NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 0000; **00**:1–24
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla

Preconditioning a mass-conserving discontinuous Galerkin discretization of the Stokes Equations

James H. Adler[1], Thomas R. Benson[1*], and Scott P. MacLachlan[2]

[1] *Department of Mathematics, Tufts University, 503 Boston Ave., Medford, MA 02155 USA*
[2] *Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's, NL, Canada A1C 5S7*

SUMMARY

The incompressible Stokes equations are a widely-used model of viscous or tightly confined flow in which convection effects are negligible. In order to strongly enforce the conservation of mass at the element scale, special discretization techniques must be employed. In this paper, we consider a discontinuous Galerkin (DG) approximation in which the velocity field is $H(\mathrm{div}, \Omega)$-conforming and divergence-free, based on the BDM_1 finite-element space, with complementary space (P_0) for the pressure. Due to the saddle-point structure and the nature of the resulting variational formulation, the linear systems can be difficult to solve. Therefore, specialized preconditioning strategies are required in order to efficiently solve these systems. We compare the effectiveness of two families of preconditioners for saddle-point systems when applied to the resulting matrix problem. Specifically, we consider block-factorization techniques, in which the velocity block is preconditioned using geometric multigrid, as well as fully-coupled monolithic multigrid methods. We present parameter study data and a serial timing comparison, and we show that a monolithic multigrid preconditioner using Braess-Sarazin style relaxation provides the fastest time to solution for the test problem considered. Copyright © 0000 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: block-factorization preconditioners, monolithic multigrid, discontinuous Galerkin, Stokes equations

1. INTRODUCTION

The incompressible Stokes equations in domain Ω can be written

$$-\nabla \cdot (2\nu\boldsymbol{\varepsilon}(\mathbf{u})) + \nabla p = \mathbf{f} \qquad \text{in } \Omega \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega, \tag{2}$$

together with appropriate boundary conditions. Here, $\mathbf{u}$ is the fluid velocity, p is the pressure, $\mathbf{f}$ is an applied external force, ν is the fluid viscosity, and $\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$ is the strain-rate tensor. These equations represent a model of viscous or tightly confined flow in which convection effects may be ignored [1]. The conservation of mass equation, (2), is important in a variety of applications, especially when coupled to more complex physics [2], where numerical instabilities can arise when this is not satisfied exactly.

In order to address the conservation issue, we consider a stable mixed finite-element formulation using discontinuous Galerkin (DG). The DG method that we consider here was

*Correspondence to: Department of Mathematics, Tufts University, 503 Boston Ave., Medford, MA 02155 USA

introduced for the Stokes problem in [3] and for the Navier-Stokes problem in [4], both of which show that the approximate velocity field is $\mathbf{H}(\text{div}, \Omega)$-conforming and divergence-free almost everywhere, which is not generally true in the case of $\mathbf{H}^1(\Omega)$-conforming discretizations (e.g. Taylor-Hood). This is accomplished by the use of the first-order finite-element spaces of Brezzi, Douglas, and Marini (BDM_1) for velocity [21], such that $\nabla \cdot \mathbf{u}$ is piecewise constant on a triangulation. Using the P_0 space for pressure results in a discretization for which weak satisfaction of $\nabla \cdot \mathbf{u} = 0$ is equivalent to strong satisfaction of the incompressibility condition.

This discretization was also developed in [2] for an incompressible magnetohydrodynamics (MHD) system. In [5], the authors extend the results of [3,4] to the case of the Stokes equations with the particular boundary condition that we consider (described in Section 2), showing the $H(\text{div}, \Omega)$-conforming DG method to be stable and optimally convergent, as well as to provide a divergence-free velocity approximation.

After discretization, the resulting linear system is of saddle-point type. Such systems arise not only in incompressible fluid dynamics, but also in finance and economics, solid mechanics, and many other fields. An extensive review of saddle-point problems can be found in [6]. As a result of the relative ubiquity of systems of this type, efficient solvers and preconditioners are necessary and are abundantly studied [7–13]. A common choice of preconditioner is the class of block-factorization approaches [1,14]. In the context of the Stokes equations, these methods typically require a preconditioner for the velocity block and a suitable approximation to the pressure Schur complement. The velocity and pressure are effectively decoupled. Alternatively, monolithic multigrid methods for these problems have also been proposed [15–17]. These methods operate on both the velocity and the pressure simulataneously.

In contrast to either of the aforementioned classes of preconditioners for saddle-point problems, solver strategies specifically tailored to the DG-method used here have also been presented. In [5], the authors consider an auxiliary space preconditioner [18] that requires projections to the curl space, which, in turn, require solving auxiliary systems corresponding to scalar Laplacians. Numerical results in [5] use a direct solver as a preconditioner for the stiffness matrix corresponding to the velocity block, noting that research is needed to develop effective approximations to the velocity block; such approximations are developed here in the context of block-factorization preconditioners. In [19], a multigrid approach for the velocity block is presented, using a vertex-based block-SOR type relaxation scheme similar to the element-based block-SOR method that we discuss in Section 3. This method is proposed within the context of an augmented Uzawa method. While these approaches show that efficient solution of these linear systems is possible, the question of the effectiveness of classical preconditioning strategies has not yet been considered.

In this paper, we examine the two families of common preconditioners described above, namely block factorization methods and monolithic multigrid methods specifically designed for the BDM_1-P_0 discretization. First, we compare with standard block-diagonal and block-triangular preconditioners. Here, the diagonal block corresponding to the discretization of velocity is approximated using a multigrid cycle and the diagonal block corresponding to the Schur complement is approximated by the pressure-space mass matrix. Within the multigrid method, we consider both point and block relaxation techniques. Finally, we consider monolithic multigrid methods that treat the coupled system at once, using either Braess-Sarazin relaxation [15] or Vanka relaxation [16]. Both of these methods will require some modification from their original forms as a result of the discretization being used. After describing the details of the preconditioners, we show parameter study results and timing comparisons to find the best scalable method for this discretization.

The remainder of this paper is organized as follows. In Section 2, the continuous and discrete problems are developed in greater detail, and the BDM_1-P_0 finite-element method is described. Then the preconditioners are explained in depth, with the block-factorization preconditioning approaches in Section 3 and the monolithic multigrid preconditioners in Section 4. Finally, numerical results are shown in Section 5 in which we examine parameter choices for each

DOI: 10.1002/nla

preconditioner and comment upon the performance of the resulting solvers for a test problem on both a uniform mesh and an unstructured mesh.

2. THE PROBLEM AND DISCRETIZATION

We consider the Stokes Equations, as given in Equations (1) and (2), on a domain $\Omega \subset \mathbb{R}^2$. Here, we take the known fluid viscosity, ν, to be constant throughout the domain.

Remark. With ν constant, by enforcing the divergence-free constraint, we can write the divergence of the strain-rate tensor as a scaled vector-Laplacian:

$$\nabla \cdot (2\nu\boldsymbol{\varepsilon}(\mathbf{u})) = 2\nu\nabla^2\mathbf{u}.$$

However, we do not make this simplification when implementing the bilinear form for the numerical studies; we discretize the full strain-rate tensor.

We consider the case of enclosed flow and, therefore, impose upon the boundary, $\partial\Omega$, a no-flux condition:

$$\mathbf{u} \cdot \mathbf{n} = 0 \qquad \text{on } \partial\Omega, \tag{3}$$

where $\mathbf{n}$ is an outward-pointing unit normal vector. We also impose a condition on the tangential component of the normal stresses

$$((2\nu\boldsymbol{\varepsilon}(\mathbf{u}) - p\mathbf{I})\mathbf{n}) \cdot \mathbf{t} = 0 \qquad \text{on } \partial\Omega, \tag{4}$$

where $\mathbf{t}$ denotes a vector pointing in the tangential direction on the boundary.

In the usual way, we define the spaces

$$\mathbf{H}_0^1(\Omega) = \{\mathbf{v} \in \mathbf{H}^1(\Omega) \colon \mathbf{v} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega\}$$

and $L^2(\Omega)/\mathbb{R}$ as the the quotient space of equivalence classes of elements of $L^2(\Omega)$ that differ by a constant. This gives the following variational formulation.

Weak Form (Continuous)**.** *Find* $(\mathbf{u}, p) \in \mathbf{H}_0^1 \times L^2(\Omega)/\mathbb{R}$ *that satisfies*

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) &= (f, \mathbf{v}) \quad \forall \mathbf{v} \in \mathbf{H}_0^1(\Omega) \\ b(\mathbf{u}, q) &= 0 \qquad \forall q \in L^2(\Omega)/\mathbb{R}. \end{aligned} \tag{5}$$

The bilinear forms are

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) &:= 2\nu \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) \colon \boldsymbol{\varepsilon}(\mathbf{v}) \, \mathrm{d}x \quad \forall \mathbf{u}, \mathbf{v} \in \mathbf{H}_0^1(\Omega), \\ b(\mathbf{v}, q) &:= -\int_\Omega q \nabla \cdot \mathbf{v} \, \mathrm{d}x \qquad \forall \mathbf{v} \in \mathbf{H}_0^1(\Omega), \ p \in L^2(\Omega)/\mathbb{R}. \end{aligned} \tag{6}$$

The linear form is

$$(\mathbf{f}, \mathbf{v}) := \int_\Omega \mathbf{f} \cdot \mathbf{v} \, \mathrm{d}x \qquad \forall \mathbf{v} \in \mathbf{H}_0^1(\Omega). \tag{7}$$

2.1. Formulation of the discontinuous Galerkin discretization

Let $\mathcal{T}_h$ be a shape-regular partition of Ω into triangles, T. Then, partition the set of all edges $\mathcal{E}_h = \mathcal{E}_h^0 + \mathcal{E}_h^\partial$, where $\mathcal{E}_h^0$ is the set of interior edges and $\mathcal{E}_h^\partial$ is the set of boundary edges. Also define

$$H^1(\mathcal{T}_h) = \{\phi \in L^2(\Omega) \colon \phi|_T \in H^1(T), \forall T \in \mathcal{T}_h\},$$

Prepared using nlaauth.cls
DOI: 10.1002/nla

and similarly for vectors (denoted $\mathbf{H}^1(\mathcal{T}_h)$) and tensors (denoted $\mathcal{H}^1(\mathcal{T}_h)$). The other spaces that we require are:

$$\mathbf{H}(\mathrm{div};\Omega) := \{\mathbf{v} \in \mathbf{L}^2(\Omega) \colon \nabla \cdot \mathbf{v} \in L^2(\Omega)\}$$
$$\mathbf{H}_0(\mathrm{div};\Omega) := \{\mathbf{v} \in \mathbf{H}(\mathrm{div};\Omega) \colon \mathbf{v} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega\}.$$

We note that $\mathbf{H}(\mathrm{div};\Omega)$ is a Hilbert space with norm

$$\|\mathbf{v}\|^2_{\mathbf{H}(\mathrm{div};\Omega)} := \|\mathbf{v}\|^2_{0,\Omega} + \|\nabla \cdot \mathbf{v}\|^2_{0,\Omega} \qquad \forall \mathbf{v} \in \mathbf{H}(\mathrm{div};\Omega),$$

where $\|\mathbf{v}\|_{0,\Omega}$ denotes the standard L^2-norm on Ω.

Consider $e \in \mathcal{E}_h^0$, shared by two elements T^1 and T^2; the value of a function f, be it a scalar, a vector, or a tensor function, on element T^1 is denoted f^1 and on element T^2 is denoted f^2. Furthermore, the unit normal to edge e pointing outward from T^1 is denoted $\mathbf{n}^1$ and from T^2 is denoted $\mathbf{n}^2$. Also, let $\mathbf{v}_t := (\mathbf{v} \cdot \mathbf{t})\mathbf{t}$ be the tangential component of a vector $\mathbf{v}$ on an edge. Now we define the average operators (see, e.g., [20]) for scalar functions $\phi \in H^1(\mathcal{T}_h)$, vector fields $\mathbf{v} \in \mathbf{H}^1(\mathcal{T}_h)$, and tensor fields $\boldsymbol{\tau} \in \mathcal{H}(\mathcal{T}_h)$ to be

$$\{\phi\} = \frac{1}{2}(\phi^1 + \phi^2) \qquad \{\mathbf{v}\} = \frac{1}{2}(\mathbf{v}^1 + \mathbf{v}^2) \qquad \{\boldsymbol{\tau}\} = \frac{1}{2}(\boldsymbol{\tau}^1 + \boldsymbol{\tau}^2)$$

on interior edges. On a boundary edge, we take $\{\phi\}$, $\{\mathbf{v}\}$, and $\{\boldsymbol{\tau}\}$ to be the trace of ϕ, $\mathbf{v}$, and $\boldsymbol{\tau}$, respectively.

For $\phi \in H^1(\mathcal{T})$, the jump operator is defined to be

$$[\![\phi]\!] = \phi^1\mathbf{n}^1 + \phi^2\mathbf{n}^2 \text{ on } e \in \mathcal{E}_h^0 \text{ and } [\![\phi]\!] = \phi\mathbf{n} \text{ on } e \in \mathcal{E}_h^\partial.$$

For $\mathbf{v} \in \mathbf{H}^1(\mathcal{T}_h)$, the jump operator is defined to be

$$[\![\mathbf{v}]\!] = \mathbf{v}^1 \odot \mathbf{n}^1 + \mathbf{v}^2 \odot \mathbf{n}^2 \text{ on } e \in \mathcal{E}_h^0 \text{ and } [\![\mathbf{v}]\!] = \mathbf{v} \odot \mathbf{n} \text{ on } e \in \mathcal{E}_h^\partial,$$

where $\mathbf{v} \odot \mathbf{n} = (\mathbf{v}\mathbf{n}^t + \mathbf{n}\mathbf{v}^t)/2$.

Now we introduce the two finite-element spaces $\mathbf{V}_h$ and Q_h. To discretize the weak form, we consider the first-order $\mathbf{H}(\mathrm{div},\Omega)$-conforming finite element spaces of Brezzi, Douglas, and Marini (BDM_1) [21]. In particular, we take

$$\mathbf{V}_h := \{\mathbf{v} \in \mathbf{H}_0(\mathrm{div};\Omega) \colon \mathbf{v}|_T \in \mathrm{BDM}_1(T)\ \forall T \in \mathcal{T}_h\}$$
$$Q_h := \{q \in L^2(\Omega)/\mathbb{R} \colon q|_T \in P_0(T)\ \forall T \in \mathcal{T}_h\}.$$

The degrees of freedom for $\mathrm{BDM}_1(T)$ in two dimensions are

$$\int_e \mathbf{u} \cdot \mathbf{n}_e q \, \mathrm{d}s \qquad \forall e \in \partial T, \forall q \in \mathbb{P}^1(e),$$

where $\mathbf{n}_e$ is the unit normal vector to edge e. As a result, $\mathbf{u} \in \mathbf{V}_h$ is a piecewise-linear vector field on the triangulation, with continous normal components across edges, but possibly discontinous tangential components. Note that in this case, we have two degrees of freedom per edge, or six degrees of freedom per triangle.

With $\mathbf{V}_h$ and Q_h defined, we write the discrete weak form of the problem:

Weak Form (Discrete)**.** *Find* $(\mathbf{u}_h, p_h)$ *in* $\mathbf{V}_h \times Q_h$ *such that:*

$$\begin{aligned} a_h(\mathbf{u}_h, \mathbf{v}) + b(\mathbf{v}, p_h) &= (\mathbf{f}, \mathbf{v}) \quad \forall \mathbf{v} \in \mathbf{V}_h \\ b(\mathbf{u}_h, q) &= 0 \qquad \forall q \in Q_h, \end{aligned} \tag{8}$$

DOI: 10.1002/nla

where

$$a_h(\mathbf{u},\mathbf{v}) := 2\nu \sum_{T\in\mathcal{T}_h} \int_T \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\,\mathrm{d}x + 2\nu\alpha \sum_{e\in\mathcal{E}_h^0} h_e^{-1} \int_e [\![\mathbf{u}_t]\!] : [\![\mathbf{v}_t]\!]\,\mathrm{d}s$$
$$- \sum_{e\in\mathcal{E}_h^0} \int_e \left[(\{\boldsymbol{\varepsilon}(\mathbf{u})\} : [\![\mathbf{v}_t]\!]) + ([\![\mathbf{u}_t]\!] : \{\boldsymbol{\varepsilon}(\mathbf{v})\}) \right] \mathrm{d}s \qquad \forall \mathbf{u},\mathbf{v} \in \mathbf{V}_h$$

$$b(\mathbf{v},q) := -\int_\Omega q\nabla\cdot\mathbf{v})\,\mathrm{d}x \qquad \forall \mathbf{v}\in\mathbf{V}_h, q\in Q_h$$

Here, h_e is the length of edge e and α is a penalty parameter that is taken to be positive and large enough to ensure well-posedness.

The well-posedness of (8) is established in Theorem 4.5 of [5], which we restate here for the convenience of the reader. A proof of this theorem and a detailed discussion of the theory of this discretization can be found therein.

Theorem (Theorem 4.5 of [5])**.** *Let $(\mathbf{V}_h, Q_h)$ be as described above. Then, problem (8) has a unique solution $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times Q_h$ that verifies*

$$\nabla\cdot\mathbf{u}_h = 0 \text{ in } \Omega.$$

Moreover, there exists a positive constant C, independent of h, such that for every $\mathbf{v}_h \in \mathbf{V}_h$ with $\nabla\cdot\mathbf{v}_h = 0$ and for every $q_h \in Q_h$, the following estimate holds:

$$\|\mathbf{u}-\mathbf{u}_h\|_{DG} \le C\|\mathbf{u}-\mathbf{v}_h\|_{DG}$$
$$\|p-p_h\|_{0,\Omega} \le C(\|p-q_h\|_{0,\Omega} + \|\mathbf{u}-\mathbf{v}_h\|_{DG},$$

with $(\mathbf{u},p)$ the solution of (5).

2.2. Saddle-point system

After discretization, we have the following linear system:

$$\underbrace{\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix}}_{A} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{9}$$

Here, F represents the discretization of the bilinear form $a(\mathbf{u},\mathbf{v})$, or the discrete representation of the divergence of the strain rate tensor $\boldsymbol{\varepsilon}(\mathbf{u})$. Note that, because this discretization uses vector elements, the block F cannot be written as a block-diagonal matrix with scalar Laplacians along the diagonal, as it is in the case of the Taylor-Hood discretization. The matrix B is the discrete divergence operator, and B^T is its adjoint, the discrete gradient operator. A thorough review on properties of such linear systems and preconditioning strategies can be found in [6]. To solve this saddle-point problem, we consider a preconditioned Krylov subspace method approach. We consider two categories of preconditioners, block-factorization methods and monolithic multigrid methods.

In the category of block-factorization methods, we look at a block-diagonal preconditioner and a block-triangular preconditioner [1, 22]. Here, the diagonal block corresponding to the divergence of the strain-rate tensor, F, is approximated using a multigrid cycle and the diagonal block corresponding to the Schur complement is approximated by the pressure-space mass matrix, which is a diagonal matrix for this discretization. Because F is not a block-diagonal matrix with scalar Laplacians on the diagonal, we cannot rely on scalar multigrid approaches for this problem. Within the multigrid method, we consider both point and block relaxation techniques.

Prepared using nlaauth.cls
DOI: 10.1002/nla

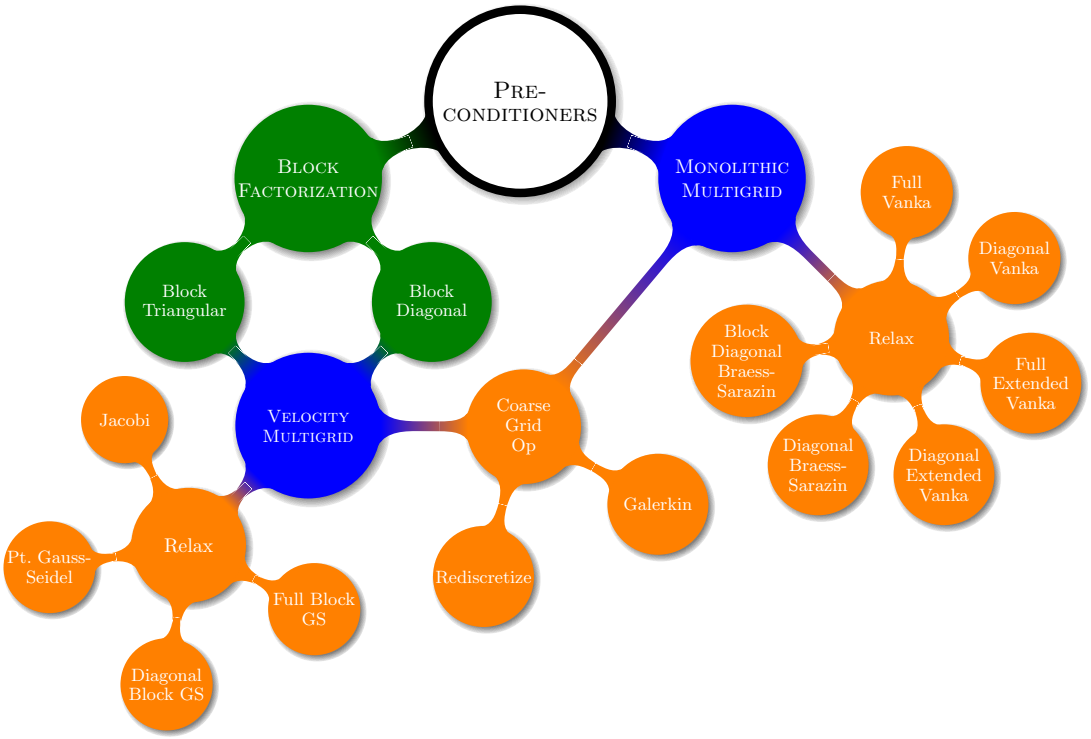


Figure 1. Preconditioning and multigrid options for saddle-point systems. In green, we highlight two block preconditioning strategies that require a multigrid solver for the velocity subsystem. In blue, we identify two approaches to applying multigrid, either for only the velocity subsystem within a block-structured preconditioner or as a monolithic method applied to the full coupled system. In orange, we highlight the options within the multigrid methods, for relaxation and coarse-grid operator construction.

In the category of monolithic multigrid methods [15, 16, 23], we investigate two families of relaxation schemes for this system. In monolithic multigrid approaches, the multigrid method treats the coupled system at once, though the individual components may decouple the system *within* the iteration. Here, we consider Braess-Sarazin relaxation [15], which is a global relaxation scheme, updating all degrees of freedom at once, and Vanka relaxation [16], which is a local relaxation scheme in which degrees of freedom are updated in patches.

We note that, as always, there is a vast parameter space of possible approaches to preconditioning these linear systems. Figure 1 presents these options in a structured way.

3. BLOCK PRECONDITIONERS

We first give details of the block preconditioning strategies considered; see [1, 7] for thorough discussions of these approaches. While many possiblities exist in this family, we consider two preconditioners, block diagonal,

$$M_d = \begin{bmatrix} \Lambda & \\ & Q \end{bmatrix}, \tag{10}$$

and block triangular,

$$M_t = \begin{bmatrix} \Lambda & B^T \\ & Q \end{bmatrix}. \tag{11}$$

DOI: 10.1002/nla

In both cases, we choose Λ^{-1} to be a multigrid cycle applied to the velocity block, F, and Q to be a pressure mass matrix, which is spectrally equivalent to the Schur complement, $-BF^{-1}B^T$, whenever the finite-element discretization is inf-sup stable [7]. In this case, the pressure mass matrix, corresponding to the P_0 discretization of pressure, is a diagonal matrix, with the volume of the corresponding element on the diagonal. We choose the positive mass matrix so that we can create an SPD preconditioner, M_d.

To treat Λ^{-1}, we use geometric interpolation operators based on the BDM_1 finite element interpolation operators, noting that the DG (edge) terms in the weak form have no influence upon the interpolation operator, $P_{\mathbf{u}}$. We also consider Galerkin coarse-grid operators, $F_c = P_{\mathbf{u}}^T F P_{\mathbf{u}}$. It is important to note here that this coarse-grid operator is not equivalent to rediscretization on the coarse grid, as would be the case for standard Galerkin discretizations, such as $P_2 - P_1$ (Taylor-Hood) elements. We will see in Section 5 that some of the methods will demonstrate a lack of convergence or scalability that can be ameliorated, to some degree, by using rediscretizion to form the coarse-grid operators.

The remaining component of the multigrid method is the choice of relaxation procedure. We consider two options, point and block schemes. As a first option, we focus on pointwise Gauss-Seidel/SOR methods, although many other options could also be considered. In order to maintain symmetry of the preconditioner, we utilize symmetric sweeping strategies, using either a symmetric sweep for both pre- and post-relaxation or a forward sweep for pre-relaxation and a backward sweep for post-relaxation. In this case, we have the standard SOR relaxation parameter, denoted ω_{SOR}.

The other relaxation scheme considered is an overlapping block Gauss-Seidel method. For a decomposition of the velocity degrees of freedom into overlapping blocks, $\{S_\ell\}$, such that each degree of freedom appears in at least one block, we update the components of the solution vector u corresponding to block S_ℓ as

$$u_\ell = u_\ell + \omega_{\mathrm{BGS}} \hat{F}_{\ell\ell}^{-1} (f - Fu)_\ell.$$

Here, the subscript ℓ on a vector indicates the restriction of the vector to those components present in the block ℓ. The matrix denoted $\hat{F}_{\ell\ell}$ is the local submatrix used in the update of block ℓ; its precise form is discussed below. Here, for the simplicity of implementation, the order in which the blocks are updated is dictated by the order of the elements in the finite-element software, and we only utilize forward sweeps for both pre- and post-relaxation. However, other orderings or coloring schemes are certainly possible and may be desirable from the perspective of parallel computation.

In this case, we choose for each block, S_ℓ, all of the degrees of freedom on element ℓ, as illustrated in Figure 2. The matrix $\hat{F}_{\ell\ell}$ is closely related to the submatrix denoted $F_{\ell\ell}$ that is obtained by restricting F to only those rows and columns corresponding to the degrees of freedom in S_ℓ. We consider two options: the *full* submatrix $\hat{F}_{\ell\ell}^{\mathrm{full}} = F_{\ell\ell}$ and the *diagonal* of the submatrix $\hat{F}_{\ell\ell} = \mathrm{diag}(F_{\ell\ell})$.

In addition, we have a parameter ω_{BGS} that must be chosen in order to obtain an effective relaxation. Numerical experiments to guide the choice of this parameter will be presented in Section 5.1. It should be noted that this is similar to the monolithic Vanka relaxation scheme later described in Section 4.1. In fact, these are the same blocks as in that case, except that here the pressure degrees of freedom are not coupled in the relaxation step.

4. MONOLITHIC MULTIGRID

As a second approach, we also investigate monolithic multigrid methods as preconditioners for (9). As opposed to Λ^{-1} above, which was applied only to the vector-Laplacian block, F, we now consider multigrid applied to the whole system (9) at once [15–17, 24].

DOI: 10.1002/nla

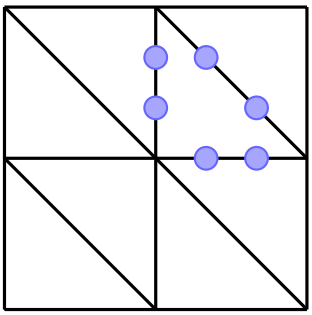


Figure 2. The degrees of freedom in one block for the overlapping block-Gauss-Seidel relaxation scheme. The block consists of the degrees of freedom on the edges corresponding to a given element. Thus, each block has 6 degrees of freedom, and a 6×6 system must be solved at each step of the block Gauss-Seidel iteration.

Again, geometric multigrid is chosen, defining an interpolation operator that acts without coupling between the velocity and pressure degrees of freedom, which take the form

$$P = \begin{bmatrix} P_{\mathbf{u}} & \\ & P_p \end{bmatrix}.$$

The blocks $P_{\mathbf{u}}$ and P_p are the natural finite-element interpolation operators for the BDM_1 and P_0 spaces, respectively. We first consider the Galerkin coarse-grid operators, $A_c = P^T A P$. As above, note that the DG terms present in the F block play no role in defining the interpolation operator $P_{\mathbf{u}}$. In particular, this means that the velocity block of the Galerkin coarse-grid matrix, $F_c = P_{\mathbf{u}}^T F P_{\mathbf{u}}$, is not equivalent to that given by rediscretization on the coarse grid. In Section 5, we will see that for some choices of relaxation scheme, rediscretization leads to better performance than the Galerkin coarse-grid operators. In these cases, we consider rediscretization of the coarse-grid operators as a possible alternative. With the grid-transfer and coarse-grid operators fixed, we turn our attention to the relaxation schemes.

4.1. Vanka Relaxation

The first relaxation scheme that we investigate is Vanka-type relaxation [16]. Vanka-type relaxation schemes can be viewed as overlapping Schwarz methods, updating small, local collections of degrees of freedom at once in either a Gauss-Seidel (multiplicative) or Jacobi (additive) fashion. In this paper, we only consider the multiplicative method, and thus effectively couple the pressure into the block Gauss-Seidel relaxation scheme described above.

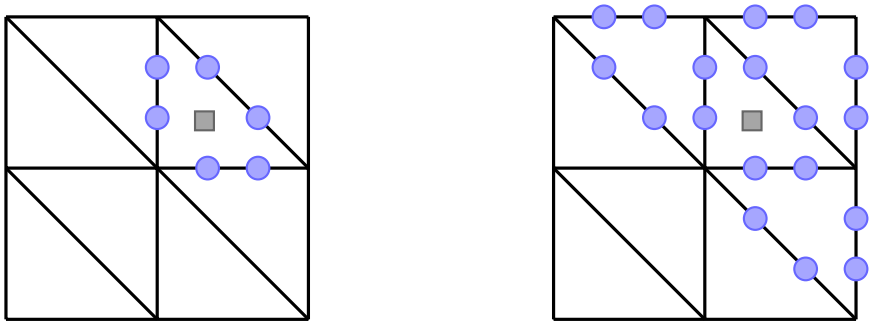
Here, we decompose the set of *all* degrees of freedom into overlapping sets $\{S_\ell\}$ so that each block S_ℓ contains some velocity and some pressure degrees of freedom. Then, the components of the (full) solution vector $x = (u, p)^T$ corresponding to the block S_ℓ are updated as

$$x_\ell = x_\ell + \omega M_{\ell\ell}^{-1} (b - Ax)_\ell, \tag{12}$$

where $M_{\ell\ell}$ is the "Vanka submatrix" corresponding to block ℓ and ω is a diagonal scaling matrix containing underrelaxation weights. Again, for simplicity, the order in which these updates occur is determined by the finite-element software, though other orderings and coloring schemes are possible.

Choices of the blocks, $\{S_\ell\}$, can vary greatly, and is discretization-specific. Two common choices are "element-wise" blocks and "pressure-based" blocks [25]. In the element-wise scheme, each ℓ refers to an element in the mesh and S_ℓ contains all degrees of freedom associated to that element. In the pressure-based scheme, each ℓ refers to a pressure degree of freedom and S_ℓ contains the degrees of freedom that are connected to that pressure degree of freedom in the divergence constraint stencil (i.e. those velocity degrees of freedom corresponding to nonzero column entries in the row of B corresponding to the pressure degree of freedom).

DOI: 10.1002/nla



(a) An "Element-Wise" Vanka block. (b) An extended Vanka block.

Figure 3. The two Vanka blocks considered in this paper. The element-wise Vanka block is shown on the left. This block consists of a single pressure degree of freedom and the 6 velocity degrees of freedom on the same element, resulting in a 7×7 system that needs to be solved at each step of the Gauss-Seidel iteration. The extended Vanka block, adding all velocity degrees of freedom on all elements that share an edge, consists of 18 velocity degrees of freedom and 1 pressure degree of freedom, is shown on the right. This block choice results in a 19×19 system at each step of the iteration.

We consider two choices of blocks in this paper, depicted in Figure 3. The first choice, shown in Figure 3a, is an element-wise block, as the associated degrees of freedom are precisely those on an element. Throughout this paper, this will be called the "element-wise block". However, note that in the case of P_0 pressure discretization, this corresponds to a pressure-based block, as each element has only one pressure degree of freedom and the velocity degrees of freedom associated to that pressure are precisely the velocity degrees of freedom on that element. In the case of this discretization, this results in blocks containing 7 degrees of freedom.

As we will see in Section 5, the element-wise blocks do not always yield a scalable solver for this problem. Thus, we introduce, as a second choice, what we call the "extended" Vanka block (Figure 3b). This is a simple geometric extension of the pressure-based interpretation of the element-wise block. That is, we begin with the element-wise block and add to it only the velocity degrees of freedom associated with elements that share an edge with the original element. For this discretization, the resulting blocks contain up to 19 degrees of freedom (less for boundary elements).

Once the blocks, $\{S_\ell\}$, have been chosen, we consider the formation of the Vanka submatrices, $M_{\ell\ell}$ in (12). Again, we consider two options. The first choice is "full" blocks, which are simply

$$M_{\ell\ell}^{\text{full}} = \begin{bmatrix} F_{\ell\ell} & B_{\ell\ell}^T \\ B_{\ell\ell} & 0 \end{bmatrix},$$

where $B_{\ell\ell}$ and $B_{\ell\ell}^T$ are defined similarly to $F_{\ell\ell}$. Since there is only a single pressure degree of freedom in each block S_ℓ, this means that $B_{\ell\ell}$ is a row vector (and $B_{\ell\ell}^T$, consequently, is a column vector). In this case, $M_{\ell\ell}^{\text{full}} = A_{\ell\ell}$, the restriction of A to rows and columns corresponding to degrees of freedom in block S_ℓ. The methods using this submatrix will be called "full Vanka" if we are using the element-wise blocks and "full extended Vanka" if we are using the extended blocks.

An alternative choice that yields less memory use and less time-per-iteration is the "diagonal" Vanka submatrix [23, 26–28]:

$$M_{\ell\ell}^{\text{diag}} = \begin{bmatrix} \text{diag}(F_{\ell\ell}) & B_{\ell\ell}^T \\ B_{\ell\ell} & 0 \end{bmatrix},$$

where $F_{\ell\ell}$, $B_{\ell\ell}$, and $B_{\ell\ell}^T$ are defined as above. The methods using this submatrix will be called "diagonal Vanka" if we are using the element-wise blocks and "diagonal extended Vanka" if we are using the extended blocks. As we will show in Section 5.4, however, this method will

DOI: 10.1002/nla

require too many more iterations than full Vanka for the reduced cost per iteration to be beneficial.

For the purposes of this investigation, we consider a diagonal scaling matrix, ω, to ensure effective relaxation. The matrix ω has the form

$$\omega = \begin{bmatrix} \omega_u I & \\ & \omega_p \end{bmatrix}, \tag{13}$$

where I is the identity matrix of the appropriate size for the number of velocity degrees of freedom in this block. Numerical experiments to guide the choice of ω_u and ω_p will be presented in Section 5.2.

4.2. Braess-Sarazin Relaxation

Whereas Vanka-type relaxation solves a series of local problems, Braess-Sarazin-type relaxation methods solve global saddle-point problems using simpler approximations to the true system [15]. The "ideal" Braess-Sarazin update takes the form

$$\begin{bmatrix} u \\ p \end{bmatrix}^{\text{new}} = \begin{bmatrix} u \\ p \end{bmatrix}^{\text{old}} + \omega_{\text{BS}} \begin{bmatrix} \alpha_{\text{BS}} C & B^T \\ B & 0 \end{bmatrix}^{-1} \left(\begin{bmatrix} f \\ g \end{bmatrix} - A \begin{bmatrix} u \\ p \end{bmatrix}^{\text{old}} \right), \tag{14}$$

where C is a simple preconditioner for F, the block corresponding to the fluid velocity, α_{BS} is an inner scaling parameter on C, and ω_{BS} is an underrelaxation parameter for the global update.

In (14), we must solve a system:

$$\begin{bmatrix} \alpha_{\text{BS}} C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \delta u \\ \delta p \end{bmatrix} = \begin{bmatrix} r_{\mathbf{u}} \\ r_p \end{bmatrix}. \tag{15}$$

This system can be factorized as

$$\begin{bmatrix} \alpha_{\text{BS}} C & \\ B & S \end{bmatrix} \begin{bmatrix} I & \frac{1}{\alpha_{\text{BS}}} C^{-1} B^T \\ & I \end{bmatrix} \begin{bmatrix} \delta u \\ \delta p \end{bmatrix} = \begin{bmatrix} r_{\mathbf{u}} \\ r_p \end{bmatrix}, \tag{16}$$

where $S = -\frac{1}{\alpha_{\text{BS}}} B C^{-1} B^T$ is the Schur complement. The solution can be computed by the algorithm:

$$S \delta p = r_p - \frac{1}{\alpha_{\text{BS}}} B C^{-1} r_{\mathbf{u}} \tag{17}$$

$$\delta u = \frac{1}{\alpha_{\text{BS}}} C^{-1} (r_{\mathbf{u}} - B^T \delta p). \tag{18}$$

In practice, it suffices to only approximately solve (17), leading to an "inexact" Braess-Sarzin method. In the numerical experiments reported in Section 5, we use a single sweep of Symmetric Gauss-Seidel.

Finally, we must make an appropriate choice for the matrix C. Generally, we choose a preconditioner for F whose inverse is easily computable. One strategy is to consider the diagonal of F, $C_{\text{diag}} = \text{diag}(F)$ [26, 27]. However, in order to improve robustness and convergence, we also consider a block-diagonal preconditioner [23], denoted $C_{\text{blkDiag}} = \text{blkDiag}(F)$, in which the blocks correspond to the two velocity DOFs on an edge in the mesh. Note that this choice of blocks is dependent upon the discretization considered.

5. NUMERICAL RESULTS

To study the relative effectiveness of the preconditioners outlined above, we consider a two-dimensional test problem posed on $\Omega = [0, 1]^2$. We will first consider a regular mesh of triangles

DOI: 10.1002/nla

on Ω to conduct a parameter study; afterwards, we will show the performance of these methods using the identified parameters, both on regular meshes as well as on meshes given by uniform refinement of an unstructured "coarse" mesh.

We use the same example as [5], with analytical solution, $(\mathbf{u}^*, p^*)$:

$$\mathbf{u}^* = \begin{pmatrix} x(1-x)(2x-1)(6y^2-6y+1) \\ y(y-1)(2y-1)(6x^2-6x+1) \end{pmatrix}, \tag{19}$$

$$p^* = x^2 - 3y^2 + \frac{8}{3}xy. \tag{20}$$

The right-hand side vector, $\mathbf{f}$, is computed by substituting (19) and (20) into (1). We fix $\nu = 0.5$ and $\alpha = 4.0$ for all experiments.

To solve the linear systems (9), we use a Krylov subspace method preconditioned with one of the above preconditioners. Note that the matrix A in (9) is symmetric. In the case of the block-diagonal preconditioner with symmetric pointwise Gauss-Seidel relaxation, the preconditioner is symmetric and positive-definite, and thus we use the MINimal RESidual (MINRES) method as the Krylov method; for the other preconditioners, we use the Generalized Minimal RESidual (GMRES) method [1, 29]. The solvers were all implemented in Trilinos [30] and the finite-element discretization was implemented using FEniCS [31].

Remark. As this paper focuses on geometric multigrid methods, it could be possible to implement the resulting algorithms in a matrix-free way using fixed operator stencils on uniform meshes or patches; see, for example, [32]. For all examples, we explicitly form and store all of the matrices involved, and report timings including computation of Galerkin coarse-grid operators in the setup phase. As a result, the reported timings offer insight into potential algebraic multigrid extensions of these methods, although we do not consider this here.

5.1. Block Preconditioning Parameter Study

First, we examine the parameter choices for the block factorization preconditioners, (10) and (11). For these methods, we have only a single parameter for each relaxation scheme to consider, ω_{SOR} for the point method and ω_{BGS} for the block Gauss-Seidel method. In each case, the initial guess is zero, and the appropriate Krylov method is run until the ℓ_2-norm of the residual has been reduced by a relative factor of 10^6. We vary the parameter in steps of 0.1, and report the number of iterations required of the Krylov method.

We begin with the point relaxation method. We see in the left plot of Figure 4 that using a V(1,1)-cycle for Λ^{-1} does not provide mesh-independent convergence, though the optimal parameter choice is clearly $\omega_{\text{SOR}} = 1.0$ for all grid sizes. The use of a W(1,1)-cycle, shown in the right plot of Figure 4, largely mitigates the dependence, but does not eliminate it — it takes 41 iterations to converge on the 32×32 grid and 53 iterations on the 256×256 grid using the block-diagonal preconditioner (10). However, the optimal parameter choice has not changed and remains at $\omega_{\text{SOR}} = 1.0$ for all grid sizes. Parameter choices $\omega_{\text{SOR}} > 1.0$ were considered and found not to yield better convergence. Additionally, the block-diagonal and the block-triangular preconditioners show qualitatively similar behavior, with the latter requiring only slightly fewer iterations to achieve convergence.

Next we consider the methods using block Gauss-Seidel relaxation, shown in Figures 5 and 6. We first use the full submatrices, with results shown in Figure 5. On the left, we see that, as with the point methods, using a single V(1,1)-cycle for Λ^{-1} does not give grid-independent scaling for either the block-diagonal or the block-triangular preconditioner. Additionally, using two or three V(1,1)-cycles for Λ^{-1} led to a reduction in iterations, but not an improvement in scaling. This was found to be generally true for both the block-diagonal and block-triangular preconditioners with all relaxation choices considered. To obtain a scalable algorithm, we can use a W(1,1)-cycle, shown in the plot on the right. Unlike the case of point relaxation, the use of W-cycles leads to perfect scaling across grid sizes for both the block-diagonal and the block-triangular preconditioner, with the latter requiring approximately 5-6 fewer iterations than the

DOI: 10.1002/nla

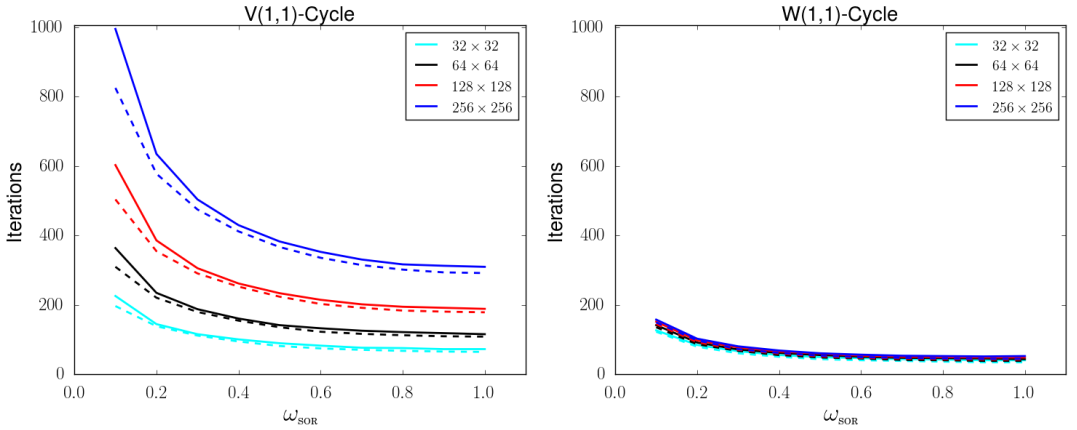


Figure 4. Iteration counts versus SOR relaxation parameter for a block-preconditioned Krylov method using a symmetric sweep of point SOR as the multigrid relaxation scheme. On the left, V(1,1)-cycles are used to precondition the velocity block; on the right, W(1,1)-cycles are used. The solid lines are for MINRES with the block-diagonal preconditioner; the dashed lines are for GMRES with the block-triangular preconditioner.

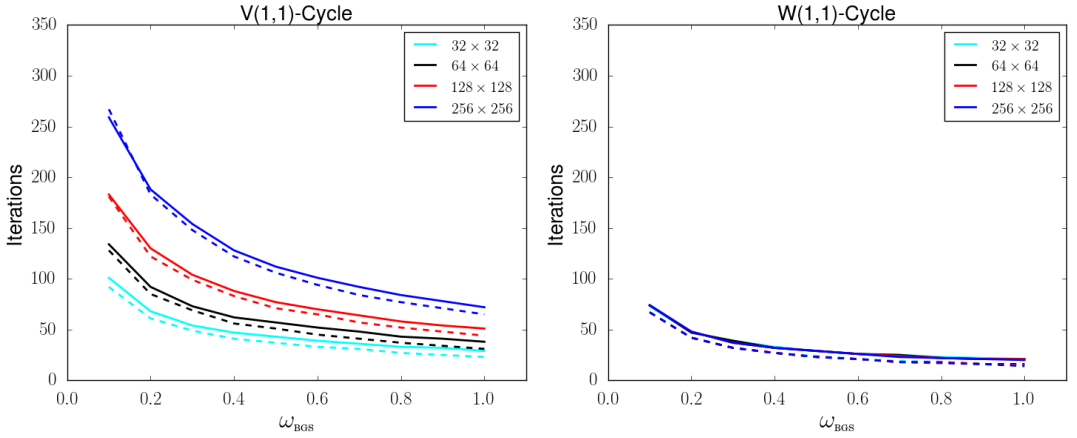


Figure 5. Iteration counts versus block Gauss-Seidel relaxation parameter, ω_{BGS}, for block-preconditioned GMRES using a forward sweep of block Gauss-Seidel with *full* submatrices as the multigrid relaxation scheme. On the left, V(1,1)-cycles are used to precondition the velocity block; on the right, W(1,1)-cycles are used. The solid lines are the block-diagonal preconditioner; the dashed lines are the block-triangular preconditioner.

former. In this case, too, the optimal parameter choice is $\omega_{\text{BGS}} = 1.0$, giving convergence in 20 iterations with the block-diagonal preconditioner or 14 iterations with the block-triangular preconditioner on the 256×256 grid.

We expect that the full method may not be the most computationally efficient method, as it requires the full submatrices to be stored and inverted [23]. To reduce the computational effort required for the iteration, we consider using the diagonal submatrices, with results shown in Figure 6. On the left, we see that the grid dependence is worse for this scheme than any of the others when using a V(1,1)-cycle — in fact, no convergence was seen on the 256×256 grid within the allotted number of iterations. It is clear, however, that $\omega_{\text{BGS}} = 0.7$ is the best parameter choice. Using, instead, a W(1,1)-cycle, is able to somewhat mitigate the grid dependence, though not eliminate it completely. The optimal parameter remains the same as with V-cycles, namely $\omega_{\text{BGS}} = 0.7$. With this optimal parameter, the method required 47 iterations to achieve convergence on the 32×32 grid and 56 iterations on the 256×256 grid.

DOI: 10.1002/nla

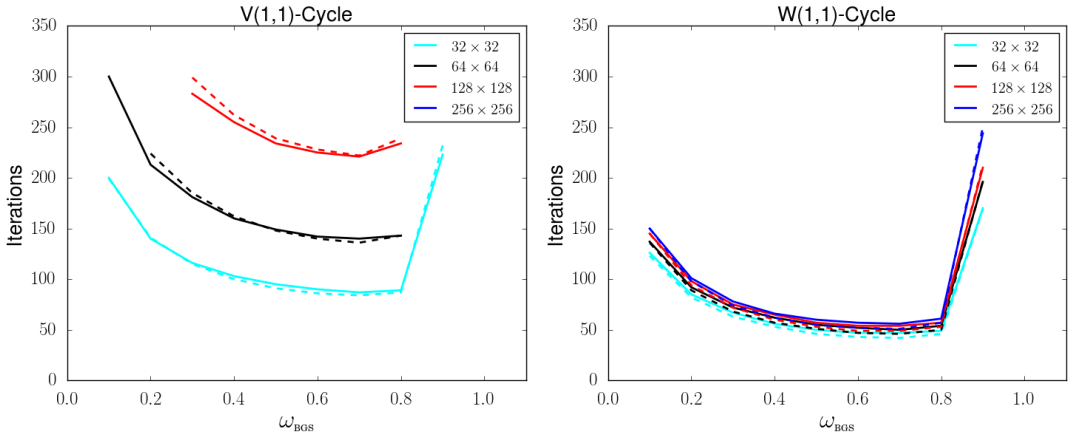


Figure 6. Iteration counts versus block Gauss-Seidel relaxation parameter, ω_{BGS}, for block-preconditioned GMRES using a forward sweep of block Gauss-Seidel with *diagonal* submatrices as the multigrid relaxation scheme. On the left, V(1,1)-cycles are used to precondition the velocity block; on the right, W(1,1)-cycles are used. The solid lines are the block-diagonal preconditioner; the dashed lines are the block-triangular preconditioner. The absence of a data point means that the Krylov method did not converge to the desired tolerance within 300 iterations.

In summary, the block-diagonal and block-triangular preconditioners (10) and (11) lead to scalable methods when a W(1,1)-cycle with the properly chosen relaxation scheme is used for Λ^{-1}. Moreover, there is no qualitative difference between the performance of the block-diagonal and the block-triangular preconditioners. In general, the block-triangular method requires about 5-6 fewer iterations than the block-diagonal method for all relaxation choices at all grid sizes. In particular, using symmetric pointwise Gauss-Seidel sweeps or block Gauss-Seidel with diagonal submatrices does not provide a truly grid-independent precondtioner. However, block Gauss-Seidel relaxation with full submatrices gives consistent performance across all grid sizes in the context of both block preconditioners.

5.2. Monolithic multigrid with Vanka relaxation parameter study

We now turn our attention to the monolithic multigrid preconditioners, for which there are two parameters in each relaxation scheme, ω_u and ω_p for the Vanka methods and ω_{BS} and α_{BS} for the Braess-Sarazin methods. In this case, these parameters are each varied independently in steps of 0.1.

We begin our investigation of the full Vanka scheme with a two-grid iteration. We find this method to offer grid-independent convergence across all grid sizes. With the convergence of the two-grid method established, we proceed to the full V-cycle. The results are shown in the top row of Figure 7. We see that on the 32×32 grid (4-level), the stopping criterion was met in 14 iterations for $(\omega_u, \omega_p) = (1.0, 0.6)$. However, on the 256×256 grid (7-level), we observe no convergence at all for this parameter choice, and the new optimal parameter choice is $(\omega_u, \omega_p) = (0.9, 0.1)$, giving convergence in 49 iterations. Thus, there is a strong grid dependence both on the optimal parameter choice as well as the convergence of the algorithm.

Since the two-grid method is successful, a natural expectation is that the use of W-cycles will also lead to good convergence. The middle row of Figure 7 shows that this is not the case. Here we see that the convergence is excellent on the 32×32 grid; however, we have only a very small region of convergence on the 256×256 grid. In this case, we believe that the difficulty is caused by a mismatch between the Galerkin coarse-grid operators, $A_c = P^T A P$, and the chosen relaxation scheme. To test this, we, instead, rediscretize the system on the coarse grids. The results are shown in the bottom row of Figure 7. After switching from Galerkin to rediscretized coarse-grid operators, we see a clearly defined range of optimal parameter choices that lead to scalable methods for the V(1,1)-cycle preconditioner. In this case, the optimal

DOI: 10.1002/nla

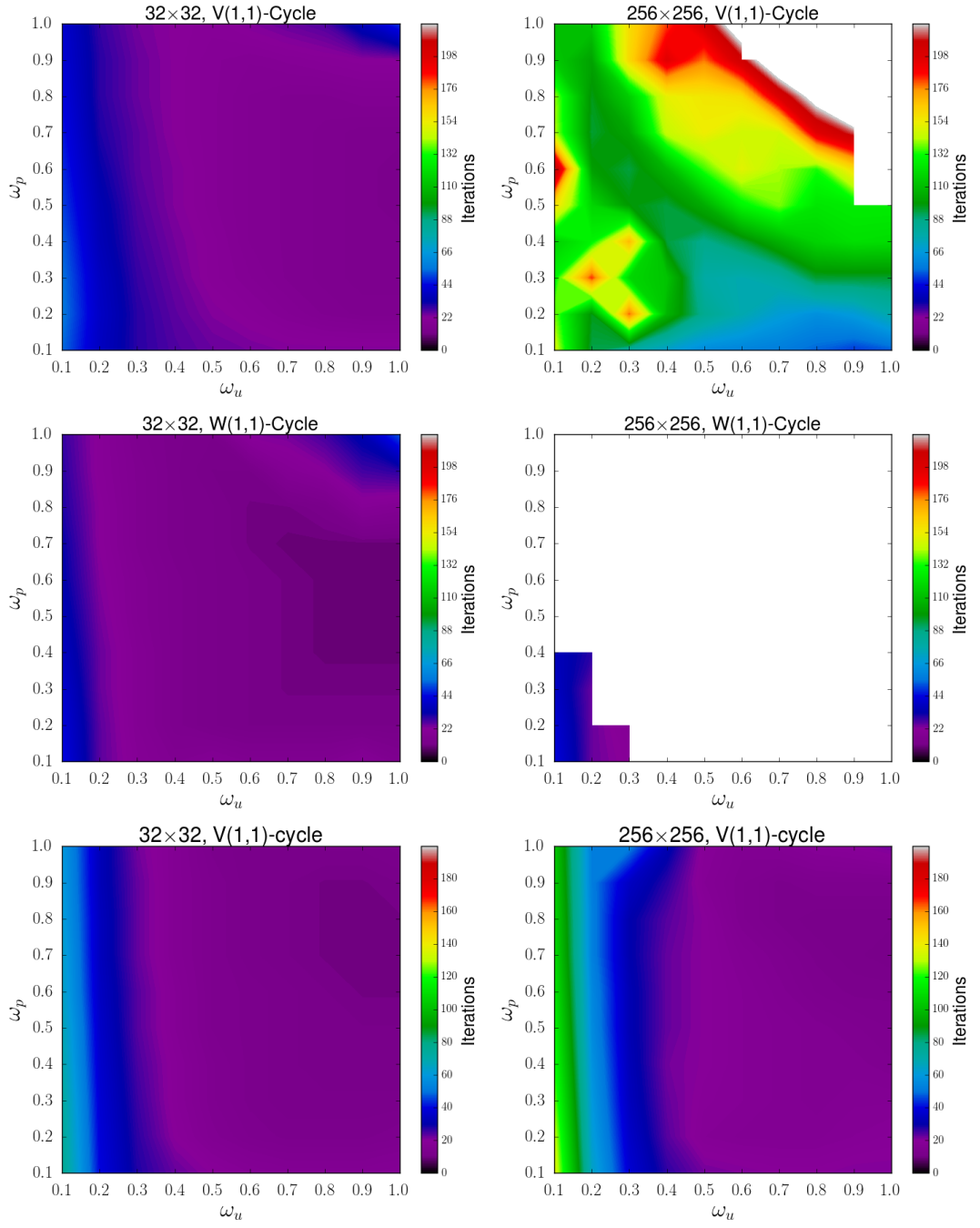


Figure 7. GMRES iteration counts with monolithic multigrid preconditioning using Vanka relaxation with the element-wise blocks and *full* submatrices as the relaxation scheme. The top row shows a V(1,1)-cycle with Galerkin coarsening; the middle row shows a W(1,1)-cycle with Galerkin coarsening; and the bottom row shows a V(1,1)-cycle with rediscretized coarse-grid operators. Results for the 32×32 grid appear on the left; results for the 256×256 grid appear on the right. The absence of a data point means that the Krylov method did not converge to the desired tolerance within 300 iterations.

choice is $(\omega_u, \omega_p) = (1.0, 0.7)$, leading to convergence in 10 iterations on the 32×32 grid and 11 iterations on the 256×256 grid. We also observe that the W(1,1)-cycle preconditioner was scalable with rediscretized coarse-grid operators.

DOI: 10.1002/nla

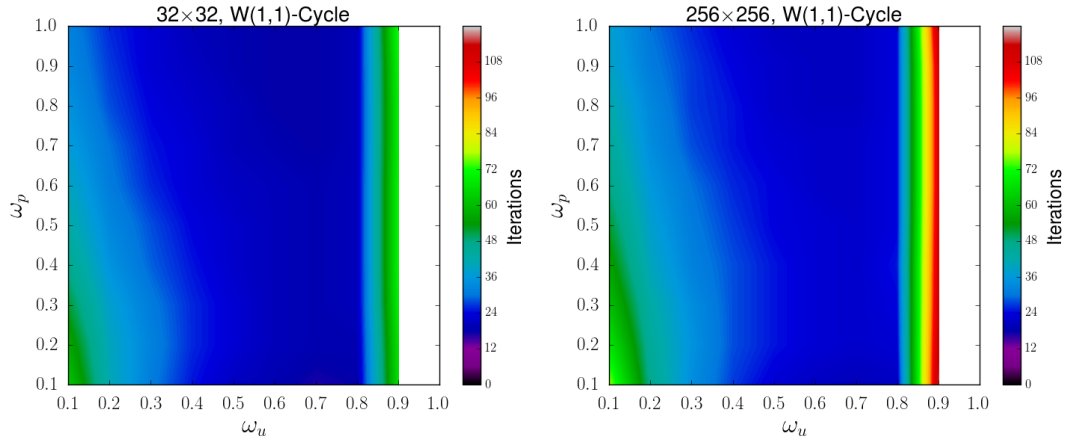


Figure 8. GMRES iteration counts with the W(1,1)-cycle monolithic multigrid preconditioner with Vanka relaxation using the element-wise blocks and *diagonal* submatrices as the relaxation scheme. On the left are results for the 32×32 grid; on the right are results for the 256×256 grid. The absence of a data point means that the Krylov method did not converge to the desired tolerance within 100 iterations.

Next, we consider the monolithic multigrid preconditioner with diagonal element-wise Vanka relaxation and observe that the two-grid preconditioner provides consistent iteration counts across grid sizes. Extending to the full V(1,1)-cycle, we note that this preconditioner, too, is unable to provide grid-independent convergence and behaves qualitatively similarly to the V(1,1)-cycle with full element-wise Vanka. Based on the intuition gained from the success of the two-grid preconditioner, we then employ W(1,1)-cycles, which, as shown in Figure 8, provide a scalable algorithm for this problem with $(\omega_u, \omega_p) \in [0.6, 0.7] \times [0.8, 1.0]$ as the optimal parameter range, giving convergence in 19-21 iterations. Since the W(1,1)-cycle preconditioner gives grid-independent convergence of GMRES, we do not consider rediscretization of the coarse-grid operator in this case.

Similar results can be shown for the Extended Vanka methods. In both the case of full submatrices and diagonal submatrices, GMRES with the two-grid preconditioner shows good, grid-independent convergence, while GMRES with the V(1,1)-cycle preconditioner does not, with convergence diminishing as grid size grows. However, upon switching to W(1,1)-cycles, both the preconditioner with extended full Vanka relaxation and extended diagonal Vanka relaxation offer a stable parameter range for (ω_u, ω_p) that offers consistent performance across grid sizes. In particular, choosing $(\omega_u, \omega_p) \in [0.7, 1.0] \times [0.5, 1.0]$ for the preconditioner with extended full Vanka relaxation gives convergence in 6 iterations on all grids. Choosing $(\omega_u, \omega_p) \in [0.5, 0.6] \times [0.4, 0.6]$ for the preconditioner with extended diagonal Vanka relaxation gives convergence in 15-17 iterations on all grids. As with the case of the preconditioner with element-wise diagonal Vanka relaxation, we do not consider the multigrid method with rediscretization of the coarse-grid operators here because of the success of the W(1,1)-cycle preconditioner.

In summary, we see that each of the Vanka-type relaxation methods gives rise to a two-grid preconditioner with grid-independent convergence. However, upon extending to a full V(1,1)-cycle, none scale with Galerkin coarsening. In the case of element-wise full Vanka, a W(1,1)-cycle was ineffective as well, requiring the use of rediscretized coarse-grid operators to obtain grid independence. In the other cases, the use of a W(1,1)-cycle is enough to provide good scaling properties across grid sizes. In the end, extended full Vanka gives the optimal performance in terms of iteration counts, requiring only 6 iterations of GMRES to converge, with a large, stable region of parameter choices that give optimal convergence.

DOI: 10.1002/nla

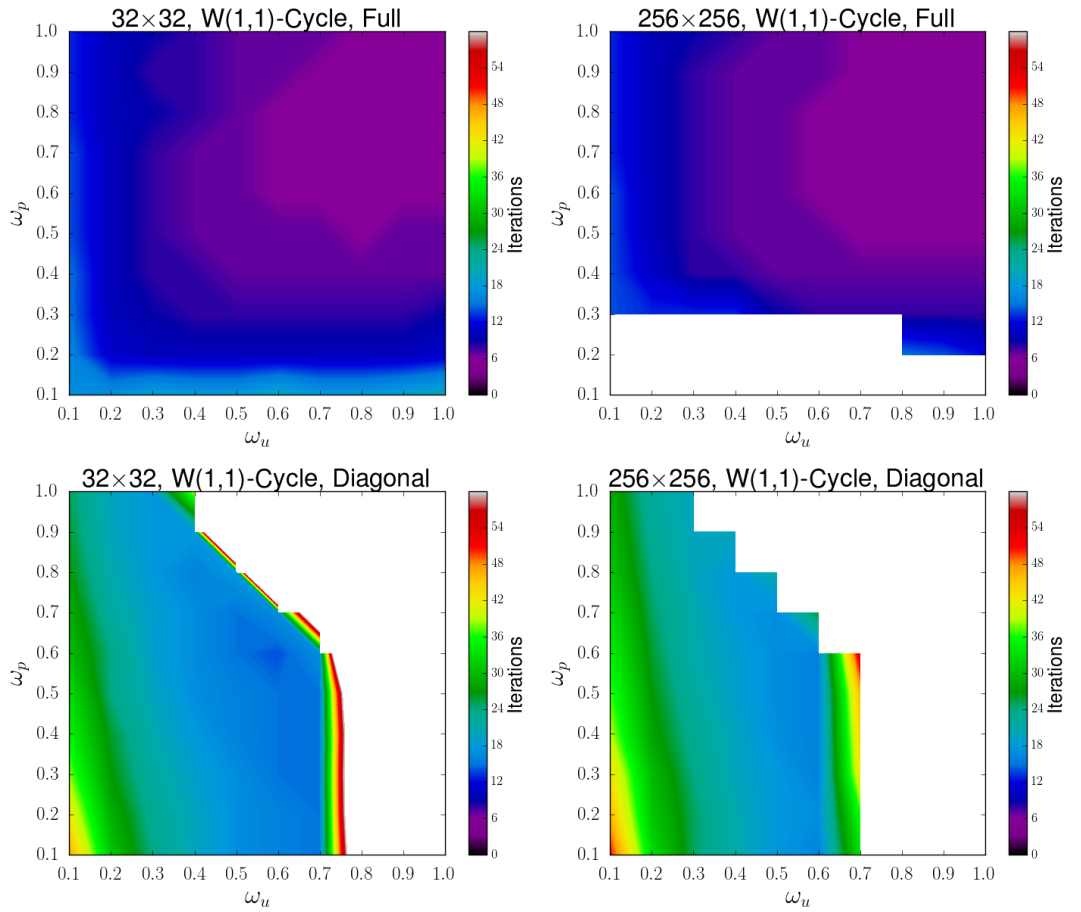


Figure 9. GMRES iteration counts with the W(1,1)-cycle monolithic multigrid preconditioner using Vanka relaxation with extended blocks. The top row shows results using the *full* submatrices; the bottom row for *diagonal* submatrices. On the left are results for the 32×32 grid; on the right are results for the 256×256 grid. The absence of a data point means that the Krylov method did not converge to the desired tolerance within 300 iterations.

5.3. Monolithic multigrid with Braess-Sarazin relaxation parameter study

For the monolithic multigrid preconditioner with Braess-Sarazin relaxation (14), we tell a similar story. The case of diagonal Braess-Sarazin, in which $C = \text{diag}(F)$, is shown in Figure 10. In the top row are results using a V(1,1)-cycle. While the optimal parameter choice $(\omega_{\text{BS}}, \alpha_{\text{BS}}) = (0.9, 2.1)$ is independent of the grid size, the performance of the algorithm certainly is not, costing 42 iterations on the 32×32 grid and 124 iterations on the 256×256 grid. As shown in the bottom row of Figure 10, the W(1,1)-cycle preconditioner leads to more consistent performance across grid sizes, ranging only from 28 iterations on the 32×32 grid to 32 iterations on the 256×256 grid. In this case, the best parameter choice for all grid sizes is observed to be $(\omega_{\text{BS}}, \alpha_{\text{BS}}) = (0.8, 2.0)$.

Results for the W(1,1)-cycle preconditioner using the block-diagonal variant of the Braess-Sarazin method are shown in Figure 11. The results for the V(1,1)-cycle preconditioner were qualitatively similar to the previous case, again proving unsuccessful in providing a scalable algorithm, with 30 iterations required for the 32×32 grid and 74 iterations required for the 256×256 problem. Using a W(1,1)-cycle instead, we see in Figure 11 that we have consistent performance across grid sizes, requiring 22 iterations on the 32×32 grid, stabilizing to 24-25 iterations for finer grids. However, it is interesting to note that the region in which the

DOI: 10.1002/nla

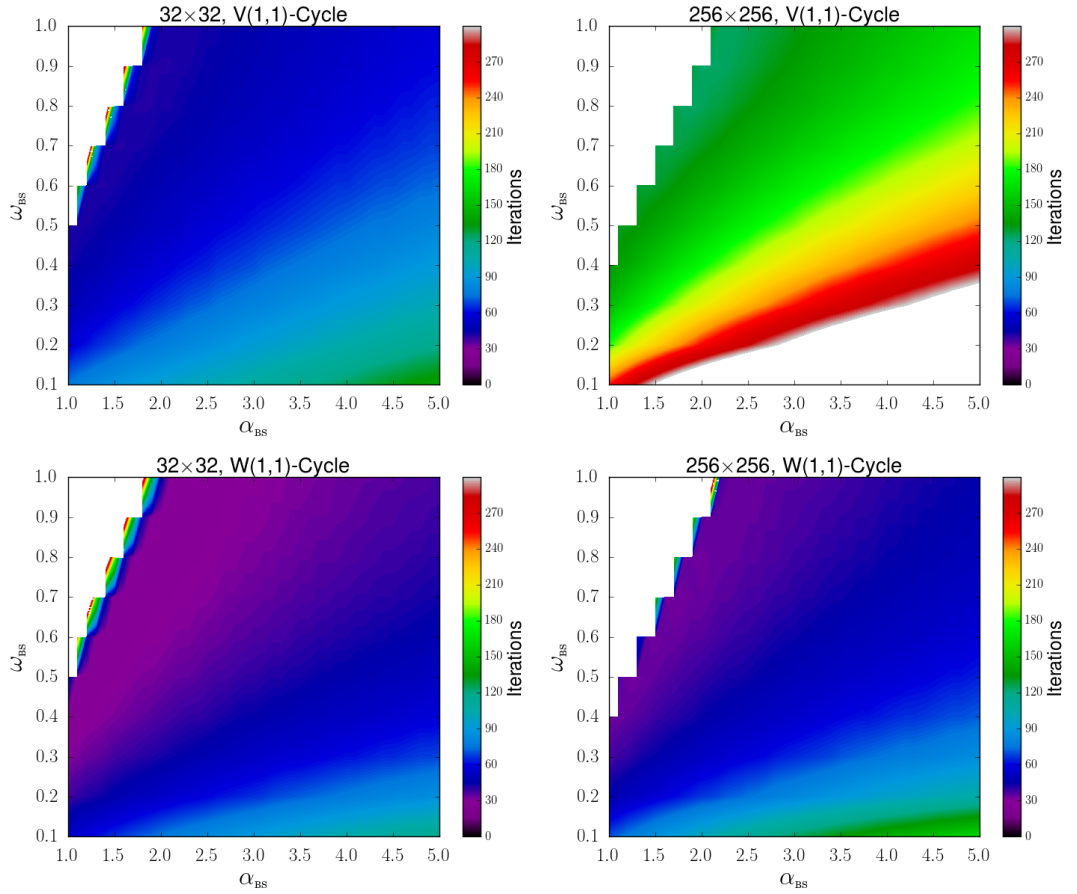


Figure 10. GMRES iteration counts with monolithic multigrid preconditioning using Braess-Sarazin with $C = \mathrm{diag}(F)$ as the relaxation scheme. On the left are results for the 32×32 grid; on the right are results for the 256×256 grid. The top row shows V(1,1)-cycles; the bottom row shows W(1,1)-cycles. The absence of a data point means that the Krylov method did not converge to the desired tolerance within 500 iterations.

method does not converge increases with each grid refinement. The optimal parameters for a given grid lie adjacent to the boundary of this region, and therefore shift with each grid refinement. A parameter choice that leads to optimal performance on the 32×32 grid is $(\omega_{\mathrm{BS}}, \alpha_{\mathrm{BS}}) = (0.8, 1.2)$. However, on the 256×256 grid, this value is no longer optimal, giving convergence in 161 iterations; a better choice of $(\omega_{\mathrm{BS}}, \alpha_{\mathrm{BS}}) = (0.8, 1.4)$ for this grid size yields convergence in 25 iterations.

This illustrates another important characteristic of both Braess-Sarazin-based methods: it is better to pick α_{BS} to be higher than the optimal choice, lest the choice lead to an algorithm that is too slow to converge. Picking $(\omega_{\mathrm{BS}}, \alpha_{\mathrm{BS}}) = (0.8, 1.4)$ on the 32×32 grid gives convergence in 24 iterations instead of the optimal 22, whereas picking $(\omega_{\mathrm{BS}}, \alpha_{\mathrm{BS}}) = (0.8, 1.2)$ on the 256×256 grid gives convergence in 161 iterations instead of the optimal 25. More generally, for each of these methods, we have identified a large region of parameter choices that give near-optimal convergence across a wide range of grid sizes.

Thus we have seen that both block factorization methods and monolithic multigrid methods lead to effective preconditioners when W(1,1)-cycles are used and when a proper relaxation scheme is chosen. In this case, the block preconditioners with full block Gauss-Seidel relaxation and the monolithic multigrid preconditioner with extended full Vanka relaxation give the best performance, with iteration counts decreasing with increasing grid size in the former case and

DOI: 10.1002/nla

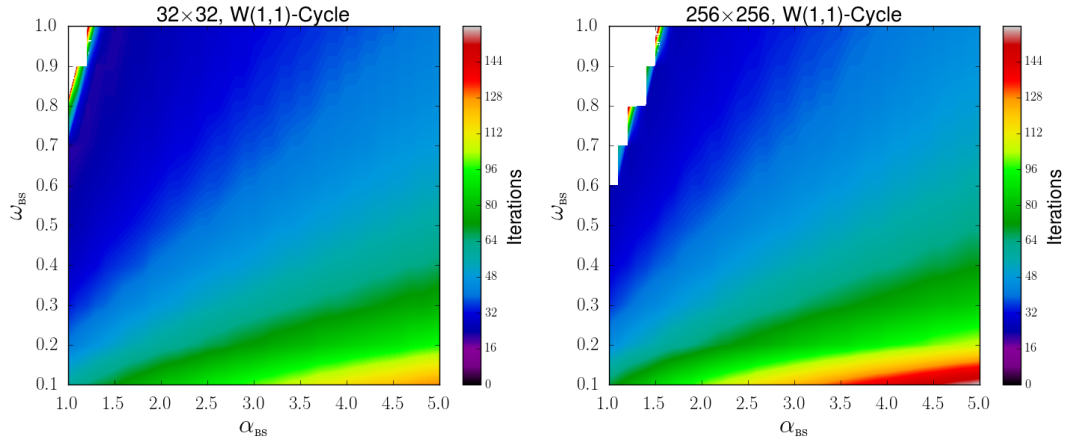


Figure 11. GMRES iteration counts using the W(1,1)-cycle monolithic multigrid preconditioner using Braess-Sarazin relaxation with $C = \text{blkDiag}(F)$ as the relaxation scheme. On the left are results for the 32×32 grid; on the right are results for the 256×256 grid. The absence of a data point means that the Krylov method did not converge to the desired tolerance within 500 iterations.

Table I. Number of multigrid levels used for each grid size and the number of degrees of freedom in the global problem.

Grid size	32×32	64×64	128×128	256×256	512×512
Multigrid Levels	4	5	6	7	8
Degrees of Freedom	8,320	33,024	131,584	525,312	2,099,200

remaining constant in the latter. In addition, the other monolithic multigrid relaxation schemes generally show better scaling than the other block-factorization preconditioners.

5.4. Performance Analysis

In this section, we consider the computational time required of each of these preconditioning strategies. All tests are performed in serial and run on a machine with 2 Intel Xeon E5-2650 v2 CPUs at 2.60 GHz configured with 128GB DDR3 RAM clocked at 1866MHz. We begin by discussing the setup requirements of each method, and then considering the solve phase. For these tests, we ran the simulations on five grid sizes, from 32×32 to 512×512. In each case, we coarsen by a factor of two in each direction down to a coarse grid of 4×4 elements, where a direct solve is used to solve the coarse-grid problem (112 degrees of freedom for velocity; 32 for pressure). Table I shows the number of levels in the multigrid hierarchy and the number of fine-grid degrees of freedom.

For the 512×512 problem, we use the optimal parameters for each solver as determined in the previous section. This means that we use the same parameter choices on each grid, with the one exception of the method using the block-diagonal Braess-Sarazin relaxation scheme, where we need to step α_{BS} by 0.1 about every two grids. Thus, we use $(\omega_{\text{BS}}, \alpha_{\text{BS}}) = (0.8, 1.2)$ on the 32×32 test problem, $(\omega_{\text{BS}}, \alpha_{\text{BS}}) = (0.8, 1.3)$ on the 64×64 and 128×128 test problems, and $(\omega_{\text{BS}}, \alpha_{\text{BS}}) = (0.8, 1.4)$ for the 256×256 and 512×512 test problems. As discussed above, we could also have used the last set of parameter values for all grids with only slight degradation in the coarsest grid results.

In the case of block preconditioning, the setup time is the cost of computing the multigrid hierarchy for the velocity block and computing the inverse of the pressure mass matrix; there should be only negligible difference between the block-diagonal and block-triangular *setup* cost. The multigrid setup time includes computing all of the grid-transfer operators, computing the coarse-grid operator on each grid, and any setup involved in the relaxation method. In the case of the point methods, the last category is nearly negligible, whereas the block methods

DOI: 10.1002/nla

Table II. The setup time in seconds of block-factorization preconditioners for various grid sizes, from 32×32 to 512×512.

	32×32	64×64	128×128	256×256	512×512
Block-Diagonal Point SGS	0.022	0.083	0.33	1.31	5.21
Block-Diagonal Full Blk-GS	0.033	0.13	0.50	1.99	8.04
Block-Diagonal Diagonal Blk-GS	0.032	0.12	0.49	1.92	7.76
Block-Triangular Point SGS	0.022	0.084	0.33	1.31	5.34
Block-Triangular Full Blk-GS	0.034	0.13	0.50	1.99	8.01
Block-Triangular Diagonal Blk-GS	0.032	0.12	0.48	1.95	7.64

Table III. The setup time in seconds of monolithic multigrid preconditioners for various grid sizes, from 32×32 to 512×512.

	32×32	64×64	128×128	256×256	512×512
Diagonal Vanka	0.048	0.19	0.75	2.97	11.96
Full Extended Vanka	0.083	0.29	1.18	4.76	19.23
Diagonal Extended Vanka	0.054	0.21	0.85	3.45	13.81
Diagonal Braess-Sarazin	0.026	0.10	0.41	1.65	6.77
Block-Diagonal Braess-Sarazin	0.025	0.10	0.41	1.66	6.83

require the computation of the blocks and the extraction and factorization of the appropriate 6×6 submatrices. Table II shows the setup cost of the block-factorization preconditioners. As expected, the point method has the smallest setup time as the result of the nearly trivial cost of the relaxation setup. Furthermore, the block Gauss-Seidel approach with full submatrices is more expensive than that with diagonal submatrices. Finally, we observe that the setup times are scaling by the desired factor of four in every case.

For the monolithic multigrid methods, we must compute two grid transfer operators at each level, $P_{\mathbf{u}}$ and P_p, and block 2×2 coarse-grid operators for each coarse level in the hierarchy. Also, the relaxation schemes each require nontrivial setup. For the Braess-Sarazin methods, we must compute C (and C^{-1}) and the triple product $S = -\frac{1}{\alpha_{\mathrm{BS}}} B C^{-1} B^T$. Finally, for the Vanka methods, we must compute the element-wise or the extended blocks, and then extract and factor the 7×7 or 19×19 (resp.) submatrices. Table III shows the setup cost of the monolithic multigrid methods that we consider here. Since neither V- nor W-cycles with full Vanka relaxation with Galerkin coarsening lead to a perfectly scalable method, we do not consider that option here. We see that the Braess-Sarazin methods require the least setup time, by at least a factor of two over the Vanka methods. Furthermore, there is little difference in the setup cost between the diagonal and the block-diagonal Braess-Sarazin methods. For the Vanka methods, we see that the element-wise diagonal method has a slight time advantage over the extended methods, and that the diagonal extended method requires less time than the full extended method, as we would expect. Again, the setup times scale by a factor of about four.

For the solve phase, we consider the time required for preconditioned MINRES or preconditioned GMRES to reduce the residual norm by a relative factor of 10^6, starting from a zero initial guess. The results for the methods with the various block-factorization preconditioners are shown in Table IV. The first observation from this table is that the block Gauss-Seidel method with full submatrices requires the fewest number of iterations to converge, and that it has the best scaling across grid sizes. A second observation is that the point method is the fastest method, both per-iteration and overall. In particular, on the 512×512 grid with the block-diagonal preconditioner, the point method requires about 0.61 seconds per iteration and the block method requires about 5.45 seconds per iteration for the full submatrices and 4.65 seconds per iteration for the diagonal submatrices. That is, the block method with full submatrices is 9 times more costly per iteration than the point method, which requires only 2.84 times as many iterations. We note here that the block Gauss-Seidel methods scale much better in terms of iteration counts, and, as such, they may have some advantatge if considering very large problem sizes.

DOI: 10.1002/nla

Table IV. The solve time in seconds of the methods utilizing block-factorization preconditioners for various grid sizes, from 32×32 to 512×512. The number of required MINRES or GMRES iterations is shown in parentheses next to the time.

	32×32	64×64	128×128	256×256	512×512
Block-Diagonal Point SGS	0.13 (41)	0.46 (44)	1.94 (49)	8.18 (53)	32.68 (54)
Block-Diagonal Full Blk-GS	0.37 (21)	1.57 (21)	6.36 (21)	25.26 (20)	103.50 (19)
Block-Diagonal Diagonal Blk-GS	0.68 (47)	3.12 (50)	14.27 (54)	63.61 (56)	264.80 (57)
Block-Triangular Point SGS	0.13 (35)	0.49 (39)	2.09 (43)	8.65 (45)	37.79 (48)
Block-Triangular Full Blk-GS	0.27 (16)	1.22 (16)	4.82 (15)	19.10 (14)	76.19 (14)
Block-Triangular Diagonal Blk-GS	0.59 (42)	2.96 (46)	13.10 (49)	56.01 (51)	246.63 (53)

Table V. The solve time in seconds of monolithic multigrid preconditioners for various grid sizes, from 32×32 to 512×512. The number of required GMRES iterations is shown in parentheses next to the time.

	32×32	64×64	128×128	256×256	512×512
Diagonal Vanka	1.01 (18)	4.68 (19)	20.86 (20)	91.39 (21)	383.13 (22)
Full Extended Vanka	0.29 (6)	1.30 (6)	5.47 (6)	22.70 (6)	93.68 (6)
Diagonal Extended Vanka	0.73 (15)	3.32 (15)	14.78 (16)	60.94 (16)	248.47 (16)
Diagonal Braess-Sarazin	0.14 (28)	0.49 (30)	1.85 (32)	7.14 (33)	30.86 (35)
Block-Diagonal Braess-Sarazin	0.11 (22)	0.38 (24)	1.34 (24)	5.21 (25)	22.04 (26)

The added cost of the preconditioners employing the block Gauss-Seidel relaxation scheme primarily comes from the need for the relaxation method to update the system residual after each block update. Of course, the submatrix solves are more expensive than the "point solves" (a single scalar multiplication and sum); however, the sum of the work required for the submatrix solves is much less than the sum of the residual updates [23].

The results for the monolithic multigrid preconditioners are shown in Table V. One observation from this table is that the full extended Vanka method provides the best iteration counts for this problem and scales perfectly across grid sizes. However, it is very expensive on a per-iteration basis, requiring 15.61 seconds per iteration for the 512×512 problem. On the other hand, we see that the Braess-Sarazin-based preconditioning methods provide the best solve times overall, despite requiring several more iterations of the Krylov method. The method using the block-diagonal Braess-Sarazin relaxation requires only 0.85 seconds per iteration for the 512×512 problem. So, the block-diagonal Braess-Sarazin method requires 4.33 times as many iterations as the full extended Vanka method, but requires only 0.055 as much time per iteration. As the iteration counts for the block-diagonal Braess-Sarazin-based method are stable, it will likely be superior to the Vanka-based approaches on all feasible problem sizes.

In Figure 12, we show the total time to solution on a log scale for each method considered above for the 512×512 grid. It is clear that the monolithic multigrid preconditioner with block-diagonal Braess-Sarazin relaxation leads to the shortest time to solution. Also, the monolithic multigrid preconditioner with diagonal Braess-Sarazin relaxation and the block preconditioners with point relaxation require approximately the same amount of time. The next group were the methods based on full block Gauss-Seidel methods: block preconditioning with full block Gauss-Seidel relaxation and monolithic multigrid with full extended Vanka relaxation. The methods that took the most time to solution were those based on diagonal block Gauss-Seidel, again emphasizing that reduced setup cost and solve time per iteration is not enough to compensate for the high iteration counts of these methods.

5.5. Solver performance on an unstructured mesh

As a final test, the preconditioners are tested on a mesh generated by uniform refinements of the unstructured coarse-grid mesh shown in Figure 13. The number of grid refinements used and the resulting fine-grid number of degrees of freedom are shown in Table VI. The number of grid refinements is equal to the number of levels in the multigrid hierarchies, and a direct solve is used on the coarsest grid (720 velocity degrees of freedom and 228 pressure degrees of

DOI: 10.1002/nla

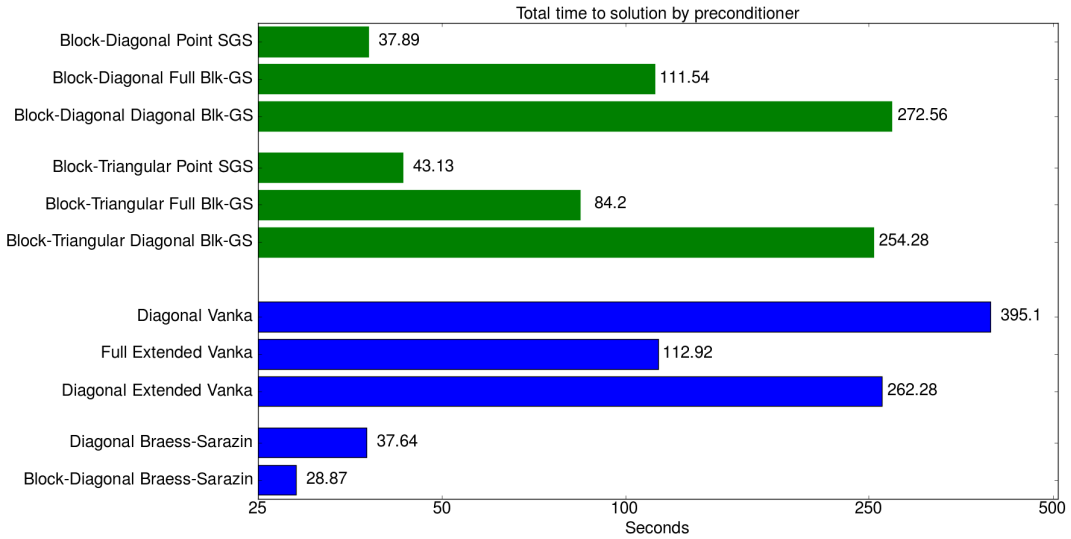


Figure 12. Total time to solution on a log scale of all of the methods considered on a uniform 512×512 mesh. The results for the block-preconditioning methods are shown in green, and the results for the monolithic multigrid preconditioning methods are shown in blue.

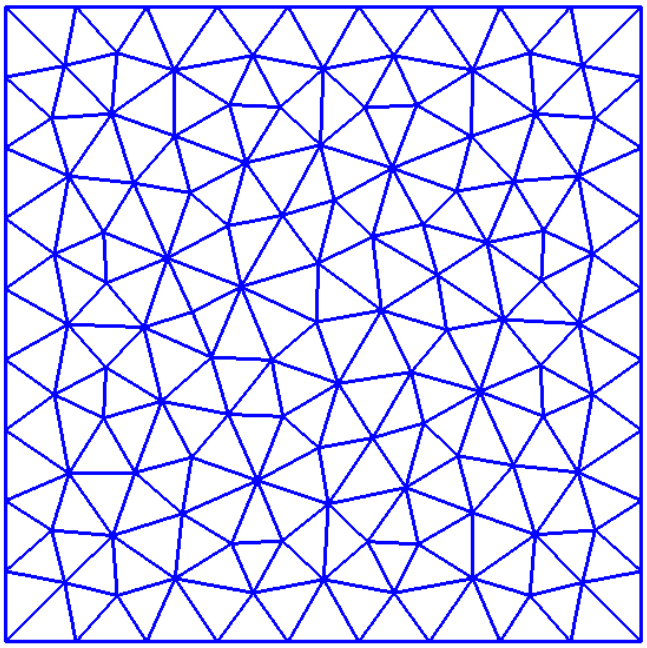


Figure 13. The unstructured coarse mesh.

Table VI. Number of degrees of freedom in the global problem for each number of grid refinements of the unstructured coarse mesh.

Grid Refinements	4	5	6	7	8
Degrees of Freedom	58,658	234,048	935,040	3,737,856	14,946,816

freedom). Note that using 6 refinements leads to a system nearly half as large as the uniform 512×512 mesh above, and using 7 refinements leads to a system nearly twice as large.

Prepared using nlaauth.cls
DOI: 10.1002/nla

Table VII. Solver performance in seconds for the tests on the refined unstructured coarse mesh with various levels of refinements. In parentheses is the number of iterations required of the Krylov solver to attain convergence to a relative tolerance of 10^{-6} from a zero initial guess.

	4-level	5-level	6-level	7-level	8-level
Block-Diagonal Point SGS	0.79 (38)	3.51 (43)	15.21 (47)	64.97 (50)	269.45 (51)
Block-Diagonal Full Blk-GS	2.72 (19)	11.27 (19)	46.45 (18)	194.08 (18)	770.07 (18)
Block-Diagonal Diagonal Blk-GS	4.48 (41)	21.26 (44)	97.85 (47)	411.20 (50)	1743.03 (51)
Block-Triangular Point SGS	0.81 (33)	3.73 (37)	16.23 (41)	70.09 (43)	300.61 (45)
Block-Triangular Full Blk-GS	2.07 (14)	9.12 (14)	36.62 (14)	150.97 (14)	632.01 (14)
Block-Triangular Diagonal Blk-GS	4.23 (38)	20.66 (41)	91.28 (44)	382.33 (46)	1585.66 (47)
Diagonal Braess-Sarazin	0.77 (25)	3.33 (27)	13.65 (29)	59.80 (31)	250.01 (32)
Block-Diagonal Braess-Sarazin	0.65 (20)	2.85 (22)	11.57 (23)	49.18 (24)	196.90 (24)
Diagonal Vanka	7.06 (16)	32.54 (17)	142.29 (18)	610.59 (19)	2434.00 (19)
Full Extended Vanka	2.48 (5)	10.48 (5)	43.79 (5)	174.54 (5)	716.55 (5)
Diagonal Extended Vanka	5.35 (13)	24.22 (14)	100.96 (14)	412.71 (14)	1754.88 (15)

For these experiments, we use the same parameters for each preconditioner as we used for the above experiments on the uniform mesh, demonstrating the relative insensitivity of these parameter choices to the underlying mesh. The total time to solution and iteration counts for each preconditioner are given in Table VII. Most generally, we see that all solvers behave as predicted above. In particular, the block preconditioners using pointwise Gauss-Seidel have the worst grid dependence of all preconditioners in terms of iteration counts, followed by the block preconditioners using diagonal block Gauss-Seidel. The monolithic multigrid methods all show good scaling properties, as do the block preconditioners with full block Gauss-Seidel relaxation. The best iteration counts are obtained using monolithic multigrid with extended full Vanka relaxation. However, the best time for all levels of refinement is given by monolithic multigrid preconditioning with block-diagonal Braess-Sarazin-type relaxation.

6. CONCLUSIONS AND FUTURE DIRECTIONS

We have presented numerical results for two families of multigrid-based preconditioners for a DG discretization of the Stokes equations, showing that block-factorization approaches as well as monolithic multigrid approaches can be designed that provide scalable algorithms for this problem. In particular, we have shown that block-diagonal and block-triangular preconditioners with both pointwise and block Gauss-Seidel methods, as well as monolithic geometric multigrid preconditioners using Braess-Sarazin-type and Vanka-type relaxation lead to effective preconditioners for this problem. The block preconditioners with full block Gauss-Seidel relaxation and monolithic multigrid with extended full Vanka are shown to give the best scaling properties, with the later giving the best iteration counts. However, the best method for total time to solution is using a monolithic multigrid preconditioner with block-diagonal Braess-Sarazin-type relaxation.

In our previous work [23], we have extended the families of monolithic relaxation strategies shown here to an $\mathbf{H}^1$-conforming discretization of two-dimensional resistive magnetohydrodynamics (MHD). In the future, we will extend these methods to MHD problems that utilize $\mathbf{H}(\text{div})$-conforming discretizations similar to those shown here, such as described in [2]. In addition, here, we relied upon geometric multigrid approaches. This could be relaxed to considering algebraic multigrid approaches for this problem. However, in the case of block preconditioning, this would require an AMG-style interpolation operator for the DG-BDM_1 velocity block. In the case of monolithic multigrid, we would require grid transfer operators that preserve the saddle-point block structure for the relaxation methods to be meaningful, in addition to dealing with the DG influence effectively on coarse grids. Finally, we are developing parallel implementations of the monolithic Vanka methods described above. This involves

DOI: 10.1002/nla

appropriate choices of blocks on the processor boundaries, as well as choosing the Schwarz-style updating scheme properly. In particular, we will do multiplicative updates on-processor and additive updates across processor.

REFERENCES

1. Elman H, Silvester D, Wathen A. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation, Oxford University Press: New York, 2005.
2. Greif C, Li D, Schötzau D, Wei X. A mixed finite element method with exactly divergence-free velocities for incompressible magnetohydrodynamics. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(4548):2840 – 2855, doi:10.1016/j.cma.2010.05.007. URL `http://www.sciencedirect.com/science/article/pii/S0045782510001507`.
3. Wang J, Ye X. New finite element methods in computational fluid dynamics by H(div) elements. *SIAM Journal on Numerical Analysis* 2007; **45**(3):1269–1286, doi:10.1137/060649227. URL `http://dx.doi.org/10.1137/060649227`.
4. Cockburn B, Kanschat G, Schötzau D. An equal-order DG method for the incompressible Navier-Stokes equations. *Journal of Scientific Computing* 2009; **40**:188–210, doi:10.1007/s10915-008-9261-1. URL `http://dx.doi.org/10.1007/s10915-008-9261-1`.
5. Ayuso de Dios B, Brezzi F, Marini L, Xu J, Zikatanov L. A simple preconditioner for a discontinuous Galerkin method for the Stokes problem. *Journal of Scientific Computing* 2014; **58**(3):517–547, doi:10.1007/s10915-013-9758-0. URL `http://dx.doi.org/10.1007/s10915-013-9758-0`.
6. Benzi M, Golub GH, Liesen J. Numerical solution of saddle point problems. *Acta Numer.* 2005; **14**:1–137, doi:10.1017/S0962492904000212. URL `http://dx.doi.org/10.1017/S0962492904000212`.
7. Verfürth R. A combined conjugate gradient - multi-grid algorithm for the numerical solution of the Stokes problem. *IMA Journal of Numerical Analysis* 1984; **4**(4):441–455, doi:10.1093/imanum/4.4.441. URL `http://imajna.oxfordjournals.org/content/4/4/441.abstract`.
8. Rusten T, Winther R. A preconditioned iterative method for saddlepoint problems. *SIAM Journal on Matrix Analysis and Applications* 1992; **13**(3):887–904, doi:10.1137/0613054. URL `http://dx.doi.org/10.1137/0613054`.
9. Elman HC, Golub GH. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM Journal on Numerical Analysis* 1994; **31**(6):1645–1661, doi:10.1137/0731085. URL `http://dx.doi.org/10.1137/0731085`.
10. Silvester D, Wathen A. Fast iterative solution of stabilised Stokes systems part II: Using general block preconditioners. *SIAM Journal on Numerical Analysis* 1994; **31**(5):pp. 1352–1367, doi:10.1137/0731070. URL `http://www.jstor.org/stable/2158225`.
11. Zulehner W. A class of smoothers for saddle point problems. *Computing* 2000; **65**(3):227–246.
12. Olshanskii MA, Vassilevski YV. Pressure Schur complement preconditioners for the discrete Oseen problem. *SIAM Journal on Scientific Computing* 2007; **29**(6):2686–2704, doi:10.1137/070679776. URL `http://dx.doi.org/10.1137/070679776`.
13. Pestana J, Wathen AJ. Natural preconditioning and iterative methods for saddle point systems. *SIAM Review* 2015; **57**(1):71–91, doi:10.1137/130934921. URL `http://dx.doi.org/10.1137/130934921`.
14. ur Rehman M, Geenen T, Vuik C, Segal G, MacLachlan SP. On iterative methods for the incompressible Stokes problem. *International Journal for Numerical Methods in Fluids* 2011; **65**(10):1180–1200, doi:10.1002/fld.2235. URL `http://dx.doi.org/10.1002/fld.2235`.
15. Braess D, Sarazin R. An efficient smoother for the Stokes problem. *Applied Numerical Mathematics* 1997; **23**(1):3 – 19, doi:10.1016/S0168-9274(96)00059-1. URL `http://www.sciencedirect.com/science/article/pii/S0168927496000591`.
16. Vanka SP. Block-implicit multigrid calculation of two-dimensional recirculating flows. *Computer Methods in Applied Mechanics and Engineering* 1986; **59**(1):29 – 48, doi:10.1016/0045-7825(86)90022-8. URL `http://www.sciencedirect.com/science/article/pii/0045782586900228`.
17. Brandt A. *Multigrid techniques: 1984 guide with applications to fluid dynamics*. GMD–Studien Nr. 85, Gesellschaft für Mathematik und Datenverarbeitung: St. Augustin, 1984.
18. Xu J. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing* 1996; **56**(3):215–235, doi:10.1007/BF02238513. URL `http://dx.doi.org/10.1007/BF02238513`.
19. Hong Q, Kraus J, Xu J, Zikatanov L. A robust multigrid method for discontinuous Galerkin discretizations of Stokes and linear elasticity equations. *Numerische Mathematik* 2015; :1–27doi:10.1007/s00211-015-0712-y. URL `http://dx.doi.org/10.1007/s00211-015-0712-y`.
20. Arnold DN, Brezzi F, Cockburn B, Marini LD. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis* 2002; **39**(5):1749–1779, doi:10.1137/S0036142901384162. URL `http://dx.doi.org/10.1137/S0036142901384162`.
21. Brezzi F, Douglas J, Marini LD. Two families of mixed finite elements for second order elliptic problems. *Numerische Mathematik* 1985; **47**:217–235, doi:10.1007/BF01389710. URL `http://dx.doi.org/10.1007/BF01389710`.
22. Murphy MF, Golub GH, Wathen AJ. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.* 2000; **21**(6):1969–1972, doi:10.1137/S1064827599355153. URL `http://dx.doi.org/10.1137/S1064827599355153`.

23. Adler JH, Benson TR, Cyr EC, MacLachlan SP, Tuminaro RS. Monolithic multigrid methods for two-dimensional resistive magnetohydrodynamics. *In revision* 2015; .
24. Trottenberg U, Oosterlee CW, Schüller A. *Multigrid*. Academic Press: London, 2001.
25. MacLachlan SP, Oosterlee CW. Local Fourier analysis for multigrid with overlapping smoothers applied to systems of PDEs. *Numerical Linear Algebra with Applications* 2011; **18**(4):751–774, doi:10.1002/nla.762. URL `http://dx.doi.org/10.1002/nla.762`.
26. Larin M, Reusken A. A comparative study of efficient iterative solvers for generalized Stokes equations. *Numerical Linear Algebra with Applications* 2008; **15**(1):13 – 34, doi:10.1002/nla.561. URL `http://dx.doi.org/10.1002/nla.561`.
27. John V, Tobiska L. Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids* 2000; **33**(4):453–473, doi:10.1002/1097-0363(20000630)33:4⟨453::AID-FLD15⟩3.0.CO;2-0. URL `http://dx.doi.org/10.1002/1097-0363(20000630)33:4<453::AID-FLD15>3.0.CO;2-0`.
28. John V, Matthies G. Higher-order finite element discretizations in a benchmark problem for incompressible flows. *International Journal for Numerical Methods in Fluids* 2001; **37**(8):885–903, doi:10.1002/fld.195. URL `http://dx.doi.org/10.1002/fld.195`.
29. Saad Y. *Iterative methods for sparse linear systems*. Second edn., Society for Industrial and Applied Mathematics: Philadelphia, PA, 2003.
30. Heroux MA, Bartlett RA, Howle VE, Hoekstra RJ, Hu JJ, Kolda TG, Lehoucq RB, Long KR, Pawlowski RP, Phipps ET, *et al.*. An overview of the Trilinos project. *ACM Trans. Math. Softw.* 2005; **31**(3):397–423, doi:http://doi.acm.org/10.1145/1089014.1089021.
31. Logg A, Mardal KA, Wells GN, *et al.*. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012, doi:10.1007/978-3-642-23099-8.
32. Gmeiner B, Rüde U, Stengel H, Waluga C, Wohlmuth B. Performance and scalability of hierarchical hybrid multigrid solvers for stokes systems. *SIAM Journal on Scientific Computing* 2015; **37**(2):C143–C168, doi:10.1137/130941353. URL `http://dx.doi.org/10.1137/130941353`.