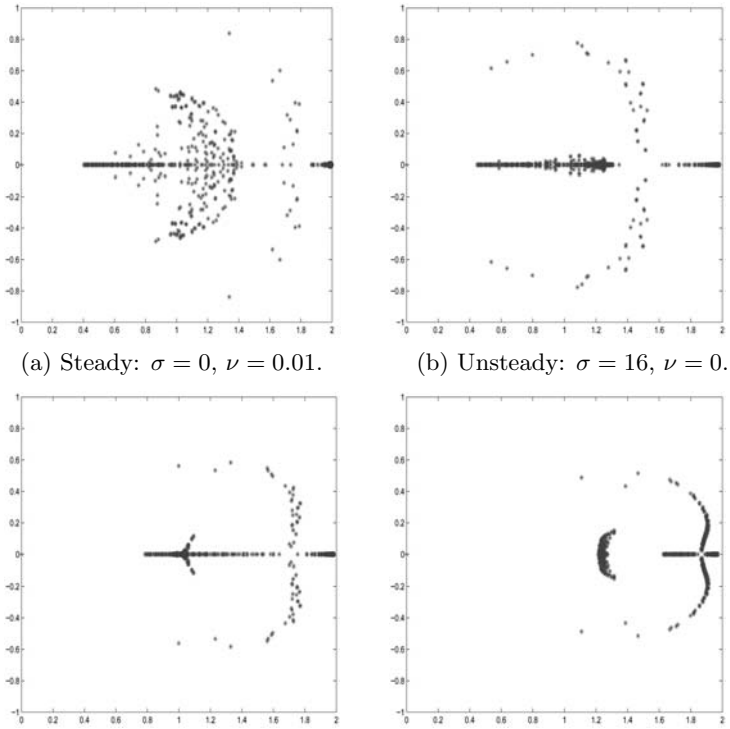
Table 5.4 shows iteration counts for a set of 3D generalized Oseen problems on a $40 \times 40 \times 40$ grid, for several values of the parameter σ. No attempt was made to find the optimal value of α for each problem; instead, the same value $\alpha = 0.1$ was used in all cases. The discrete problem has $n = 187,200$ and $m = 64,000$, for a total of $N = 251,200$ unknowns. It can be seen that the iteration counts are remarkably stable. When the optimal value of α is used, the rate of convergence improves for increasing σ and decreasing ν.

Virtually optimal (i.e., grid-independent) rates of convergence are obtained when σ is not held constant but is taken to be inversely proportional to h, as in the case of genuinely unsteady flow problems; see [25]. To show this, we use a set of 3D, unsteady Oseen problems with $\nu = 0.001$ and $\sigma = h^{-1}$. The results for HSS preconditioning (with the fixed value $\alpha = 0.5$) are reported in Table 5.5. The total number

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

TABLE 5.5
Results for 3D unsteady Oseen problem with $\sigma = h^{-1}$, $\nu = 0.001$ (exact solves, $\alpha = 0.5$).

Grid	n	m	Iterations	CPU time
$16 \times 16 \times 16$	11,520	4,096	12	0.80
$32 \times 32 \times 32$	95,232	32,768	13	9.88
$64 \times 64 \times 64$	774,144	262,144	15	155.
$80 \times 80 \times 80$	1,516,800	512,000	16	563.



(a) Steady: $\sigma = 0$, $\nu = 0.01$. (b) Unsteady: $\sigma = 16$, $\nu = 0.1$.

(c) Unsteady: $\sigma = 16$, $\nu = 0.01$. (d) Unsteady: $\sigma = 16$, $\nu = 0.001$.

FIG. 5.1. *Eigenvalues of discrete generalized Oseen problem preconditioned with HSS.*

of unknowns ranges from 15,616 for the coarsest grid to 2,028,800 for the finest one.

These experiments show the good behavior of HSS preconditioning on Oseen-type problems, in particular for unsteady or quasi-steady problems. The performance is especially good for small values of the viscosity ν. This is in contrast to most other solvers, which tend to deteriorate for small values of ν.

Figures 5.1(a)–(d) display the eigenvalues of the HSS-preconditioned discrete Oseen operator on a 16×16 grid for different values of σ and ν. As predicted by the theory in [4, 19], the eigenvalues lie in the interior of the disk of center $(1, 0)$ and radius 1, except for the zero eigenvalue corresponding to the one-dimensional null space of the discrete gradient B^T. Since this lone zero eigenvalue has no effect on the convergence of preconditioned GMRES, we did not remove it by imposing (2.5) or other auxiliary conditions; see the discussion in [12, Chapter 2.3]. Note that the clustering of the spectrum away from the zero eigenvalue becomes more pronounced as ν gets smaller.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 5.6
Results for 2D steady Stokes problem with $\sigma = 0$, $\nu = 1$ (exact solves).

Grid	Iterations	CPU time
16×16	30	0.13
32×32	39	0.96
64×64	49	8.33
128×128	62	79.7
256×256	81	813.

TABLE 5.7
Results for 2D unsteady Stokes problem with $\sigma = 40$, $\nu = 0.001$ (exact solves).

Grid	Iterations	CPU time
16×16	8	0.06
32×32	9	0.32
64×64	11	2.63
128×128	15	26.6
256×256	20	292.

TABLE 5.8
Results for 3D steady Stokes problem with $\sigma = 0$, $\nu = 1$ (exact solves).

Grid	Iterations	CPU time
$16 \times 16 \times 16$	28	1.15
$32 \times 32 \times 32$	35	16.3
$64 \times 64 \times 64$	45	246.
$80 \times 80 \times 80$	51	772.

5.2. Stokes flow. Here we consider the generalized Stokes problem on the unit square (or unit cube in three dimensions). The boundary conditions, discretization scheme, and GMRES stopping criterion are the same as in the previous section. HSS preconditioning was carried out with "exact" solves using MATLAB's sparse Cholesky decomposition with an approximate minimum degree reordering; inexact solves are considered in section 5.3.

For the first two sets of experiments, we fix σ and ν and investigate the behavior of the preconditioned iteration for different values of h for a 2D problem. Iteration counts and CPU times (in seconds) are reported in Table 5.6 for $\sigma = 0$ and $\nu = 1$ and in Table 5.7 for $\sigma = 40$ and $\nu = 0.001$. For the steady ($\sigma = 0$) problem the best value of α depends on h; our rule of thumb is to take $\alpha \approx 5h$. The constant value $\alpha = 0.25$ was used for the unsteady problem. As for the Oseen-type case, our experience is that for unsteady problems the best value of α does not depend on h; hence, it can be determined for a coarse grid and used on finer grids with good results. Overall, the rate of convergence is not overly sensitive to the choice of α.

These results show that the rate of convergence with HSS preconditioning suffers some deterioration as $h \to 0$. On the other hand, while not h-independent, the rate convergence is quite good, especially for $\sigma > 0$. (A Fourier analysis of HSS preconditioning for the Stokes case will be presented elsewhere.)

Next we consider a steady 3D Stokes problem. Here we found that a good rule of thumb for the choice of the HSS parameter is $\alpha \approx 2h$. The results are shown in Table 5.8. Again, the convergence rate is reasonable, but not independent of the mesh size.

As in the 2D case, the performance of HSS preconditioning tends to improve as ν gets smaller and as σ gets larger. Table 5.9 shows iteration counts for a set of 3D generalized Stokes problems on a $40 \times 40 \times 40$ grid, for several values of the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 5.9
Iteration count for 3D generalized Stokes problem (exact solves).

σ	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\nu = 10^{-6}$
1	45	27	16	13
10	32	19	15	12
20	30	18	14	11
50	28	15	13	11
100	25	14	12	10

TABLE 5.10
Results for 3D unsteady Stokes problem with $\sigma = h^{-1}$, $\nu = 0.001$ (exact solves).

Grid	Iterations	CPU time
$16 \times 16 \times 16$	12	0.78
$32 \times 32 \times 32$	12	8.78
$64 \times 64 \times 64$	12	126.
$80 \times 80 \times 80$	14	455.

parameter σ. The fixed value $\alpha = 0.5$ was used in all cases. The discrete problem has $n = 187,200$ and $m = 64,000$, for a total of $N = 251,200$ unknowns. It can be seen that the iteration counts improve for increasing σ and decreasing ν.

Optimal (i.e., grid-independent) rates of convergence are obtained when σ is not held constant but is taken to be inversely proportional to h (see, e.g., [25]). To show this, we use a set of 3D, unsteady Stokes problems with $\nu = 0.001$ and $\sigma = h^{-1}$. The results for HSS preconditioning (with the fixed value $\alpha = 0.5$) are reported in Table 5.10.

These experiments show the excellent behavior of HSS preconditioning for nonstationary Stokes problems, especially for small values of the viscosity ν and for a wide range of values of σ. On the other hand, the behavior of the preconditioner is suboptimal for steady Stokes problems (as can be verified by Fourier analysis).

It is interesting to note that for the Stokes problem the HSS-preconditioned matrix is diagonalizable and all its eigenvalues are real (and, of course, positive). Hence, there is a nonstandard inner product in which the preconditioned matrix is symmetric positive definite; see [6].

5.3. The effect of inexact solves. In all the results presented so far we have always used "exact" solves (in the form of sparse direct factorizations) for the linear systems that arise in the application of HSS preconditioning. As is well known, the storage and arithmetic complexity of direct solvers scales superlinearly with problem size when they are applied to linear systems arising from the discretization of elliptic PDEs; even with the best fill-reducing orderings currently available, for 3D problems their cost becomes quickly prohibitive as h is decreased. The poor scaling of sparse direct solvers is largely responsible for the superlinear scaling observed in the solution timings reported in the previous two sections. Other factors include (possibly) the nonconstant number of iterations as well as memory traffic issues.

Faster solution and better scalability may be achieved if the exact solves are replaced by inexact ones using inner iterations. We recall that for the (generalized) Stokes problem, all the inner solves correspond to shifted Poisson-type equations, which can be (approximately) solved efficiently by standard multigrid or by preconditioned conjugate gradients. For the Oseen problem, two (or three in three dimensions) of the systems are shifted Poisson-type equations, and one corresponds to a nonsymmetric but strongly elliptic problem. The latter system can be solved by

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

TABLE 5.11
Inexact versus exact solves, 2D unsteady Stokes problem with $\sigma = h^{-1}$, $\nu = 0.001$.

Grid	Inexact set-up	Solve	Total	Its	Exact set-up	Solve	Total	Its
32×32	0.04	0.18	0.22	13	0.05	0.10	0.15	13
64×64	0.19	0.94	1.13	16	0.20	0.51	0.71	14
128×128	1.24	5.60	6.84	18	1.07	2.74	3.81	14
256×256	10.4	29.0	39.4	19	8.10	13.5	21.6	15
512×512	106.	192.	298.	23	87.8	74.7	162.	16

TABLE 5.12
Inexact versus exact solves, 2D unsteady Oseen problem with $\sigma = h^{-1}$, $\nu = 0.001$.

Grid	Inexact set-up	Solve	Total	Its	Exact set-up	Solve	Total	Its
32×32	0.05	0.33	0.38	13	0.05	0.11	0.16	13
64×64	0.25	1.72	1.97	13	0.34	0.52	0.86	13
128×128	2.25	8.63	10.8	16	2.96	3.18	6.14	16
256×256	24.1	46.2	70.3	20	32.3	20.3	52.6	20
512×512	206.	343.	549.	22	503.	129.	632.	22

preconditioned GMRES or by a suitable multigrid method.

In Tables 5.11–5.12 we compare results for HSS-preconditioned FGMRES with inexact solves with those for HSS-preconditioned GMRES with exact solves. The test problems are unsteady 2D Stokes and Oseen problems with $\nu = 0.001$. We used drop tolerance-based incomplete Cholesky factorization preconditioning, available in MATLAB (function `cholinc`), for the two linear systems in (4.5). For (4.8) we used incomplete Cholesky in the Stokes case (where the Schur complement is symmetric and positive definite; see (4.9)) and incomplete LU (function `luinc`) in the Oseen case.

The drop tolerance τ in the incomplete factorizations was chosen so as to ensure that just 1–2 inner PCG iterations sufficed to reduce the inner residuals by at least a factor of 10. This low accuracy is already enough to ensure that the total number of (outer) FGMRES iterations is nearly the same as that of GMRES with exact solves. The use of incomplete factorizations leads to a reduction in memory requirements for the triangular factors, but this is partially offset by the memory overhead induced by the use of a flexible outer iteration. For the two shifted Laplacians in (4.5), which are very well conditioned, we used $\tau = 10^{-2}$. For the Schur complement system (4.9), in the Stokes case we used $\tau = 10^{-3}$ for all grids, except for the 512×512 case, where we used $\tau = 10^{-4}$. The 512×512 problem has a total of $N = 785,408$ unknowns. For the Schur complement in the Oseen problem we set the drop tolerance to 10^{-4} on all grids except for the 512×512 case, where we used $\tau = 10^{-5}$. Larger drop tolerances can be used at the expense of more inner iterations, and this tends to increase solution times.

The first observation to be made is that the use of inexact solves does not lead to a strong increase in the number of outer (FGMRES) iterations, even when a very loose tolerance ($tol = 10^{-1}$) is used for the inner iterations. This is especially true for the Oseen problem, for which the number of outer iterations is exactly the same as with exact solves.

Second, it appears from the results in Tables 5.11–5.12 that replacing the exact inner solves with inexact ones does not typically result in faster solution and may actually lead to higher computing times. Indeed, even for the set-up phase the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

advantages of using incomplete factorizations instead of complete ones are small or nonexistent. Inexact solves are clearly counterproductive for the Stokes problem. The situation is somewhat better for the Oseen problem; however, the only case where inexact solves resulted in faster time to solution is in the case of the 512×512 grid. Note that in this case, the savings are entirely a result of reduced set-up costs. It should be kept in mind that the incomplete factorizations and inner iterations used are not very efficient. For the complete factorizations, the MATLAB code makes use of highly optimized sparse direct solvers. Also, the matrices are first reordered with an approximate minimum degree heuristic, which results in relatively low fill-in. In contrast, the implementations of the incomplete Cholesky and incomplete LU factorization functions in MATLAB are not very efficient. Furthermore, minimum degree reordering of the equations tends to have an adverse effect on the rate of convergence of the inner iterations; hence, no reordering was used when computing the incomplete factorizations. Because of this, the incomplete factors are not much cheaper to compute than the complete factors, and the additional costs induced by the (few) inner Krylov subspace iterations needed to satisfy the convergence criteria for the inexact solves lead in almost all cases to increased overall solution costs compared to the case of exact solves.

Nevertheless, the fact that HSS preconditioning does not significantly deteriorate in the presence of inexact solves is encouraging. We expect that with a more optimized implementation, or with the use of more efficient solvers for elliptic problems (i.e., multigrid schemes) it will be possible to achieve faster solutions and better scalability than with exact solves, at least for sufficiently large problems.

Finally, we note that similar conclusions hold for the case of steady problems and problems in three dimensions.

5.4. Comparison with other methods. Extensive experimental comparisons with other solvers have been carried out in [19]. We give here a brief summary of our findings.

To our knowledge, the only other solver explicitly developed for the rotation form of the Navier–Stokes equations is Olshanskii's block triangular preconditioner; see, e.g., [21] or the brief discussion in section 4.2. This preconditioner is very effective for the solution of unsteady Oseen-type problems. For steady problems, the exact variant of the preconditioner appears to be both h- and ν-independent; when inexact inner solves are used, however, a fairly high number of inner iterations may be needed to keep the number of outer iterations from growing as ν becomes smaller.

A possible drawback of this approach is the need to solve linear systems with coefficient matrix A at each iteration. In this system, all components of the velocities are coupled. Standard sparse solvers (direct or iterative) do not perform well on such systems; instead, specialized multigrid solvers have to be used; see [24]. The work involved in the inversion of the (2,2) block is comparable to that required by the Schur complement solve in HSS preconditioning; note, however, that in the case of HSS the eigenvalues of the Schur complement are shifted away from zero (see (4.8)). The diagonal shift in (4.8) helps making the problem arising in HSS preconditioning better conditioned.

Summarizing, for many problems Olshanskii's preconditioner is about as effective as or better than HSS preconditioning on Oseen-type problems in terms of iteration counts. We do not have an efficient implementation of Olshanskii's preconditioner in MATLAB, and therefore we do not have a way to compare timings. The main advantage of HSS preconditioning, in our opinion, is the greater ease of implemen-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

tation. Furthermore, the linear systems that needs to be solved with HSS have a more convenient structure and better conditioning than the systems to be solved with Olshanskii's preconditioner, so the cost per iteration is generally lower.

Numerical experiments were also performed comparing HSS with Uzawa-type and SIMPLE preconditioning; see [19]. These techniques exhibit significant deterioration for small ν and were found to be generally inferior to both HSS and Olshanskii's preconditioner.

Concerning the Stokes problem, both steady and generalized, there are many optimal solvers available. In particular, the steady case is best handled by block diagonal preconditioners of the form

$$\mathcal{P} = \begin{bmatrix} \hat{A} & O \\ O & \hat{S} \end{bmatrix}, \tag{5.1}$$

where $\hat{A}$ is spectrally equivalent to A and $\hat{S}$ is spectrally equivalent to $S = BA^{-1}B^T$; see [32]. When using LBB-stable discretizations, finding spectrally equivalent approximations for the Schur complement S is easy: the pressure mass matrix (or a diagonal approximation of it) will do; for MAC, it is enough to take $S = I_m$. We performed some experiments with exact solves for the (1,1) block (i.e., $\hat{A} = A$) and we verified the expected h-independent convergence. The number of iterations for the finer grids settled around 20 in the 2D case. Since this preconditioner is cheaper than HSS, requiring only two Poisson-type solves rather than three at each step, we conclude that HSS cannot be recommended for the *steady* Stokes problem, since its rate of convergence is not h-independent.

For generalized Stokes problems (case $\sigma > 0$), on the other hand, HSS offers an attractive alternative to existing preconditioners (see [8, 17, 25]). The rates of convergence obtained with HSS and its cost per iteration appear to be competitive with those that have been reported in the literature; furthermore, implementation of the HSS preconditioner is straightforward.

5.5. Behavior of the Picard iteration. An important question that has not been addressed so far in this paper is that of the rate of convergence of the nonlinear Picard iteration for the rotation form of the Navier–Stokes equations. For the standard (convection) form of the steady Navier–Stokes equations, conditions for the (global) convergence of the Picard iteration have been given in [16]. We are not aware of any such analysis for the rotation form of the equations.

Limited numerical experience reported by Olshanskii in [23] suggests that the rate of convergence of Picard's iteration for the rotation form tends to deteriorate for decreasing values of the viscosity ν. Although some deterioration is also observed for the convective form of the equations, the phenomenon appears to be more pronounced for the rotation form. This is probably the price one pays for the avoidance of convection in the linear solves.

Our own experience is in agreement with the results reported by Olshanskii. In particular, we found that Picard's iteration for the rotation form tends to converge fast for relatively large ν (say, between 2 and 6 iterations for $\nu > 0.05$ in a 2D confined flow problem); however, convergence slows down dramatically for smaller ν (for example, 68 iterations for $\nu = 0.03$), and the iteration fails to converge for smaller values of ν.

In contrast, Picard's iteration was found to converge when using the convective form of the equations, even for small values of the viscosity. The number of iterations for solving the same confined flow problem was found to be 4, 18, and 80 for $\nu = 0.1$,

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

TABLE 5.13
Picard iterations for rotation form with a regularization term $\xi h^2 I$.

Problem	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$
32×32, $\xi = 10$	3	4	6
error	6.21e-04	5.81e-04	5.79e-04
64×64, $\xi = 12$	3	4	6
error	1.49e-04	1.38e-05	1.36e-04

0.01, and 0.001, respectively. All these results are essentially independent of the mesh size, at least for fine enough grids. In these experiments, convergence of the Picard iteration is achieved when the 2-norm of the (nonlinear) residual is reduced below $tol = 10^{-5}$ times the initial residual (see (8.34) in [12]), starting from a zero initial guess. Exact solves were used at each Picard iteration.

Clearly, this is a serious drawback of the rotation form for steady problems. We found, however, that the convergence of Picard's iteration can be greatly improved by adding to the momentum equations (2.13) a term of the form $\xi \mathbf{u}$, where $\xi > 0$ is a moderate constant. Upon discretization, this amounts to the addition of a term of the form $\xi\, h^2 I$ to the (1,1) block of the system, and can be regarded as a form of *pseudo–time stepping*. Note that this modification changes the solution of the discrete problem; however, the extra term is small and tends to zero as $h \to 0$. The modification can also be interpreted as a kind of (velocity) stabilization. Although we do not have a rigorous justification for this form of regularization, we mention that the technique is similar to the one used in [13] for saddle point problems arising from the discretization of the curl-curl form of Maxwell's equations.

Nonlinear iteration counts for the modified problem on two grids with two different values of ξ are reported in Table 5.13. We also include the discrete L^2-norm of the difference between the velocity field obtained by Picard's iteration on the modified problem and the true (analytical) solution of our test problem. This error is very small in all cases. However, how to choose ξ remains an open question. Choosing too small a value of ξ leads to slow convergence for very small ν; for example, on the 32×32 grid, using $\xi = 7$ results in 3, 6, and 45 iterations for $\nu = 0.1$, 0.01, and 0.001, respectively. On the other hand, choosing ξ too large could in principle lead to a loss of accuracy in the computed solution. In our (admittedly somewhat limited) experience, however, somewhat overestimating ξ does not lead to a noticeable worsening of the error.

6. Conclusions. In this paper we have described a preconditioning technique for the large sparse linear systems arising from discretizations of the Navier–Stokes equations in rotation form. The basic idea is to decouple the symmetric and skew-symmetric components of the linearized Navier–Stokes system and to alternate between the two. The preconditioner contains a parameter which requires some tuning for optimal performance; we described some heuristics that can be used to find good estimates for the optimal parameter. However, in most cases the performance is not overly sensitive to the value of this parameter. The preconditioner can also be interpreted as an approximate block factorization of the (global) system matrix which becomes increasingly accurate in the vanishing viscosity limit $\nu \to 0$.

Implementation of the preconditioner is straightforward (more so than with other methods), and the numerical experiments show good performance on a wide range of cases, especially for unsteady problems and for low-viscosity problems. While the preconditioner cannot be recommended for steady Stokes problems, it performs very

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

well in the case of quasi-steady and unsteady problems. We remark that these appear to be precisely the kind of problems for which the rotation form is best suited; see [27].

Although our computational experience has been limited to uniform MAC discretizations and simple geometries, the preconditioner should be applicable to more complicated problems and discretizations, including unstructured grids.

Finally, although we have not discussed parallelization issues in this paper, the preconditioner can easily take advantage of existing elliptic solvers for parallel platforms, such as domain decomposition techniques.

Acknowledgments. The authors would like to thank Maxim Olshanskii and Eldad Haber for helpful suggestions.

REFERENCES

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[2] Z.-Z. Bai, G. H. Golub, and M. K. Ng, *Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 603–626.

[3] M. Benzi, M. J. Gander, and G. H. Golub, *Optimization of the Hermitian and skew-Hermitian splitting iteration for saddle-point problems*, BIT, 43 (2003), pp. 881–900.

[4] M. Benzi and G. H. Golub, *A preconditioner for generalized saddle point problems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 20–41.

[5] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[6] M. Benzi and V. Simoncini, *On the eigenvalues of a class of saddle point matrices*, Numer. Math., 103 (2006), pp. 173–196.

[7] M. Benzi and D. B. Szyld, *Existence and uniqueness of splittings for stationary iterative methods with applications to alternating methods*, Numer. Math., 76 (1997), pp. 309–321.

[8] J. Cahouet and J. P. Chabard, *Some fast 3-D finite element solvers for the generalized Stokes problem*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 869–895.

[9] W. Chang, F. Giraldo, and J. B. Perot, *Analysis of an exact fractional step method*, J. Comput. Phys., 180 (2002), pp. 183–199.

[10] H. C. Elman, *Preconditioning for the steady-state Navier–Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.

[11] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Performance and analysis of saddle point preconditioners for the discrete steady-state Navier–Stokes equations*, Numer. Math., 90 (2002), pp. 665–688.

[12] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Numer. Math. Sci. Comput., Oxford University Press, Oxford, UK, 2005.

[13] C. Greif and D. Schötzau, *Preconditioners for the discretized time-harmonic Maxwell equations in mixed form*, Numer. Linear Algebra Appl., 14 (2007), pp. 281–297.

[14] P. M. Gresho, *Incompressible fluid dynamics: Some fundamental formulation issues*, Ann. Rev. Fluid Mech., 23 (1991), pp. 413–453.

[15] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, Phys. Fluids, 8 (1965), pp. 2182–2189.

[16] O. A. Karakashian, *On a Galerkin–Lagrange multiplier method for the stationary Navier–Stokes equations*, SIAM J. Numer. Anal., 19 (1982), pp. 909–923.

[17] G. M. Kobelkov and M. A. Olshanskii, *Effective preconditioning of Uzawa type scheme for a generalized Stokes problem*, Numer. Math., 86 (2000), pp. 443–470.

[18] C. Li and C. Vuik, *Eigenvalue analysis of the SIMPLE preconditioning for incompressible flow*, Numer. Linear Algebra Appl., 11 (2004), pp. 511–523.

[19] J. Liu, *Preconditioned Krylov Subspace Methods for Incompressible Flow Problems*, Ph.D. thesis, Department of Mathematics and Computer Science, Emory University, Atlanta, 2006; available online at http://www.uwf.edu/jliu/research.htm.

[20] G. Lube and M. A. Olshanskii, *Stable finite element calculation of incompressible flows using the rotation form of convection*, IMA J. Numer. Anal., 22 (2002), pp. 437–461.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

[21] M. A. Olshanskii, *An iterative solver for the Oseen problem and numerical solution of incompressible Navier–Stokes equations*, Numer. Linear Algebra Appl., 6 (1999), pp. 353–378.
[22] M. A. Olshanskii, *A low order Galerkin finite element method for the Navier–Stokes equations of steady incompressible flow: A stabilization issue and iterative methods*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 5515–5536.
[23] M. A. Olshanskii, *Preconditioned iterations for the linearized Navier–Stokes system in rotation form*, in Proceedings of the Second MIT Conference on Computational Fluid and Solid Mechanics, K. J. Bathe, ed., Elsevier, New York, 2003, pp. 1074–1077.
[24] M. A. Olshanskii and A. Reusken, *Navier–Stokes equations in rotation form: A robust multigrid solver for the velocity problem*, SIAM J. Sci. Comput., 23 (2002), pp. 1683–1706.
[25] J. Peters, V. Reichelt, and A. Reusken, *Fast iterative solvers for discrete Stokes equations*, SIAM J. Sci. Comput., 27 (2005), pp. 646–666.
[26] A. Quarteroni, F. Saleri, and A. Veneziani, *Factorization methods for the numerical approximation of Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 188 (2000), pp. 505–526.
[27] L. G. Rebholz, *An energy- and helicity-conserving finite element scheme for the Navier–Stokes equations*, SIAM J. Numer. Anal., 45 (2007), pp. 1622–1638.
[28] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[30] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[31] F. Saleri and A. Veneziani, *Pressure correction algebraic splitting methods for the incompressible Navier–Stokes equations*, SIAM J. Numer. Anal., 43 (2005), pp. 174–194.
[32] D. J. Silvester and A. J. Wathen, *Fast iterative solution of stabilised Stokes systems. Part* II: *Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.
[33] V. Simoncini and M. Benzi, *Spectral properties of the Hermitian and skew-Hermitian splitting preconditioner for saddle point problems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 377–389.
[34] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monogr. Appl. Comput. Math. 13, Cambridge University Press, Cambridge, UK, 2003.
[35] A. Veneziani, *Block factorized preconditioners for high-order accurate in time approximation to the Navier–Stokes equations*, Numer. Methods Partial Differential Equations, 19 (2003), pp. 487–510.
[36] P. Wesseling, *Principles of Computational Fluid Dynamics*, Springer, Berlin, 2001.
[37] T. A. Zang, *On the rotation and skew-symmetric forms for incompressible flow simulations*, Appl. Numer. Math., 7 (1991), pp. 27–40.

Downloaded 06/18/20 to 109.31.5.235. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.