SIAM J. Sci. Comput.
Vol. 38, No. 1, pp. B1–B24

MONOLITHIC MULTIGRID METHODS FOR TWO-DIMENSIONAL RESISTIVE MAGNETOHYDRODYNAMICS[*]

JAMES H. ADLER[†], THOMAS R. BENSON[†], ERIC C. CYR[‡], SCOTT P. MACLACHLAN[§], AND RAYMOND S. TUMINARO[¶]

Abstract. Magnetohydrodynamic (MHD) representations are used to model a wide range of plasma physics applications and are characterized by a nonlinear system of partial differential equations that strongly couples a charged fluid with the evolution of electromagnetic fields. The resulting linear systems that arise from discretization and linearization of the nonlinear problem are generally difficult to solve. In this paper, we investigate multigrid preconditioners for this system. We consider two well-known multigrid relaxation methods for incompressible fluid dynamics: Braess–Sarazin relaxation and Vanka relaxation. We first extend these to the context of steady-state one-fluid viscoresistive MHD. Then we compare the two relaxation procedures within a multigrid-preconditioned GMRES method employed within Newton's method. To isolate the effects of the different relaxation methods, we use structured grids, inf-sup stable finite elements, and geometric interpolation. We present convergence and timing results for a two-dimensional, steady-state test problem.

Key words. monolithic multigrid, magnetohydrodynamics, Braess–Sarazin relaxation, Vanka relaxation

AMS subject classifications. 65F10, 65N55, 65N22, 76W05

DOI. 10.1137/151006135

1. Introduction.

Magnetohydrodynamics (MHD) models the flow of a charged fluid in the presence of electromagnetic fields. There are myriad formulations of the MHD model, depending on the domain and physical parameters in question. In this paper, we treat the ions and the electrons together as a single fluid. The resulting model couples the Navier–Stokes equations with Maxwell's equations, forming a nonlinear system of partial differential equations (PDEs). Depending on assumptions associated with the coupling between the electric field, current density, and Ohm's law, one can obtain a variety of different formulations such as ideal MHD, resistive MHD, and Hall MHD [21]. In this paper, we consider an incompressible viscoresistive formulation of the MHD system. Moreover, we focus on time-independent solutions, as our primary concern is with the linear solvers.

In order to treat this nonlinear system of PDEs, we discretize the nonlinear equations and then linearize using automatic differentiation and Newton's method [33]. In

[*]Submitted to the journal's Computational Methods in Science and Engineering section January 29, 2015; accepted for publication (in revised form) September 22, 2015; published electronically January 6, 2016. This work was partially supported by NSF grant DMS-1216972. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000. Part of this material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program.

http://www.siam.org/journals/sisc/38-1/100613.html

[†]Department of Mathematics, Tufts University, Medford, MA 02155 (james.adler@tufts.edu, thomas.benson@alumni.tufts.edu).

[‡]Sandia National Laboratories, P.O. Box 5800, MS 1320, Albuquerque, NM 87185 (eccyr@sandia.gov).

[§]Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's, NL, Canada A1C 5S7 (smaclachlan@mun.ca). This author's work was supported by an NSERC discovery grant.

[¶]Sandia National Laboratories, P.O. Box 969, MS 9159, Livermore, CA 94551 (rstumin@sandia.gov).

addition, we use a mixed finite-element discretization that results in a saddle-point linear system. Systems of this type arise in a variety of applications, including fluid dynamics [5]. There are two dominant families of solution algorithms for these types of systems, namely, block preconditioning [16, 17, 26, 45, 47] and monolithic multigrid [6, 30, 46].

Block preconditioners manipulate a segregated Jacobian operator so that the essential coupling in the problem is easily resolved using workhorse algorithms such as algebraic multigrid methods [37] on simpler problems [45]. In this way, these methods leverage the good parallel and mesh resolution scalability of existing preconditioner technology. Block preconditioning techniques have also been applied to MHD models. For the viscoresistive model of MHD, [15] uses an approximate block factorization scheme that decomposes the MHD system into two systems, incompressible Navier–Stokes and a velocity-magnetics coupling, and solves them independently, leveraging well-known solvers for incompressible Navier–Stokes. Additionally, in [12, 13, 28, 34], physics-based preconditioners are used to factor the matrix into systems that are amenable to multigrid methods.

In contrast, we will consider the latter family of monolithic multigrid methods, which have been used less commonly for MHD. In these approaches, relaxation techniques are developed for and applied directly to the fully coupled system. This idea is not new and dates back to the earliest treatments of multigrid for coupled systems [7, 8, 9]. For fluid dynamics, monolithic multigrid treatments of the Stokes equations can be found in [6, 19, 32, 39]. These techniques have been applied to the Navier–Stokes problem as well [27, 29, 46]. In the field of fluid-structure interaction, [20] presents a monolithic AMG for that coupled system. For MHD, monolithic nonlinear multigrid solvers have been used in the context of finite-difference discretizations [1]. Alternatively, first-order system least squares finite-element methods along with nested iteration and AMG to solve the resulting linear systems have been used on resistive MHD formulations [2, 3, 4]. Finally, in [41] a monolithic AMG preconditioner is developed for an equal-order stabilized viscoresistive MHD formulation. However, this methodology relies on the unknowns being collocated at the mesh nodes and thus is not applicable for the mixed discretization considered here.

In this paper, we will consider a mixed finite-element discretization of a viscoresistive formulation of the MHD model, described in section 2. To precondition the resulting linear systems, we employ a geometric multigrid preconditioner, detailed in section 3, in which we choose relaxation schemes that are extended from two well-known fluid dynamics techniques, namely, Vanka relaxation [46] and Braess–Sarazin relaxation [6]. Finally, we show in section 4 that these monolithic techniques lead to effective preconditioners for this system.

2. Steady-state viscoresistive MHD.

The MHD equations are a coupling of the Navier–Stokes equations with Maxwell's equations, where additional body forces come from electromagnetic effects [3, 31]. We pursue the one-fluid viscoresistive MHD system [21], where the dependent variables are the fluid velocity $\mathbf{u}$, the hydrodynamic pressure p, and the magnetic field $\mathbf{B}$. The equations are

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla\cdot(\mathbf{T}+\mathbf{T}_M) + \nabla p = \mathbf{0}, \tag{2.1}$$

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla\times(\mathbf{u}\times\mathbf{B}) + \nabla\times\left(\frac{1}{\mathrm{Re}_m}\nabla\times\mathbf{B}\right) = \mathbf{0} \tag{2.2}$$

$$\rho\nabla\cdot\mathbf{u} = 0, \tag{2.3}$$

$$\nabla\cdot\mathbf{B} = 0, \tag{2.4}$$

where the Newtonian and magnetic stress tensors are

$$\mathbf{T} = \frac{1}{\mathrm{Re}} \left[\nabla \mathbf{u} + \nabla \mathbf{u}^T\right] \tag{2.5}$$

and

$$\mathbf{T}_M = \mathbf{B} \otimes \mathbf{B} - \frac{1}{2} \|\mathbf{B}\|^2 \mathbf{I}, \tag{2.6}$$

respectively [22]. Additionally, we define the standard nondimensional Reynolds number, Re, and magnetic Reynolds number, Re_m,

$$\mathrm{Re} = \frac{\rho U L}{\nu}, \qquad \mathrm{Re}_m = \frac{\mu_0 U L}{\eta}, \tag{2.7}$$

for a characteristic velocity, U, and a characteristic length scale, L. The physical parameters, all assumed constant, are the fluid viscosity ν, the fluid density ρ, the magnetic permeability of free space μ_0, and the magnetic resistivity η. A third nondimensional constant arises in the test problem below, the Hartmann number, defined as

$$Ha = \sqrt{\mathrm{Re}\mathrm{Re}_m} = \frac{BL}{\sqrt{\eta\nu}}, \tag{2.8}$$

for a characteristic magnetic field magnitude B. The Reynolds number relates the fluid velocity to the fluid viscosity; the magnetic Reynolds number relates the fluid velocity to the magnetic resistivity; and the square of the Hartmann number relates the magnetic forces and the viscous forces [31]. In particular, higher values for the Hartmann number are associated with a stronger electromagnetic influence upon the flow, representing stronger coupling between the fluid and electromagnetic variables.

For this study, we focus on a two-dimensional geometry. We first note that we can write $\mathbf{B} = \nabla \times \mathbf{A}$ for a vector potential, $\mathbf{A}$, since it must satisfy $\nabla \cdot \mathbf{B} = 0$ (as such, (2.4) is automatically satisfied, and we no longer include it in the formulation). Since we want $\mathbf{B}$ to have nonzero components only in the plane, we demand that $\mathbf{A} = (0, 0, A_z)^T$. Using this simplification, (2.2) reduces to the following scalar equation:

$$\frac{\partial A_z}{\partial t} + \mathbf{u} \cdot \nabla A_z - \frac{1}{\mathrm{Re}_m} \nabla^2 A_z + E_z^0 = 0, \tag{2.9}$$

where E_z^0 is the z-component of the electrostatic part, $\mathbf{E}_{\mathrm{stat}}$, of the electric field, $\mathbf{E}$, that vanishes in the derivation of (2.2) because $\nabla \times \mathbf{E}_{\mathrm{stat}} = 0$. The magnetic stress tensor (2.6) can also be rewritten in terms of A_z:

$$\mathbf{T}_M = \begin{bmatrix} \frac{1}{2}\left[\left(\frac{\partial A_z}{\partial y}\right)^2 - \left(\frac{\partial A_z}{\partial x}\right)^2\right] & -\frac{\partial A_z}{\partial y}\frac{\partial A_z}{\partial x} \\ -\frac{\partial A_z}{\partial x}\frac{\partial A_z}{\partial y} & \frac{1}{2}\left[\left(\frac{\partial A_z}{\partial x}\right)^2 - \left(\frac{\partial A_z}{\partial y}\right)^2\right] \end{bmatrix}, \tag{2.10}$$

where $\hat{\mathbf{e}}_x$ and $\hat{\mathbf{e}}_y$ are the unit vectors in the x- and y-directions, respectively. For this paper, we consider this vector potential formulation, given by (2.1), (2.3), and (2.9), with viscous and magnetic stress tensors defined by (2.5) and (2.10), respectively.

2.1. Discretization and linearization. In this paper, we consider the time-steady case of (2.1), (2.3), and (2.9), with (2.5) and (2.10), and henceforth drop the time derivatives. To begin, note that the boundary conditions we consider are homogeneous Dirichlet for the fluid velocity, nonhomogenous Dirichlet for the magnetic vector potential, and none for the fluid pressure. Thus, we define finite-dimensional test spaces $\mathbf{V}^h \subset \mathbf{H}_0^1$, $V^h \subset H_0^1$, and $Q^h \subset L_0^2$. We write the discretized weak form of these equations:

$$\int_\Omega \left\{ \left[(\mathbf{u}^h \cdot \nabla)\mathbf{u}^h\right] \cdot \mathbf{v}^h + (\mathbf{T}^h + \mathbf{T}_M^h) : \nabla \mathbf{v}^h + \nabla p^h \cdot \mathbf{v}^h \right\} \, \mathrm{d}\mathbf{x} = \mathbf{0} \quad \forall \mathbf{v}^h \in \mathbf{V}^h, \tag{2.11}$$

$$\int_\Omega \left[\left(\mathbf{u}^h \cdot \nabla A_z^h + E_z^0\right) \psi^h + \frac{1}{Re_m} \nabla A_z^h \cdot \nabla \psi^h \right] \mathrm{d}\mathbf{x} = 0 \quad \forall \psi^h \in V^h, \tag{2.12}$$

$$\int_\Omega q^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}\mathbf{x} = 0 \quad \forall q^h \in Q^h, \tag{2.13}$$

where $\mathbf{T}^h = \mathbf{T}(\mathbf{u}^h)$ and $\mathbf{T}_M^h = \mathbf{T}_M(A_z^h)$.

The discrete variational form, (2.11)–(2.13), is nonlinear, and thus automatic differentiation is used in the context of Newton's method to linearize the system [33]. After this process, linear systems of the following block form are solved:

$$\mathcal{A}x = \begin{bmatrix} F & Z & B \\ Y & D & 0 \\ B^T & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{\mathbf{u}} \\ x_a \\ x_p \end{bmatrix} = \begin{bmatrix} f_{\mathbf{u}} \\ f_a \\ f_p \end{bmatrix}, \tag{2.14}$$

where $x_{\mathbf{u}}$, x_a, and x_p are the discrete Newton corrections for $\mathbf{u}$, A_z, and p, respectively, and $f_{\mathbf{u}}$, f_a, and f_p are the corresponding blocks of the nonlinear residual.

As with the Navier–Stokes equations, well-posedness of the discrete system requires that the discretization of the velocity and pressure unknowns satisfy an "inf-sup" condition. In this case, the fluid velocity and pressure variables, $x_{\mathbf{u}}$ and x_p, are discretized using Taylor–Hood ($\mathbf{Q}_2 - Q_1$) elements that are well known to be stable [10, 18]. The basis functions associated with these elements are biquadratic for each component of the velocity field ($\mathbf{V}^h = \mathbf{Q}_2$) and bilinear for the pressure ($Q^h = Q_1$). For the vector potential variable, x_a, a discretization using (scalar) Q_2 elements is used as the test space, V^h. This affords the advantage that the degrees of freedom will be collocated with the velocity field degrees of freedom.

While similar discretizations have been considered in the literature before [15, 35, 41], we note that proofs of inf-sup stability for (2.11)–(2.13) remain open, both in the continuum and at the discrete level. For the variational formulation of the full **B**-field stationary, incompressible MHD system, it is known that augmenting an inf-sup stable finite-element pair for the velocity-pressure degrees of freedom with an appropriate finite-element space for the magnetic degrees of freedom yields a system that is stable overall [23, 40], under suitable small-data or low-Reynolds number assumptions. Following these approaches would likely yield a similar result for this formulation, and we will consider this in future work.

3. Monolithic multigrid.

System (2.14) is of saddle-point type [5]. Such systems arise in a variety of applications, including fluid dynamics. This paper develops a monolithic multigrid preconditioner for this system using geometric coarsening. The focus of this section, and this paper, is the development of effective relaxation schemes.

Geometric multigrid methods effectively use complementary processes of relaxation and coarse-grid correction to effectively damp all components of the error in a linear system [11, 44]. In particular, a relaxation technique is employed to quickly damp the oscillatory components of the error. Subsequently, a coarse-grid correction scheme, in which a projected problem is solved on a coarser grid and the solution is interpolated as an error correction on the fine grid, is used to damp the smooth components of the error. This process may be applied recursively to multigrid hierarchies consisting of several levels to achieve an optimal algorithm for solving a linear system.

Here, we investigate the choice of the relaxation scheme, leaving the rest of the multigrid method fixed. We use geometric interpolation and restriction operators on regular grids with coarsening by a factor of two in each direction. These grid transfer operators are applied componentwise (that is, the velocity space is coarsened independently of the pressure space, etc.). In particular, we define the interpolation operators to be the blocked versions of the standard $\mathbf{Q}_2 - Q_2 - Q_1$ finite-element interpolation operators. That is,

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

where $P_{\mathbf{u}}$ is a vector-biquadratic interpolation operator ($\mathbf{Q}_2$), P_a is a scalar-biquadratic interpolation operator (Q_2), and P_p is a bilinear interpolation operator (Q_1). Furthermore, we use the Galerkin coarse grid operator $\mathcal{A}_c = P^T \mathcal{A} P$. Finally, we use one prerelaxation step (i.e., before restricting to the coarse grid) and one postrelaxation step (i.e., after interpolating the correction to the fine grid). With these components fixed, we extend and compare two relaxation techniques from the fluid dynamics literature. We discuss the extension of the Vanka scheme [46] to system (2.14) in section 3.1, and then we describe the extension of the Braess–Sarazin scheme [6] in section 3.2.

Remark 1. We use the term “monolithic” multigrid here to describe a multigrid method in which all components of the solution (here, $\mathbf{u}$, A_z, and p) are simultaneously transferred between grids using a block-structured interpolation operator, such as the one above. This is in contrast to “block-preconditioning” approaches, where simpler multigrid cycles may be used for some or all components of the solution, but coupling between components only occurs on the scale of the original discretization.

3.1. Vanka relaxation. Vanka relaxation was originally developed for use with multigrid solvers for the Navier–Stokes equations in primitive variables. A general description of the Vanka scheme for incompressible fluids can be found in [27, 29, 46]. The method describes an overlapping block-Gauss–Seidel iteration [38] (or an overlapping multiplicative Schwarz method) in which the choice of blocks (subdomains) is motivated by the underlying saddle-point problem, in such a way as to ensure that the submatrices are also saddle-point matrices. The original algorithm was discussed in the context of a staggered-mesh (marker and cell, MAC) finite-difference scheme, which was analyzed using local mode analysis in [42]. Vanka methods for finite-element discretizations of the Stokes equations have been analyzed in that setting in [30]. Here, we extend the Vanka scheme to the finite-element discretization of this MHD formulation.

Define the sets of degrees of freedom (DOFs) to be $\mathcal{S}_{\mathbf{u}} = \{u_1, \ldots, u_{n_u}\}$, $\mathcal{S}_a = \{a_1, \ldots, a_{n_a}\}$, and $\mathcal{S}_p = \{p_1, \ldots, p_{n_p}\}$, where n_u, n_a, and n_p are the numbers of $\mathbf{u}$, A_z, and p DOFs, respectively, in the system. Let $\mathcal{S} = \mathcal{S}_{\mathbf{u}} \cup \mathcal{S}_a \cup \mathcal{S}_p$ be the set of all DOFs in the system. The N Vanka blocks, $S_\ell \subset S$, $\ell = 1, \ldots, N$, are chosen such that each block contains some elements of $\mathcal{S}_{\mathbf{u}}$, some elements of $\mathcal{S}_a$, and some elements of

$\mathcal{S}_p$. Moreover, the Vanka scheme is an *overlapping* block-Gauss–Seidel method, and thus DOFs are allowed to appear in multiple blocks as long as $\cup_\ell \mathcal{S}_\ell = \mathcal{S}$ (i.e., each DOF appears in at least one block).

Remark 2. Note that the members of $\mathcal{S}_{\mathbf{u}}$, $\mathcal{S}_a$, and $\mathcal{S}_p$ are not the values in the vectors $x_{\mathbf{u}}$, x_a, and x_p, though there is an obvious connection between the two. Specifically, the value corresponding to the DOF u_i is the ith entry in the vector $x_{\mathbf{u}}$. A similar relation holds for $\mathcal{S}_a$ and x_a, as well as $\mathcal{S}_p$ and x_p.

In fluid-dynamics applications, the standard approach to decomposing $\mathcal{S}$ into the subsets, $\mathcal{S}_\ell$, is to "seed" the choice of the Vanka blocks by the incompressibility constraint, or (equivalently) by the pressure degrees of freedom. Algebraically, this means considering each row, ℓ, of the matrix B^T in (2.14), and defining $\mathcal{S}_\ell$ to be the DOFs with nonzero entries in that row, as well as the DOF, $p_\ell \in \mathcal{S}_p$, to which that row corresponds. Geometrically, this is the same as considering pressure node ℓ and defining $\mathcal{S}_\ell$ to be all of the DOFs contained in the stencil surrounding it, with the exception of the other pressure DOFs (so that only a single pressure DOF appears in each Vanka block). Thus, the incompressibility constraint is enforced for the DOFs in the Vanka block at each step of the Gauss–Seidel iteration.

Since, in the block form given in (2.14), there is no coupling between the magnetics DOFs and the incompressibility constraint (i.e., there are no length-one algebraic graph connections from nodes associated with DOFs from the matrix B^T to those associated with the matrix Z), proceeding in this way does not give a proper decomposition of $\mathcal{S}$, since no decomposition of $\mathcal{S}_a$ is done. Since we are working with a $\mathbf{Q}_2 - Q_2 - Q_1$ finite-element method, however, we have vector potential DOFs x_a that are collocated with the velocity DOFs ($x_{\mathbf{u}}$). Thus, we extend the classical definition of the Vanka blocks by augmenting them with the magnetics DOFs that are collocated with the velocity DOFs that are associated with the local incompressibility constraint equation particular to that block. This is the only dependence on the discretization in this method, and it could easily be generalized to other discretizations. Alternatively, if one considers the complete symbolic finite-element stencil (i.e., including entries that numerically but not symbolically evaluate to zero), a Vanka block $\mathcal{S}_\ell$ may be thought of as those DOFs present in a stencil centered on pressure node ℓ, except for the other pressure DOFs. For the $\mathbf{Q}_2 - Q_2 - Q_1$ discretization in two dimensions, the Vanka blocks will have a maximal size of 76 DOFs per block (25 each for the x- and y-components of velocity, 25 for the vector potential, and 1 for the pressure).

The Vanka blocks are collections of DOFs that we use to define the remainder of the method. For each Vanka block $\mathcal{S}_\ell$, the Vanka step computes updates to the global solution x as

$$x \leftarrow x + V_\ell \left(\omega M_{\ell\ell}^{-1}\right) V_\ell^T \left(b - \mathcal{A}x\right). \tag{3.1}$$

Here, V_ℓ^T is a "restriction operator" that restricts global vectors to local vectors, containing only entries corresponding to the DOFs in Vanka block $\mathcal{S}_\ell$. Likewise, V_ℓ is the corresponding "prolongation operator" that returns local vectors to global vectors [38]. We also require the use of an underrelaxation parameter here, termed ω. Finally, $M_{\ell\ell} = V_\ell^T M V_\ell$, where M is some appropriate preconditioner for $\mathcal{A}$. Note that $M_{\ell\ell}$ will maintain the saddle-point structure of $\mathcal{A}$. In our case, it is the choice of M that differentiates different Vanka methods.

Remark 3. For our purposes, we consider ω in (3.1) to be a scalar. It is possible, however, to consider a diagonal matrix scaling instead, with

$$\omega = \begin{bmatrix} \omega_{\mathbf{u}} I_{\mathbf{u}} & & \\ & \omega_a I_a & \\ & & \omega_p I_p \end{bmatrix},$$

where $I_{\mathbf{u}}$, I_a, and I_p are identity matrices of appropriate size for the velocity, magnetics, and pressure DOFs in each Vanka block. As we show in section 4.2, scaling each component independently was not necessary to achieve good performance in the case of the test problem that we consider.

Next, we define three different Vanka methods: "Full" Vanka, "Diagonal" Vanka, and "Economy" Vanka. These correspond to three choices of M. To assist in describing these methods, we make the following definitions, relative to the system (2.14):

$$\hat{F} = \begin{bmatrix} F & Z \\ Y & D \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}, \quad x_{\hat{\mathbf{u}}} = \begin{bmatrix} x_{\mathbf{u}} \\ x_a \end{bmatrix}, \quad f_{\hat{\mathbf{u}}} = \begin{bmatrix} f_{\mathbf{u}} \\ f_a \end{bmatrix}. \tag{3.2}$$

Then system (2.14) is rewritten

$$\mathcal{A}x = \begin{bmatrix} \hat{F} & \hat{B} \\ \hat{B}^T & 0 \end{bmatrix} \begin{bmatrix} x_{\hat{\mathbf{u}}} \\ x_p \end{bmatrix} = \begin{bmatrix} f_{\hat{\mathbf{u}}} \\ f_p \end{bmatrix}. \tag{3.3}$$

Thus, (2.14) is in exactly the same block form as the linear systems associated with the incompressible Stokes or Navier–Stokes equations [6, 27, 29, 46].

The Full Vanka method is defined by taking $M_{\text{full}} = \mathcal{A}$:

$$M_{\text{full}} = \begin{bmatrix} \hat{F} & \hat{B} \\ \hat{B}^T & 0 \end{bmatrix}. \tag{3.4}$$

In this case, the Vanka submatrix $M_{\ell\ell}$ is effectively dense. Even though these submatrices are not large (maximally 76×76 in the case of a $\mathbf{Q}_2 - Q_2 - Q_1$ discretization), solving systems with them is computationally expensive, especially as the number of Vanka blocks grows large. To ameliorate this computational burden, we consider alternative choices for M that result in submatrices that can be more cheaply inverted at each step.

First, we define the Diagonal Vanka method [27, 29], which was also originally used for incompressible Navier–Stokes. We extend it here to MHD simply by using

$$M = M_{\text{diag}} = \begin{bmatrix} \text{diag}(\hat{F}) & \hat{B} \\ \hat{B}^T & 0 \end{bmatrix}, \tag{3.5}$$

where

$$\text{diag}(\hat{F}) = \begin{bmatrix} \text{diag}(F) & \\ & \text{diag}(D) \end{bmatrix}.$$

The fluid-pressure coupling is exactly the same as the Diagonal Vanka method for Navier–Stokes, and we have added a diagonal component of the magnetics variables as well. The choice of $M = M_{\text{diag}}$ results in Vanka submatrices, $M_{\ell\ell}$ that are much less dense than those resulting from the choice $M = M_{\text{full}}$. Moreover, solving the resulting linear systems requires only three vector operations (and no explicit matrix inversion or matrix-vector products) on vectors of size one less than the dimension of $M_{\ell\ell}$: a componentwise product (for the inversion of the diagonal block), a scalar multiplication, and a dot product.

We will see in section 4.3.1 that solving systems with M_{diag} is substantially cheaper and faster *per iteration* than with M_{full}, but the resulting preconditioner deteriorates as the fluid velocity and magnetics coupling, represented by Y and Z, grow stronger. This is because this approach essentially decouples the magnetics from the velocity-pressure unknowns when updating the ℓth block. This can be seen by considering the following block factorization of (3.5):

$$\begin{bmatrix} \operatorname{diag}(\hat{F}) & \hat{B} \\ \hat{B}^T & 0 \end{bmatrix} = \begin{bmatrix} \operatorname{diag}(\hat{F}) & 0 \\ \hat{B}^T & \hat{S} \end{bmatrix} \begin{bmatrix} I & \operatorname{diag}(\hat{F})^{-1}\hat{B} \\ 0 & I \end{bmatrix}, \tag{3.6}$$

where $\hat{S} = -\hat{B}^T \operatorname{diag}(\hat{F})^{-1}\hat{B}$. Expanding this expression, we have

$$\hat{S} = -\begin{bmatrix} B^T & 0 \end{bmatrix} \begin{bmatrix} \operatorname{diag}(F)^{-1} & 0 \\ 0 & \operatorname{diag}(D)^{-1} \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix} = -B^T \operatorname{diag}(F)^{-1} B. \tag{3.7}$$

That is, a solve with (3.6) will decouple the magnetics from the velocity-pressure update on block ℓ, since solves with $\hat{S}$ will only receive contributions from F, the matrix representing velocity-velocity coupling on Vanka block ℓ.

As an alternative that preserves some physics coupling and reduces the density of the submatrices, we propose an Economy Vanka method. For this approach, we define

$$M = M_{\text{econ}} = \begin{bmatrix} \operatorname{blkDiag}(\hat{F}) & \hat{B} \\ \hat{B}^T & 0 \end{bmatrix}, \tag{3.8}$$

where blkDiag($\hat{F}$) is a special block-diagonal submatrix of $\hat{F}$. To form blkDiag($\hat{F}$) in the case of the $\mathbf{Q}_2 - Q_2 - Q_1$ discretization, we remove the off-node connections in a given row of $\hat{F}$, or, equivalently, we keep only the entries in a row corresponding to velocity and magnetics DOFs collocated with the DOF corresponding to the diagonal entry of that row. In this case, the resultant matrix, blkDiag($\hat{F}$), will have three entries remaining in each row: one x-component of velocity, one y-component of velocity, and one vector potential. The Vanka submatrices arising from the choice of $M = M_{\text{econ}}$ will inherit this increased sparsity while still preserving some of the velocity-magnetics coupling. Thus, these submatrices are cheaper to apply than those of the Full approach. By precomputing the (dense) LU-factors of the submatrices corresponding to these diagonal blocks, we can cheaply apply the inverses of the Economy Vanka submatrices by a series of solves with these factors, plus one scalar multiplication and a dot product, both with vectors of size one less than the dimension of $M_{\ell\ell}$.

3.2. Braess–Sarazin relaxation. Braess–Sarazin-type algorithms were originally developed as a relaxation scheme for the Stokes equations [6, 29]. They have also been studied in the context of the incompressible Navier–Stokes equations [27]. Whereas the Vanka methods require solving several small, local saddle-point systems in a block Gauss–Seidel fashion, the Braess–Sarazin methods require solving a greatly simplified but global saddle-point system.

Using (3.3), we proceed directly as in the original algorithm. We must solve systems of the form

$$\begin{bmatrix} \alpha C & \hat{B} \\ \hat{B}^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha C & 0 \\ \hat{B}^T & S \end{bmatrix} \begin{bmatrix} I & \frac{1}{\alpha}C^{-1}\hat{B} \\ 0 & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} d \\ g \end{bmatrix}, \tag{3.9}$$

where C is some appropriate preconditioner for $\hat{F}$, the inverse of which is easy to apply; α is a chosen relaxation parameter; and $S = -\frac{1}{\alpha}\hat{B}^T C^{-1}\hat{B}$ is the Schur complement. Solutions of (3.9) are computed by

$$Sy = g - \frac{1}{\alpha}\hat{B}^T C^{-1} d, \tag{3.10}$$

$$x = \frac{1}{\alpha} C^{-1}(d - \hat{B}y). \tag{3.11}$$

In practice, (3.10) is not solved exactly; an approximate solve is sufficient. For this study, we explicitly form the matrix S for a given choice of C, and we consider approximately solving (3.10) using a single sweep of either symmetric (point) Gauss–Seidel (SGS) or weighted (point) Jacobi. SGS does not require another parameter to be chosen, and we will see that it tends to be more effective in terms of linear iteration counts. Jacobi requires another parameter choice but is a perfectly parallelizable method. Solves with more sweeps were also considered, but we observed only slight improvement in the average linear iteration count and almost no improvement in total time to solution when using multiple sweeps of SGS.

The ideal Braess–Sarazin update then takes the form

$$\begin{bmatrix} x_{\hat{\mathbf{u}}} \\ x_p \end{bmatrix}^{(k+1)} = \begin{bmatrix} x_{\hat{\mathbf{u}}} \\ x_p \end{bmatrix}^{(k)} + \begin{bmatrix} \alpha C & \hat{B} \\ \hat{B}^T & 0 \end{bmatrix}^{-1} \left(\begin{bmatrix} f_{\hat{\mathbf{u}}} \\ f_p \end{bmatrix} - \begin{bmatrix} \hat{F} & \hat{B} \\ \hat{B}^T & 0 \end{bmatrix} \begin{bmatrix} x_{\hat{\mathbf{u}}} \\ x_p \end{bmatrix}^{(k)} \right),$$

where $\hat{F}$, $\hat{B}$, $x_{\hat{\mathbf{u}}}$, and $f_{\hat{\mathbf{u}}}$ are as in (3.2), and C is some appropriate preconditioner for $\hat{F}$. The linear system is approximately solved using (3.10)–(3.11) using SGS or weighted Jacobi to approximately solve the Schur complement system, as described above.

We note that (3.9) could also be seen as the starting point for the well-known family of block-factorization preconditioners [5, 16, 18]; because we use it only to define the smoother on each level of the multigrid hierarchy, we are able to choose C to be based on very simple approximations to $\hat{F}$ and still have a very effective preconditioner. For this study, we consider two possible choices for C. The first possibility is simply

$$C_{\text{diag}} = \text{diag}(\hat{F}) = \begin{bmatrix} \text{diag}(F) & \\ & \text{diag}(D) \end{bmatrix}.$$

This choice, referred to hereafter as Diagonal Braess–Sarazin, affords the advantage that C_{diag}^{-1} is nearly trivial to compute and apply.

Applying the same reasoning as in section 3.1 with (3.6)–(3.7), we see that Diagonal Braess–Sarazin results in a Schur complement of the form $S = -B^T \,\text{diag}(F)^{-1}B$. In this case, Diagonal Braess–Sarazin essentially decouples as the original Braess–Sarazin on the fluid system (velocity and pressure) and weighted Jacobi on the magnetics degrees of freedom—there is no coupling of the magnetics with the fluids. We expect, then, that this decoupling leads to the diminishing performance of the solver that is observed in section 4.2 when the strength of the coupling between the fluid velocity and magnetic potential is increased.

Thus, in order to achieve a more robust method, we consider the same block-diagonal approximation to $\hat{F}$ that we considered when forming M_{econ}, (3.8), for Economy Vanka above. That is, we use $C_{\text{blkdiag}} = \text{blkDiag}(\hat{F})$. This choice, referred to hereafter as Block–Diagonal Braess–Sarazin, results in a C_{blkdiag}^{-1} that has only 3 entries per row and is thus also easy to apply. Moreover, it recognizes the coupling between the magnetic effects and the fluid effects. We see that preserving this coupling numerically in the relaxation scheme is important for achieving convergence in physical regimes in which the magnetic forces couple strongly to those of the fluid.

4. Numerical experiments.

For numerical experiments, we study a modified Hartmann flow vector potential system [31, 41]. This is an appealing test problem for this study because there is a known analytical solution against which we will verify the numerical results. We fix the stopping criterion for the nonlinear iteration to be that the norm of the nonlinear residual has been reduced below 10^{-8}. A relative residual reduction of 10^{-5} is the linear solver stopping criteria. Within the multigrid method, the number of levels is chosen so that the coarsest grid corresponds to an 8×8 mesh (3556 DOFs). Finally, we use geometric interpolation operators and Galerkin coarse-grid operators, as described above, as well as one prerelaxation step and one postrelaxation step (a V(1,1) cycle). The only component to the solver that varies is the choice of the relaxation technique: Full Vanka, Economy Vanka, Diagonal Vanka, Block-Diagonal Braess–Sarazin, and Diagonal Braess–Sarazin.

We have implemented these solvers using the Trilinos framework [24] within the Drekar multiphysics application [43]. Drekar manages the simulation interface—the nonlinear iterations, the discretization, and the linear solver. Multigrid preconditioners are implemented within the MueLu package [36], and the Teko package [14], designed for block preconditioning, is used for the Braess–Sarazin methods. Finally, the Vanka methods are implemented within the Ifpack2 package [24], as an interface for block Gauss–Seidel methods, which already existed there—requiring only that the blocks and the Vanka submatrix form be specified. 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.

4.1. Test problem and numerical verification.

The modified Hartmann flow problem is a steady-state problem that we consider in two dimensions, posed over a square domain $\Omega = [-L, L]^2$. This models a section of a duct or channel through which a fluid is flowing and is subjected to a transverse magnetic field $\mathbf{B}_0 = (0, B_0, 0)$, applied perpendicularly to the direction of the flow. In this test problem, the fluid flow is driven in the x-direction by an applied pressure gradient $\frac{\partial p}{\partial x} = -G_0$, and thus the magnetic field $\mathbf{B}_0$ is nonzero only in the y-direction. Also, we assume that the channel has insulating walls. Thus, as stated above, we have Dirichlet boundary conditions for the fluid velocity and the magnetic vector potential on all four walls.

This configuration results in velocity having only a single nonzero component, u_x, and the magnetic field having two, B_x and B_y. It can be shown that $B_y = B_0$ [31], the applied magnetic field. The analytical solution to this problem is then $\mathbf{u} = (u_x, 0, 0)$ and $\mathbf{B} = (B_x, B_0, 0)$, where

$$u_x = \frac{\eta\rho G_0 Ha}{B_0^2}\left(\frac{\cosh(Ha) - \cosh(yHa/L)}{\sinh(Ha)}\right), \tag{4.1}$$

$$B_x = \frac{B_0 \mathrm{Re}_m}{Ha}\left(\frac{\sinh(yHa/L) - y\sinh(Ha)/L}{\cosh(Ha)}\right). \tag{4.2}$$

From this expression, and recalling that $\mathbf{B} = \nabla \times \mathbf{A}$, we have that A_z must be

$$A_z = -B_0 x + \frac{\mu_0 \rho G_0 L^2}{B_0}\left(\frac{\cosh(yHa/L)}{\sinh(Ha)Ha} - \frac{y^2}{2L^2}\right). \tag{4.3}$$

Due to the insulating walls of the channel, the electrostatic field component is constant and is given by

$$E_z^0 = \frac{\eta\rho G_0}{B_0}\left(1 - \frac{Ha}{\tanh(Ha)}\right).$$

Recall the definitions of the Reynolds number, Re, the magnetic Reynolds number, Re_m, and the Hartmann number, Ha, given in (2.7) and (2.8). In the case of the Hartmann problem, we have U proportional to the maximum x-direction velocity:

$$U = \frac{\eta\rho G_0}{B_0^2}\frac{Ha}{\tanh(Ha)}.$$

For the Hartmann number, the characteristic magnetic field magnitude is $B = B_0$. For the numerical experiments, we took $L = 1$, giving the computational domain $\Omega = [-1, 1]^2$. Furthermore, we took $\rho = \nu = \eta = \mu_0 = 1$, and $G_0 = 50$. We selected different values for B_0 to produce the desired Hartmann numbers.

In Figure 1, we show a numerical verification study. We observe that, for a representative sample of the solvers, we are in excellent agreement with the analytical solution. The linear solver used to generate the numerical results for Figure 1 was multigrid-preconditioned GMRES with Block-Diagonal Braess–Sarazin relaxation with $\alpha = 1.0$ using SGS for the approximate Schur complement solve on a problem for which the size of the fine grid was 512×512. It should be noted that the results were indistinguishable for the other solvers with their optimal parameters. Table 1 shows the number of rows and the number of nonzero entries in the fine-grid matrix for a number of grid sizes, as well the number of Newton steps and total GMRES iterations required to resolve the simulation. We also show the discrete L_2-norm of the error in the u_x and the A_z components of the final solution. We have spatial convergence of the nonlinear problem in the L_2-norm in $\mathscr{O}(h^4)$. Note that this is for the full solution. For quadratic elements applied to the linearizations, we expect $\mathscr{O}(h^3)$ convergence rates for the update in Newton's method and hence get one more order for the full solution. Furthermore, with $Ha = 20$, the number of Newton steps and GMRES iterations remains effectively constant as the resolution is increased. At $Ha = 80$ on a grid with 128×128 elements, the linear solver requires more iterations to resolve the solution, but the nonlinear solver is unaffected. At higher resolution, however, this is not an issue and the iteration counts appear closer to those in the $Ha = 20$ case, where the influence of the magnetic field is less strong.

4.2. Parameter studies. Each of the relaxation techniques described in section 3 has at least one parameter that must be chosen: ω for Vanka and α for Braess–Sarazin. With the Braess–Sarazin method, we consider different techniques for solving (3.11), namely, classical weighted Jacobi and SGS iterations. The former yields the advantage that it is perfectly parallelizable, but requires choosing an underrelaxation parameter ω_J to get convergence. The latter gives slightly better serial convergence (by about two iterations per Newton step on average) and requires no additional parameters.

4.2.1. Vanka. First, we consider Vanka relaxation. We investigate two physical regimes: one with Hartmann number $Ha = 20$ and the other with $Ha = 80$. We vary the Vanka parameter ω from 0.1 to 1.0 in steps of 0.1. Figure 2 shows the average number of linear iterations per Newton step required to reduce the relative linear residual below 10^{-5}. The absence of a data point implies that the linear solver

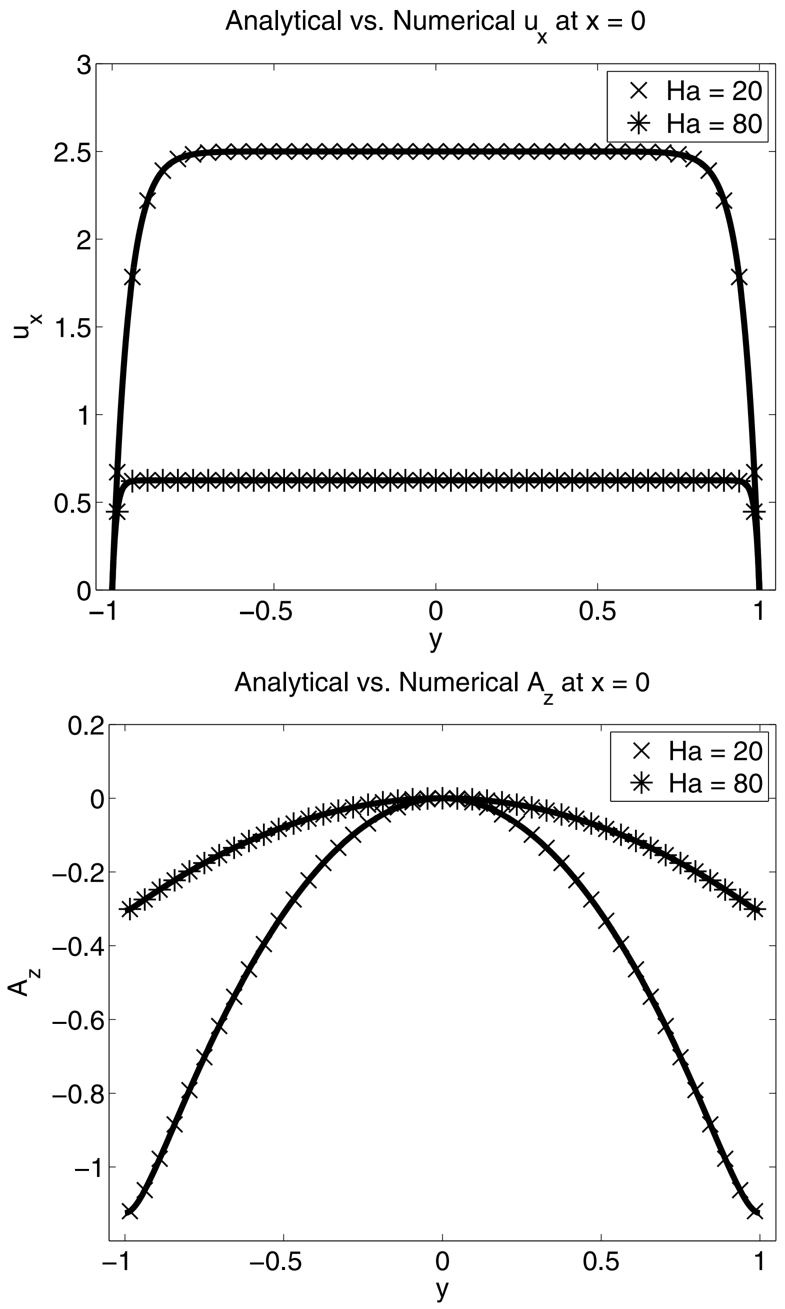

FIG. 1. *Numerical verification of the Block-Diagonal Braess–Sarazin solver for the Hartmann test problem. On the top, the true versus the computed velocity (u_x) solutions; on the bottom, the true versus the computed vector potential (A_z) solutions. The solid lines show the analytical solution, and the $\times$ and $\ast$ symbols indicate the numerical solution sampled every 12 elements along the line $x = 0$. Recall that the solutions for u_x should be independent of x, and the solutions for A_z should only vary linearly with x.*

Table 1

Data for typical simulations using the Braess–Sarazin method with $\alpha = 1.0$ and SGS as the approximate Schur complement solver. "N" is the row dimension of the system matrix. "nnz" is the number of nonzeros in the matrix, including zeros that arise numerically but not symbolically in the finite-element discretization. "Newton" gives the number of Newton steps to reach an absolute nonlinear residual tolerance of 10^{-8}. "GMRES" indicates the total number of GMRES iterations required throughout the simulation. $\|e_{u_x}\|_2$ and $\|e_a\|_2$ are the discrete L_2-norms of the error in the u_x and A_z components of the numerical solution, respectively.

	Grid	N	nnz	Newton	GMRES	$\|e_{u_x}\|_2$	$\|e_a\|_2$
$Ha = 20$	128^2	214,778	12,069,136	5	53	3.36e-06	2.48e-06
	256^2	855,556	48,222,736	5	52	2.10e-07	1.57e-07
	512^2	3,415,044	192,783,376	5	52	1.31e-08	9.85e-09
	1024^2	13,645,828	770,918,416	5	52	8.19e-10	6.16e-10
$Ha = 80$	128^2	214,778	12,069,136	5	80	1.19e-04	1.35e-04
	256^2	855,556	48,222,736	5	59	7.32e-06	1.02e-05
	512^2	3,415,044	192,783,376	4	43	4.53e-07	6.71e-07
	1024^2	13,645,828	770,918,416	4	41	2.82e-08	4.20e-08

failed to converge to the desired tolerance within fifty iterations for at least one of the Newton steps.

The optimal choice for Full Vanka is $\omega = 0.6$, and with this choice of ω, the linear systems converge in an average of six iterations for the $Ha = 20$ problem. This convergence is independent of the fine-grid size. For the $Ha = 80$ problem, we see convergence in about seven iterations—slightly less for the 512×512 problem and slightly more for the 128×128 problem. Overall, the optimal parameter choice for the method is quite insensitive to the grid size and Hartmann number.

For Diagonal Vanka, the optimal parameter has now shifted to $\omega = 0.5$ for both Hartmann numbers. For the $Ha = 20$ test problem, this method required an average of 10–11 GMRES iterations per linear solve. For $Ha = 80$, however, the performance of the Diagonal Vanka method is not robust to large Hartmann numbers. The optimal parameter shifts to $\omega = 0.3$ and the average number of GMRES iterations per linear solve has increased to 20–22, now requiring twice as many as for the smaller Hartmann numbers. This is not surprising—as we increase the Hartmann number, the magnetic coupling to the fluid flow becomes stronger. This coupling is largely ignored by relaxation when we take the diagonal of $\hat{\mathcal{A}}_{\ell\ell} = \left[\begin{smallmatrix} F & Z \\ Y & D \end{smallmatrix}\right]_{\ell\ell}$. That is, the contributions of $Z_{\ell\ell}$ and $Y_{\ell\ell}$ are ignored by relaxation.

As the proposed compromise between the robustness of Full Vanka and the simplicity of Diagonal Vanka, the results for Economy Vanka are shown in the center row of Figure 2. For the low Hartmann number, $Ha = 20$, this method performs similarly to the Diagonal Vanka method, having an optimal parameter of $\omega = 0.5$ and requiring an average of 10–11 GMRES iterations per linear solve. However, for the $Ha = 80$ test problem, it remains robust, requiring 11–13 GMRES iterations with an optimal parameter of $\omega = 0.5$. Thus, Economy Vanka takes a few more iterations than Full Vanka, but it is also very robust across Hartmann parameter values, unlike Diagonal Vanka.

As a final observation of this parameter study, we note that the Full Vanka method offers more parameter choice flexibility in that it gives similar convergence results for ω values between 0.5 and 0.7. However, with both the Economy and Diagonal Vanka methods, we see that varying the relaxation parameter by 0.1 in either direction from

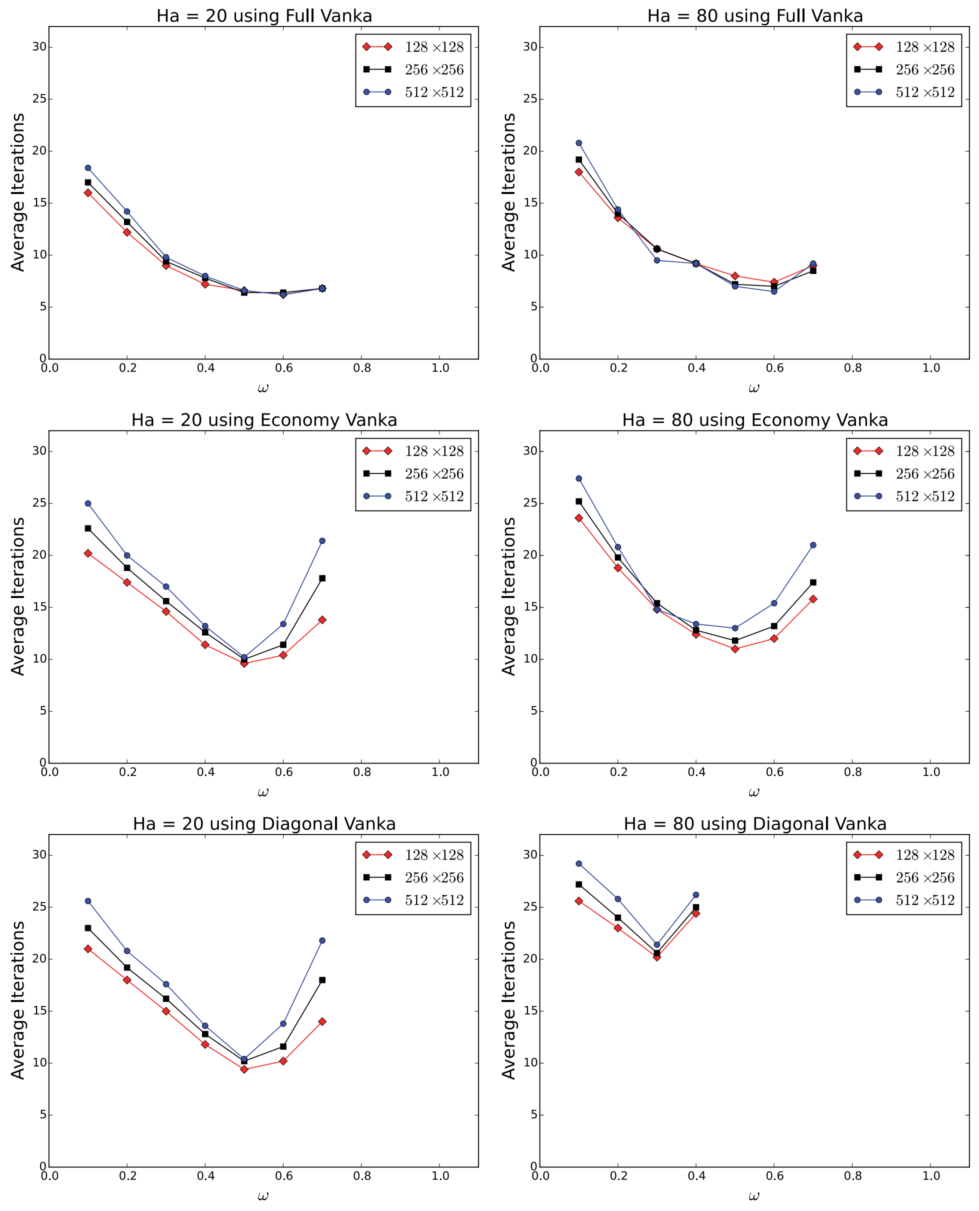

Fig. 2. *The average number of GMRES iterations using the Vanka method required to solve the linear systems to a relative tolerance of* 10^{-5} *over all Newton steps for various choices of* ω *with the listed fine-grid size. The top row shows Full Vanka; the middle row shows Economy Vanka; and the bottom row shows Diagonal Vanka. The left column shows* $Ha = 20$*, and the right column shows* $Ha = 80$.

the optimal $\omega = 0.5$ incurs a substantial cost of 3–6 GMRES iterations per linear solve for most problems (for $Ha = 80$, it appears that choosing $\omega = 0.4$ for Economy Vanka is nearly as effective as $\omega = 0.5$).

4.2.2. Braess–Sarazin. Next, we examine Braess–Sarazin as the relaxation scheme and vary the Hartmann number as $Ha = 1$, $Ha = 20$, and $Ha = 80$. We

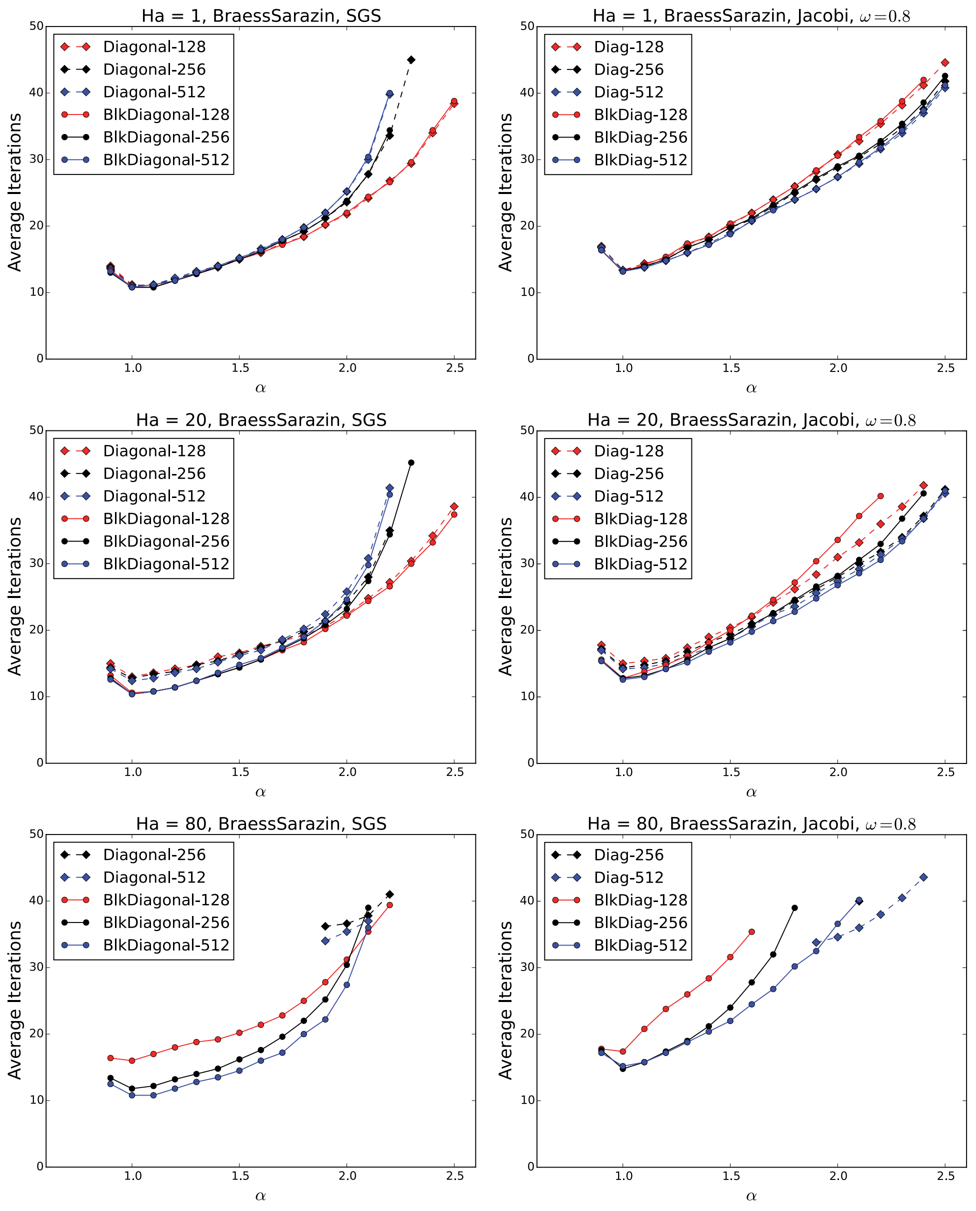

FIG. 3. *The average number of GMRES iterations using the Braess–Sarazin method required to solve the linear systems to a relative tolerance of* 10^{-5} *over all Newton steps for various choices of* α. *The top row shows Ha* $= 1$*; the middle row shows Ha* $= 20$*; and the bottom row shows Ha* $= 80$. *On the left, we use one sweep of SGS to solve the approximate Schur complement system* (3.10)*; on the right, we use one sweep of weighted Jacobi with* $\omega_J = 0.8$.

vary the Braess–Sarazin parameter α from 0.7 to 2.5 in steps of 0.1. In Figure 3, the average number of linear iterations required to reduce the relative linear residual below 10^{-5} is shown. The absence of a data point implies that the linear solver failed to converge to the desired tolerance within the allotted fifty iterations for at least one of the Newton steps. On the left of Figure 3, we use a single sweep of SGS to relax on the Schur complement solve (3.10). On the right, we use a single Jacobi iteration to solve (3.10) with $\omega_J = 0.8$. Over all ω_J between 0.1 and 1.0 in steps of 0.1, choosing $\omega_J = 0.7$ or $\omega_J = 0.8$ offered the best convergence; for uniformity, we show $\omega_J = 0.8$ results throughout this section.

For $Ha = 1$, the optimal parameter choice is clearly $\alpha = 1.0$. Moreover, for $\alpha = 1.0$, Diagonal and Block-Diagonal Braess–Sarazin are identical in the average number of iterations per Newton solve. Notice that there is some flexibility in the parameter choice. Choosing an α near 1.0 still offers speedy convergence—only when $\alpha > 1.9$ does the convergence suffer more rapidly. This trend is more noticeable with the methods that use SGS to solve (3.10), but is also true for the Jacobi case.

For $Ha = 20$, again, the optimal parameter choice is $\alpha = 1.0$. Here we see that the block-diagonal method is more effective than the diagonal method, requiring about two fewer iterations per Newton solve. The trend from the $Ha = 1$ results holds and choosing α near 1.0 reliably ensures good convergence. Again, this is clearer for the SGS methods than the Jacobi methods.

Analogously to the case of Diagonal Vanka, in the $Ha = 80$, the breakdown of the Diagonal Braess–Sarazin method is seen. The method is unable to converge at all for the 128×128 fine-grid test problem, and it requires inconsistent choices of α to converge on the larger grids. Again, taking the diagonal of $\hat{F}$ ignores the contributions from Y and Z, the fluid velocity to magnetics coupling. As the Hartmann number increases, this coupling becomes more pronounced, and this preconditioner is unable to meaningfully capture that important coupling. Hence, from these results, the utility of the block-diagonal method is clear. The optimal parameter for that method is still $\alpha = 1.0$, and at the higher grid sizes, the average number of iterations per Newton step has not increased relative to the $Ha = 1$ and $Ha = 20$ scenarios.

4.2.3. Toward nonuniform meshes. The parameter studies above have been run on uniform meshes. The velocity solution, $\mathbf{u}$, to the Hartmann problem, however, develops a sharp boundary layer as Ha grows larger, as shown in Figure 1. In this case, we could consider a boundary layer–type mesh, in which more elements are located in the regions close to $y = \pm 1$, where $\mathbf{u}$ changes rapidly. To do this, we maintain the tensor-product structure of the grids and shift the points in the y-direction such that half of them are evenly spaced in $[-1, -\tau] \cup [\tau, 1]$ and the other half are evenly spaced in $[-\tau, \tau]$. The points in the x-direction remain evenly spaced in $[-1, 1]$. We then vary the value of τ, effectively shifting the aspect ratios of the cells in the mesh. Here, we note that the aspect ratio of the elements in the boundary layer is $\frac{1}{2\tau}$.

Figure 4 shows the average number of linear iterations required as the aspect ratio of the boundary elements grows (i.e., the spacing in the y-direction grows smaller relative to the spacing in the x-direction), using the optimal parameter choices from sections 4.2.1 and 4.2.2. We see that as the aspect ratio increases, the preconditioner using Full Vanka relaxation maintains stable average iteration counts. However, the performance of the preconditioners using the Block-Diagonal Braess–Sarazin and Economy Vanka schemes diminishes significantly as the aspect ratio increases. In the results below, we return to cells with aspect ratio of unity, noting that the conclusions in this case may not apply to nonuniform meshes, particularly in the case of anisotropic elements.

4.3. Performance analysis. With the algorithms validated against the analytical solution and the optimal parameters found, we now investigate the serial performance of these methods. First, we isolate the performance of the relaxation schemes

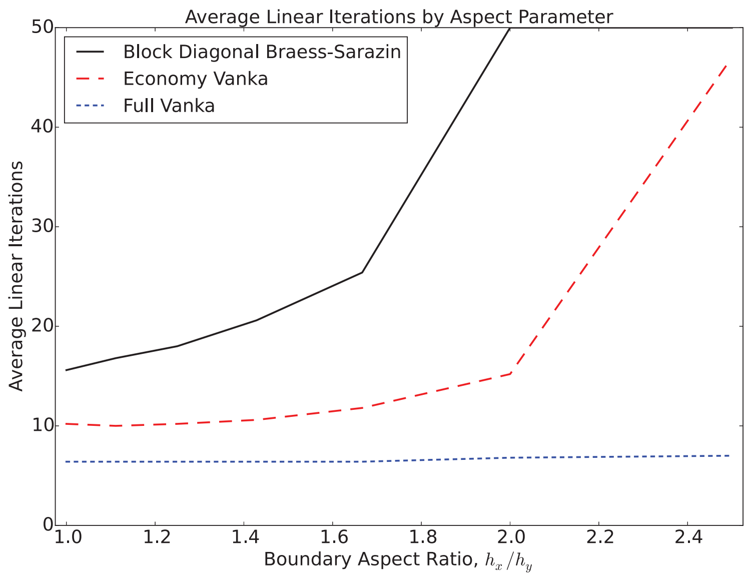

FIG. 4. *The average number of linear iterations required to solve the Hartmann problem with* $Ha = 20$ *on a boundary-layer mesh with* 256×256 *elements.*

independent of the multigrid method and the nonlinear iteration. Then we compare performance for a set of full simulations.

4.3.1. Aggregate timing breakdown. The first set of timing results investigates the performance of these relaxation techniques by themselves. To gather these results, we ran 50 iterations of GMRES on the finest grid on the “zeroth” Newton step using a “one-level multigrid” preconditioner. This forces only one sweep of relaxation per iteration of GMRES. There is only one setup phase per experiment. While we only consider the zeroth Newton step for these tests, similar results are found for other iterations.

Table 2 shows the results for the Vanka methods that we consider. In the setup phase, Economy and Diagonal Vanka are much cheaper than Full Vanka—by nearly a factor of four for Economy Vanka and just slightly more for Diagonal Vanka. This is true both for the extraction of the Vanka matrices and for their factorization. Full Vanka extracts and factors a 76×76 dense matrix for each block. Economy Vanka extracts 75 rows of 4 entries and one row of 75 entries and factors the 3×3 diagonal blocks as they are extracted. Diagonal Vanka only extracts a diagonal of length 75 and one column and one row of length 75. In all cases, we see that the setup time increases by a factor of 4 upon doubling the number of elements in each direction.

Finally, the solve phase results are also shown aggregated over 50 iterations of the scheme. We note first that the total times for the different methods are very different. At all grid sizes, Full Vanka takes more time than Economy Vanka, which takes more time than Diagonal Vanka. However, they are all of the same order of magnitude and within 10% of each other at each grid size. This is because the vast majority of the time is spent updating residuals. The issue is common to overlapping block Gauss–Seidel-type methods. Specifically, the residual must be updated after each

Table 2

Aggregated timings in seconds for 50 *iterations of the three Vanka relaxation schemes. There is only one call to the setup phase, followed by* 50 *calls to the solve phase. "Extract" and "Factor" are the extraction and factorization of the Vanka matrices, and the following "Total" is the total time to set up the method, including some additional software overhead. For the solve phase, "Resid" indicates the time spent updating the residual, "Block^{-1}" indicates the amount of time spent applying the block inverses (recall that the blocks are factored in setup), and the following "Prec*x" indicates the total time spent applying the preconditioner, including some additional overhead.*

		128×128			256×256			512×512		
		Full	Econ	Diag	Full	Econ	Diag	Full	Econ	Diag
Setup	Extract	4.756	1.142	1.031	19.32	4.489	3.941	76.51	18.382	14.337
	Factor	0.793	0.188	0.123	3.212	0.632	0.489	12.63	3.008	1.933
	Total	6.103	1.843	1.335	24.83	7.189	5.174	98.1	30.21	19.14
Solve	Resid	112.3	110.2	112.3	470.2	483.2	465.8	1848	1821	1828
	Block^{-1}	8.282	3.845	1.119	40.33	16.73	4.503	162.2	67.71	17.73
	Prec*x	124.7	118	117.4	528	517	486	2078	1958	1904

block's update has been computed. We implement this as efficiently as possible—only updating those entries in the residual that change. However, it is still required to do as many residual updates as there are pressure DOFs (i.e., one per Vanka block). This is a significant cost, which scales by a factor of 4 as the number of elements in each direction doubles.

If we isolate the time it takes to apply the inverses of the Vanka matrices, we can see the computational advantages offered by Economy and Diagonal Vanka over Full Vanka. Specifically, the inverses for Full Vanka take about 2.4 times as long to apply as those for Economy Vanka for the larger grid sizes. Moreover, the inverses for Economy Vanka take about 3.7 or 3.8 times as long to apply as those for Diagonal Vanka on the larger grid sizes. Nevertheless, even if we were to suppose that the residual computations were done for free, the cheapest Vanka method would take nearly four times as long as the Braess–Sarazin methods.

Table 3 shows the results for the Braess–Sarazin methods, broken down by grid size, the choice of C, and the choice of S^{-1}. In the setup phase, we observe that the explicit computation of C^{-1} is very nearly a factor of 10 greater for the block-diagonal version of Braess–Sarazin. This is not unexpected—after all, we form and explicitly invert the 3×3 blocks in this computation, whereas we need only compute the inverse of a diagonal for the diagonal method. Since computation of $G = C^{-1}B$ then requires a matrix-vector product where the matrix has 4 entries per row, we expect the block-diagonal method to be more expensive than a simple diagonal scaling. In fact, it is almost exactly 4 times more costly. As expected, the time for computing $S = -B^T G$ is essentially the same for both choices of C at each grid size. Finally, we note that as we double the number of elements in each direction, we see an increase in the total setup time by a factor of 4—exactly what we expect.

The solve phase shows timings for 50 iterations. The first notable thing is that using one sweep of SGS costs a very small amount more than using Jacobi. This is reasonable since we must sweep through the grid points twice—forward and backward. Also, the diagonal method is marginally faster than the block-diagonal method. Again, this is not surprising since applying C^{-1} is marginally more expensive in the block-diagonal method. As we double the number of elements in each direction, we again see that the timings scale by a factor of (just slightly smaller than) 4. We do

Table 3

Aggregated timings in seconds for 50 *iterations of the two Braess–Sarazin relaxation schemes, each with two solvers for* (3.10). *There is only one call to the setup phase, followed by* 50 *calls to the solve phase. "C^{-1}" shows the time spent computing C^{-1} for the given method, "$C^{-1}B$" is the time spent explicitly computing $G = C^{-1}B$, "S" is the time spent explicitly computing $S = B^TC^{-1}B = B^TG$, and the following "Total" is the total time to set up the method. "Op*x" indicates the time spent computing* 50 *matrix-vector products, "Relax" indicates the amount of time spent applying the method* (3.10)–(3.11), *and the following "Prec*x" indicates the total time spent applying the preconditioner. This includes computing one new residual each iteration.*

	Grid	128 × 128				256 × 256				512 × 512			
	C	Diagonal		3 × 3		Diagonal		3 × 3		Diagonal		3 × 3	
	S^{-1}	Jac	SGS	Jac	SGS	Jac	SGS	Jac	SGS	Jac	SGS	Jac	SGS
Setup	C^{-1}	0.012	0.016	0.138	0.122	0.046	0.054	0.476	0.542	0.213	0.203	2.011	1.95
	$C^{-1}B$	0.020	0.014	0.061	0.057	0.049	0.047	0.228	0.223	0.186	0.190	0.834	0.834
	S	0.169	0.176	0.178	0.171	0.671	0.677	0.698	0.673	2.692	2.667	2.639	2.658
	Total	0.206	0.212	0.404	0.376	0.776	0.788	1.492	1.529	3.114	3.085	5.849	5.77
Solve	Op*x	0.631	0.712	0.720	0.635	2.493	2.772	2.522	2.810	10.92	10.54	10.76	9.66
	Relax	0.230	0.304	0.288	0.304	0.874	1.153	1.029	1.253	3.657	4.382	4.001	4.728
	Prec*x	1.088	1.25	1.229	1.173	4.085	4.602	4.244	4.739	17.18	17.59	17.33	17.18

see a substantial gap between the time it takes to apply the Braess–Sarazin iteration, (3.10)–(3.11), and the total time spent applying the preconditioner. Part of this gap is explainable as time spent computing residuals: one per iteration to allow the preconditioner code to accept an initial guess when being used as a relaxation scheme. The bulk of this time is the matrix-vector product, so we include the time to do 50 matrix-vector products as well, computed as the time GMRES spends computing the next update direction. The rest of the time is spent in software overhead, mostly converting data objects to the appropriate type and other interface functions that are necessary for the software.

4.3.2. Full simulation timing studies. Finally, we compare the two relaxation procedures for a full simulation. Tables 4 and 5 show timing results for nonlinear solves using Vanka. Tables 6 and 7 show timing results for nonlinear solves using Braess–Sarazin. In the Vanka tables, "Type" indicates which of Full, Economy, or Diagonal Vanka was used. For Braess–Sarazin, this indicates whether the 3 × 3 block-diagonal method or the diagonal method was used. Also in the Braess–Sarazin tables, "S^{-1}" indicates whether one sweep of SGS or Jacobi was used to solve (3.10). "Setup" is the time in seconds to set up the multigrid hierarchy. This time includes computing P and $\mathcal{A}_c$ for each grid, as well as setting up everything for the relaxation scheme. For Vanka, this means computing the Vanka blocks and then forming and factoring the Vanka matrices. For Braess–Sarazin, this means computing C^{-1} and S. "GMRES" indicates the total number of seconds spent iterating in GMRES. The number in parentheses in this column indicates the number of Newton steps required to converge. "Prec" indicates the number of seconds of GMRES that are spent in calls to the preconditioner. The number in parentheses here is the total number of calls to the preconditioner, which is equal to the total number of GMRES iterations across all Newton steps. Finally, "Total" is the total time in seconds spent in Trilinos, including discretization time. We also refer to this as "total time to solution."

Looking at the Vanka setup results in Tables 4 and 5, Full Vanka lags significantly

Table 4
Timing data for the three Vanka methods applied in the full Newton solve for the Hartmann problem with $Ha = 1$ and $Ha = 20$. Three different finest-grid sizes are shown.

		$Ha = 1$				$Ha = 20$			
	Type	Setup	GMRES	Prec	Total	Setup	GMRES	Prec	Total
128^2	Full	49.1	213 (5)	213 (32)	289	50.8	218 (5)	217 (31)	297
	Econ	17.1	322 (5)	321 (51)	366	17.0	304 (5)	303 (48)	348
	Diag	11.3	319 (5)	318 (50)	357	11.4	297 (5)	296 (47)	336
256^2	Full	198	918 (5)	916 (33)	1220	198	895 (5)	892 (32)	1199
	Econ	68.7	1374 (5)	1370 (52)	1548	68.6	1313 (5)	1309 (50)	1486
	Diag	44.1	1338 (5)	1333 (52)	1486	44.3	1354 (5)	1350 (51)	1504
512^2	Full	735	3306 (5)	3297 (33)	4430	737	3126 (5)	3117 (31)	4259
	Econ	258	5186 (5)	5168 (55)	5837	263	4923 (5)	4907 (51)	5585
	Diag	165	5022 (5)	5004 (55)	5575	163	4761 (5)	4745 (52)	5308

Table 5
Timing data for the three Vanka methods applied in the full Newton solve for the Hartmann problem with $Ha = 80$. Three different finest-grid sizes are shown.

	Type	Setup	GMRES	Prec	Total
128^2	Full	50.6	257 (5)	256 (37)	335
	Econ	17.1	346 (5)	345 (55)	390
	Diag	11.1	614 (5)	611 (101)	652
256^2	Full	198	975 (5)	972 (35)	1278
	Econ	68.4	1547 (5)	1542 (59)	1720
	Diag	43.2	2470 (5)	2458 (103)	2615
512^2	Full	596	2629 (4)	2622 (26)	3554
	Econ	207	4949 (4)	4930 (52)	5483
	Diag	172	10220 (5)	10160 (107)	10790

in setup time. Economy Vanka takes nearly a third of the time as Full Vanka, and Diagonal takes about a fourth of the time. This is the case for every grid size. Moreover, as the grid size doubles in each direction, the time to set up the multigrid hierarchies increases by a factor of 4. Thus we have a scalable setup phase.

Inspection of the solve phase timings in Tables 4 and 5 shows that the preconditioning time dominates the total GMRES time (including orthogonalization, etc.) in the linear solves. Furthermore, Full Vanka is faster on all grid sizes in the solve phase because it requires significantly fewer GMRES iterations per step. The additional per-iteration cost of Full Vanka implied by Table 2 is more than offset by the reduced number of iterations required to reach the convergence tolerance of the linear solves, making it the faster algorithm. Comparing Economy Vanka and Diagonal Vanka, we see that these methods require nearly the same number of total GMRES iterations in the physical regimes in which Diagonal Vanka converges well (shown here as $Ha = 1$ and $Ha = 20$). Except in the 256×256 case at $Ha = 20$, Diagonal Vanka is faster in the solve phase. For the total time to solution, the reduced number of iterations for Full Vanka is able to offset the increased setup cost, and it is the fastest of the Vanka methods at all grid sizes.

For Braess-Sarazin, in Tables 6 and 7, we see that the setup for the two methods is nearly the same, with Diagonal Braess–Sarazin being only a few seconds faster,

Table 6
Timing data for the two Braess–Sarazin methods applied in the full Newton solve for the Hartmann problem with $Ha = 1$. Timings for three finest-grid sizes are shown. For each Braess–Sarazin method, we show results for two approximate solvers for (3.10).

			$Ha = 1$				$Ha = 20$			
	Type	S^{-1}	Setup	GMRES	Prec	Total	Setup	GMRES	Prec	Total
128^2	3×3	SGS	10.2	7.62 (5)	6.04 (54)	46.5	9.63	6.38 (5)	5.06 (53)	43.5
	3×3	Jacobi	10.4	8.21 (5)	6.43 (66)	46.8	10.1	8.39 (5)	6.59 (64)	46.1
	Diag	SGS	8.45	6.62 (5)	5.2 (56)	43.2	8.76	8.46 (5)	6.62 (65)	45.6
	Diag	Jacobi	8.58	7.74 (5)	5.99 (67)	43.6	8.84	9.24 (5)	7.15 (71)	46.5
256^2	3×3	SGS	39.1	28.5 (5)	22.3 (54)	179	39.5	27.5 (5)	21.6 (52)	179
	3×3	Jacobi	41.1	30.9 (5)	24.1 (66)	183	39.7	30.9 (5)	23.9 (64)	181
	Diag	SGS	36.9	28.4 (5)	22.3 (55)	182	34.2	30.3 (5)	23.4 (64)	177
	Diag	Jacobi	35.0	33.2 (5)	25.5 (67)	180	35.0	34.2 (5)	26.1 (72)	180
512^2	3×3	SGS	148	91.4 (5)	71.6 (54)	665	147	87.9 (5)	68.7 (52)	663
	3×3	Jacobi	144	103 (5)	79.2 (66)	653	148	97.5 (5)	75.0 (63)	662
	Diag	SGS	135	98.5 (5)	77.1 (55)	665	127	98.2 (5)	75.5 (62)	644
	Diag	Jacobi	129	115 (5)	87.5 (67)	674	128	112 (5)	84.8 (71)	666
1024^2	3×3	SGS	476	260 (4)	205 (43)	2064	603	314 (5)	248 (52)	2497
	3×3	Jacobi	465	302 (4)	231 (53)	2179	608	374 (5)	291 (63)	2589
	Diag	SGS	413	269 (4)	210 (45)	2002	516	364 (5)	283 (61)	2465
	Diag	Jacobi	416	322 (4)	247 (55)	2068	520	417 (5)	319 (71)	2520

Table 7
Timing data for the two Braess–Sarazin methods applied in the full Newton solve for the Hartmann problem with $Ha = 80$. Timings for three finest-grid sizes are shown. For each Braess–Sarazin method, we show results for two approximate solvers for (3.10).

	Type	S^{-1}	Setup	GMRES	Prec	Total
128^2	3×3	SGS	10.5	10.0 (5)	7.79 (80)	48.2
	3×3	Jacobi	9.92	8.74 (5)	6.71 (79)	46.0
256^2	3×3	SGS	39.1	30.6 (5)	23.9 (59)	180
	3×3	Jacobi	39.9	33.4 (5)	25.7 (70)	183
512^2	3×3	SGS	116	70.2 (4)	54.7 (43)	531
	3×3	Jacobi	125	95.7 (4)	73.6 (56)	576
1024^2	3×3	SGS	478	268 (4)	212 (41)	2095
	3×3	Jacobi	481	335 (4)	257 (56)	2154

even on the finest grid case. Furthermore, the hierarchy setup time scales by a factor of 4 as we double the number of elements in each direction.

As with the Vanka methods above, preconditioning dominates the GMRES time. Within a Braess–Sarazin method, using SGS to solve (3.10) results in fewer iterations of GMRES, which translates to less computational time.

Remark 4. The memory footprint of Full Vanka was much higher than the other methods, requiring more than twice as much memory as any other approach. Furthermore, Economy Vanka was marginally more expensive than Diagonal Vanka, as expected. The Diagonal Braess–Sarazin method had the smallest memory footprint, requiring only slightly less than the Block-Diagonal Braess–Sarazin method.

Comparing the Vanka methods to the Braess–Sarazin methods, every variation of Braess–Sarazin is an order of magnitude faster than every variation of Vanka for a given problem on a given grid size. Furthermore, in cases in which the diagonal versions of Vanka and Braess–Sarazin methods fail to provide good convergence, the Economy or Block-Diagonal methods, respectively, provide an alternative that is comparable in both per-iteration computational time and memory usage.

5. Conclusions and future work.

We present extensions of two relaxation techniques from the incompressible fluid dynamics literature to a single-fluid resistive MHD problem in the context of a monolithic geometric multigrid method. We show that both the Vanka and the Braess–Sarazin approaches are effective relaxation schemes for this system with properly chosen relaxation parameters. Furthermore, we present three varieties of Vanka relaxation and two varieties of Braess–Sarazin relaxation. We find that the diagonal approximation versions of each of them lack robustness in physical regimes with strong coupling between the fluid velocity and magnetics (i.e., those with high Hartmann number). In the case of Vanka, the Full and Economy methods are robust across a range of Hartmann numbers and grid sizes, as is the block-diagonal approximation version of Braess–Sarazin.

However, in the serial timings, we see that nearly every aspect of the Braess–Sarazin methods is computationally faster than the Vanka methods. A large overhead in the Vanka computations is seen due to the need to continually update the residuals at each step of the Gauss–Seidel iteration. As the block size is large (76×76), this is not a trivial computation. Within the Vanka method, the reduced number of iterations for Full Vanka is enough to offset the greater per-iteration cost, making it the fastest of the Vanka methods in total time to solution. Within Braess–Sarazin, we generally observe that the total time to solution for the methods is nearly the same for $Ha = 1$, but as the Hartmann number increases, Diagonal Braess–Sarazin becomes less effective and the block-diagonal version becomes the more efficient choice.

A main topic of future work will be considering the finite-element discretization. First, we will examine the inf-sup stability of vector-potential formulations for two-dimensional stationary, incompressible MHD, following [23, 40]. In addition, we will investigate formulations that keep the full $\mathbf{B}$-field (instead of using a vector potential) and involve another Lagrange multiplier for the solenoidal constraint [25, 40]. Extending the multigrid methods presented here to these settings should be straightforward, using the second Lagrange multiplier to define appropriate Vanka and Braess–Sarazin relaxation schemes. Additionally, comparing the monolithic multigrid methods discussed here with block-factorization preconditioners for three-dimensional problems, particularly in parallel settings, is a second main focus of current and future research. Finally, as we noted above, the application of these methods to problems on nonuniform, unstructured, and anisotropic meshes is a topic requiring future research.

REFERENCES

[1] M. F. Adams, R. Samtaney, and A. Brandt, *Toward textbook multigrid efficiency for fully implicit resistive magnetohydrodynamics*, J. Comput. Phys., 229 (2010), pp. 6208–6219.

[2] J. H. Adler, M. Brezina, T. A. Manteuffel, S. F. McCormick, J. W. Ruge, and L. Tang, *Island coalescence using parallel first-order system least squares on incompressible resistive magnetohydrodynamics*, SIAM J. Sci. Comput., 35 (2013), pp. S171–S191.

[3] J. H. Adler, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *First-order system least squares for incompressible resistive magnetohydrodynamics*, SIAM J. Sci. Comput., 32 (2010), pp. 229–248.

[4] J. H. Adler, T. A. Manteuffel, S. F. McCormick, J. W. Ruge, and G. D. Sanders, *Nested iteration and first-order system least squares for incompressible, resistive magne-*

tohydrodynamics, SIAM J. Sci. Comput., 32 (2010), pp. 1506–1526.

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

[6] D. Braess and R. Sarazin, *An efficient smoother for the Stokes problem*, Appl. Numer. Math., 23 (1997), pp. 3–19.

[7] A. Brandt, *Multigrid Techniques:* 1984 *Guide with Applications to Fluid Dynamics*, GMD–Studien 85, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

[8] A. Brandt and N. Dinar, *Multigrid solutions to elliptic flow problems*, in Numerical Methods for Partial Differential Equations, S. V. Parter, ed., Academic Press, New York, 1979, pp. 53–147.

[9] A. Brandt and O. E. Livne, *Multigrid Techniques*, Classics Appl. Math. 67, SIAM, Philadelphia, 2011.

[10] S. Brenner and L. Scott, *The Mathematical Theory of Finite Element Methods*, Texts Appl. Math. 15, Springer-Verlag, New York, 1994.

[11] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[12] L. Chacón, J. M. Finn, and D. A. Knoll, *Nonlinear study of the curvature-driven parallel velocity shear-tearing instability*, Phys. Plasmas, 9 (2002), pp. 1164–1176.

[13] L. Chacón, D. A. Knoll, and J. M. Finn, *An implicit, nonlinear reduced resistive MHD solver*, J. Comput. Phys., 178 (2002), pp. 15–36.

[14] E. Cyr, J. Shadid, and R. Tuminaro, *Teko: A block preconditioning capability with concrete example applications in Navier-Stokes and MHD*, submitted for publication.

[15] E. C. Cyr, J. N. Shadid, R. S. Tuminaro, R. P. Pawlowski, and L. Chacón, *A new approximate block factorization preconditioner for two-dimensional incompressible (reduced) resistive MHD*, SIAM J. Sci. Comput., 35 (2013), pp. B701–B730.

[16] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *Block preconditioners based on approximate commutators*, SIAM J. Sci. Comput., 27 (2006), pp. 1651–1668.

[17] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations*, J. Comput. Phys., 227 (2008), pp. 1790–1808.

[18] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2005.

[19] F. J. Gaspar, Y. Notay, C. W. Oosterlee, and C. Rodrigo, *A simple and efficient segregated smoother for the discrete Stokes equations*, SIAM J. Sci. Comput., 36 (2014), pp. A1187–A1206.

[20] M. W. Gee, U. Küttler, and W. A. Wall, *Truly monolithic algebraic multigrid for fluid-structure interaction*, Internat. J. Numer. Methods Engrg., 85 (2011), pp. 987–1016.

[21] H. Goedbloed and S. Poedts, *Principles of Magnetohydrodynamics: With Applications to Laboratory and Astrophysical Plasmas*, Cambridge University Press, Cambridge, England, 2004.

[22] R. Goldston and P. Rutherford, *Introduction to Plasma Physics*, Taylor and Francis Group, New York, 1995.

[23] M. D. Gunzburger, A. J. Meir, and J. S. Peterson, *On the existence, uniqueness, and finite element approximation of solutions of the equations of stationary, incompressible magnetohydrodynamics*, Math. Comp., 56 (1991), pp. pp. 523–563.

[24] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An overview of the Trilinos project*, ACM Trans. Math. Software, 31 (2005), pp. 397–423.

[25] P. Houston, D. Schötzau, and X. Wei, *A mixed DG method for linearized incompressible magnetohydrodynamics*, J. Sci. Comput., 40 (2009), pp. 281–314.

[26] V. E. Howle and R. C. Kirby, *Block preconditioners for finite element discretization of incompressible flow with thermal convection*, Numer. Linear Algebra Appl., 19 (2012), pp. 427–440.

[27] V. John and L. Tobiska, *Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 33 (2000), pp. 453–473.

[28] D. A. Knoll and L. Chacón, *Coalescence of magnetic islands, sloshing, and the pressure problem*, Phys. Plasmas, 13 (2006), 032307.

[29] M. Larin and A. Reusken, *A comparative study of efficient iterative solvers for generalized*

Stokes equations, Numer. Linear Algebra Appl., 15 (2008), pp. 13–34.

[30] S. P. MacLachlan and C. W. Oosterlee, *Local Fourier analysis for multigrid with overlapping smoothers applied to systems of PDEs*, Numer. Linear Algebra Appl., 18 (2011), pp. 751–774.

[31] R. Moreau, *Magnetohydrodynamics*, Fluid Mech. Appl. 3, Kluwer Academic, Dordrecht, The Netherlands, 1990; translated from the French.

[32] C. Oosterlee and F. Gaspar, *Multigrid relaxation methods for systems of saddle point type*, Appl. Numer. Math., 58 (2008), pp. 1933–1950.

[33] R. P. Pawlowski, E. T. Phipps, and A. G. Salinger, *Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part* I: *Template-based generic programming*, Sci. Program., 20 (2012), pp. 197–219.

[34] B. Philip, L. Chacón, and M. Pernice, *Implicit adaptive mesh refinement for* 2*D reduced resistive magnetohydrodynamics*, J. Comput. Phys., 227 (2008), pp. 8855–8874.

[35] E. G. Phillips, H. C. Elman, E. C. Cyr, J. N. Shadid, and R. P. Pawlowski, *A block preconditioner for an exact penalty formulation for stationary MHD*, SIAM J. Sci. Comput., 36 (2014), pp. B930–B951.

[36] A. Prokopenko, J. J. Hu, T. A. Wiesner, C. M. Siefert, and R. S. Tuminaro, *MueLu User's Guide* 1.0, Tech. Rep. SAND2014-18874, Sandia National Labs, 2014.

[37] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.

[38] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[39] J. Schöberl and W. Zulehner, *On Schwarz-type smoothers for saddle point problems*, Numer. Math., 95 (2003), pp. 377–399.

[40] D. Schötzau, *Mixed finite element methods for stationary incompressible magneto-hydrodynamics*, Numer. Math., 96 (2004), pp. 771–800.

[41] J. Shadid, R. Pawlowski, J. Banks, L. Chacón, P. Lin, and R. Tuminaro, *Towards a scalable fully-implicit fully-coupled resistive MHD formulation with stabilized FE methods*, J. Comput. Phys., 229 (2010), pp. 7649–7671.

[42] S. Sivaloganathan, *The use of local mode analysis in the design and comparison of multigrid methods*, Comput. Phys. Comm., 65 (1991), pp. 246–252.

[43] T. Smith, J. Shadid, R. Pawlowski, E. Cyr, and P. Weber, *Reactor core sub-assembly simulations using a stabilized finite element method*, in Proceedings of the 14th International Topical Meeting on Nuclear Reactor ThermalHydraulics, NURETH-14, Paper NURETH-14-500, 2011.

[44] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.

[45] M. ur Rehman, T. Geenen, C. Vuik, G. Segal, and S. P. MacLachlan, *On iterative methods for the incompressible Stokes problem*, Internat. J. Numer. Methods Fluids, 65 (2011), pp. 1180–1200.

[46] S. P. Vanka, *Block-implicit multigrid calculation of two-dimensional recirculating flows*, Comput. Methods Appl. Mech. Engrg., 59 (1986), pp. 29–48.

[47] R. Verfürth, *A combined conjugate gradient–multigrid algorithm for the numerical solution of the Stokes problem*, IMA J. Numer. Anal., 4 (1984), pp. 441–455.