LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-848035

MATRIX-FREE GPU-ACCELERATED SADDLE-POINT SOLVERS FOR HIGH-ORDER PROBLEMS IN H(div)

T. Kolev

April 24, 2023

SIAM Journal on Scientific Computing

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

arXiv:2304.12387v1 [math.NA] 24 Apr 2023

MATRIX-FREE GPU-ACCELERATED SADDLE-POINT SOLVERS FOR HIGH-ORDER PROBLEMS IN $\boldsymbol{H}(\mathrm{div})$

WILL PAZNER[1], TZANIO KOLEV[2], AND PANAYOT VASSILEVSKI[1]

Abstract. This work describes the development of matrix-free GPU-accelerated solvers for high-order finite element problems in $\boldsymbol{H}(\mathrm{div})$. The solvers are applicable to grad-div and Darcy problems in saddle-point formulation, and have applications in radiation diffusion and porous media flow problems, among others. Using the interpolation–histopolation basis (cf. *SIAM J. Sci. Comput.*, 45 (2023), A675–A702), efficient matrix-free preconditioners can be constructed for the $(1,1)$-block and Schur complement of the block system. With these approximations, block-preconditioned MINRES converges in a number of iterations that is independent of the mesh size and polynomial degree. The approximate Schur complement takes the form of an M-matrix graph Laplacian, and therefore can be well-preconditioned by highly scalable algebraic multigrid methods. High-performance GPU-accelerated algorithms for all components of the solution algorithm are developed, discussed, and benchmarked. Numerical results are presented on a number of challenging test cases, including the "crooked pipe" grad-div problem, the SPE10 reservoir modeling benchmark problem, and a nonlinear radiation diffusion test case.

1. Introduction

This paper is concerned with the efficient solution of high-order finite element problems posed in the space $\boldsymbol{H}(\mathrm{div})$; in particular, we are interested in the grad–div problem

$$-\nabla(\alpha\nabla\cdot\boldsymbol{u})+\beta\boldsymbol{u}=\boldsymbol{f}, \tag{1}$$

and the closely related Darcy-type problem

$$\begin{cases}\boldsymbol{u}+\varepsilon\nabla p=\boldsymbol{f},\\ \nabla\cdot\boldsymbol{u}+\gamma p=g,\end{cases} \tag{2}$$

where $\boldsymbol{u}\colon\Omega\to\mathbb{R}^d$ is a vector-valued unknown, $p\colon\Omega\to\mathbb{R}$ is a scalar unknown ($\Omega\subseteq\mathbb{R}^d$, $d\in\{2,3\}$), and α,β,γ and ε are potentially heterogeneous scalar coefficients. Such problems arise in numerous applications, including, for example, radiation diffusion and porous media flow, among others [9, 13]. We consider finite element discretizations for (1) and (2), where $\boldsymbol{u}$ is approximated in the Raviart–Thomas $\boldsymbol{H}(\mathrm{div})$ finite element space, and p is approximated in the discontinuous L^2 finite element space.

The solution of such problems via high-order finite element methods presents several advantageous features, as well as some challenges. High-order methods promise high accuracy per degree of freedom, and have been demonstrated to achieve high performance on modern supercomputing architectures, in particular those based on graphics processing units (GPUs) and other accelerators [1, 27, 36]. However, the solution of the resulting large linear systems remains challenging for a number of reasons. With increasing polynomial degree, the systems become both more ill-conditioned, and less sparse: the number of nonzeros in the matrix scales like $\mathcal{O}(p^{2d})$ and the condition number typically scales like $\mathcal{O}(p^4/h^2)$, where p is the polynomial degree and d is the spatial dimension [33]. Consequently, it is necessary to develop effective matrix-free preconditioners that give rise to well-conditioned systems without having access to an assembled matrix. There are a number of existing approaches for matrix-free preconditioning elliptic equations; among these are p-multigrid, low-order-refined (also known as FEM–SEM) preconditioning, and hybrid Schwarz preconditioning [30, 32, 35]. However, the extension of these approaches to problems in $\boldsymbol{H}(\mathrm{div})$ is not straightforward; for example, multigrid-type methods with simple point smoothers are known not to perform well for such problems [5].

Effective preconditioners for low-order $\boldsymbol{H}(\mathrm{div})$ discretizations include multigrid with Schwarz smoothers [6], auxiliary space algebraic multigrid preconditioning (e.g. the auxiliary divergence solver, ADS) [28], and algebraic hybridization [16]. We briefly describe some existing approaches for extending these approaches

[1]Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR
[2]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA

to high-order discretizations. In [37], a low-order-refined approach for preconditioning grad–div problems in $\boldsymbol{H}(\mathrm{div})$ was proposed. In that work, the high-order operator discretized using the so-called interpolation–histopolation basis was shown to be spectrally equivalent to the lowest-order discretization posed on a refined mesh. Then, an effective preconditioner for the low-order discretization (for example, the AMG-based ADS solver) can be used to precondition the high-order operator. In [11], a vertex-star domain decomposition was combined with the use of a finite element basis with favorable sparsity properties to develop optimal-complexity multigrid solvers.

In the present work, we develop efficient matrix-free preconditioners for high-order $\boldsymbol{H}(\mathrm{div})$ problems by considering the associated 2×2 saddle-point system. Such saddle-point systems can be preconditioned effectively using block preconditioners [8, 39, 43]. The main challenge associated with developing such methods is the construction of spectrally equivalent approximations to the inverse of the Schur complement. Using favorable properties of the interpolation–histopolation basis (cf. [37]) it is possible to both enhance the sparsity of the block system, and to form sparse M-matrix approximations to the Schur complement that are effectively preconditioned with standard algebraic multigrid methods. Although the block system is indefinite and contains more unknowns that the positive-definite system obtained by discretizing the grad–div problem directly, it is much less computationally expensive to precondition, resulting in significantly decreased cost per iteration, and faster overall solve time compared with alternative approaches. Additionally, the described methods have low memory requirements, are highly scalable, and are amenable to GPU acceleration.

The structure of this paper is as follows. In Section 2 we describe the high-order finite element discretization and the associated linear systems. This section also describes the construction of the block preconditioners, and includes results concerning the estimation of the condition number of the preconditioned system. Section 3 describes the algorithmic details and GPU acceleration of the proposed methods. Numerical examples are presented in Section 4; these include several challenging grad–div and Darcy benchmark problems, as well as a fully nonlinear time-dependent radiation-diffusion example. We end with conclusions in Section 5.

2. Discretization and linear system

In the discrete setting, we consider a mesh $\mathcal{T}$ of the domain Ω, where each element $\kappa \in \mathcal{T}$ is the image of the unit cube $[0,1]^d$ under a (typically isoparametric) transformation T_κ (i.e., $\kappa = T_\kappa([0,1]^d)$). In other words, we consider quadrilateral meshes in two spatial dimensions, and hexahedral meshes in three dimensions. The variational problem associated with (1) is given by

$$(\alpha \nabla \cdot \boldsymbol{u}, \nabla \cdot \boldsymbol{v}) + (\beta \boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}). \tag{3}$$

To discretize (3), we take $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}_h$, where $\boldsymbol{V}_h$ is the degree-p Raviart–Thomas finite element space,

$$\boldsymbol{V}_h = \{\, \boldsymbol{x} \in \boldsymbol{H}(\mathrm{div}) : \boldsymbol{v}|_\kappa \circ T_\kappa \in \boldsymbol{V}_h(\kappa) \,\}.$$

The local space $\boldsymbol{V}_h(\kappa)$ is the image of the reference space $\boldsymbol{V}_h(\widehat{\kappa})$ under the $\boldsymbol{H}(\mathrm{div})$ Piola transformation $\boldsymbol{V}_h(\kappa) = \det(J_\kappa)^{-1} J_\kappa \boldsymbol{V}_h(\widehat{\kappa})$, where J_κ is the Jacobian of the element transformation T_κ. In three dimensions, the reference space is given by $\boldsymbol{V}_h(\widehat{\kappa}) = \mathcal{Q}_{p,p-1,p-1} \times \mathcal{Q}_{p-1,p,p-1} \times \mathcal{Q}_{p-1,p-1,p}$, where $\mathcal{Q}_{p_1,p_2,p_3}$ is the space of trivariate polynomials of degree at most p_i in variable x_i. $\boldsymbol{V}_h(\widehat{\kappa})$ is defined analogously in two dimensions.

After discretization, the linear system of equations

$$\mathsf{A}\mathbf{u} = \mathbf{f} \tag{4}$$

is obtained, where A is the matrix given by the sum of the $\boldsymbol{H}(\mathrm{div})$ stiffness and mass matrices, $\mathbf{u}$ is a vector of degrees of freedom representing $\boldsymbol{u}$, and $\mathbf{f}$ is a dual vector representing $\boldsymbol{f}$.

The grad–div problem (1) can be equivalently recast as the first-order system

$$\begin{cases} \beta \boldsymbol{u} - \nabla(\alpha q) = \boldsymbol{f}, \\ \nabla \cdot \boldsymbol{u} - q = 0, \end{cases} \tag{5}$$

leading to the symmetric variational problem

$$\begin{aligned} (\beta \boldsymbol{u}, \boldsymbol{\sigma}) + (\alpha q, \nabla \cdot \boldsymbol{\sigma}) &= (\boldsymbol{f}, \boldsymbol{\sigma}), \\ (\alpha \nabla \cdot \boldsymbol{u}, \tau) - (\alpha q, \tau) &= 0, \end{aligned} \tag{6}$$

where $\boldsymbol{u}, \boldsymbol{\sigma} \in \boldsymbol{V}_h$ and $q, \tau \in W_h$, where W_h is the degree-$(p-1)$ L^2 finite element space

$$W_h = \{\, w \in L^2(\Omega) : w|_\kappa \circ T_\kappa \in \mathsf{W}_h(\kappa) \,\},$$

The local space $W_h(\kappa)$ is the image of the reference space $W_h(\widehat{\kappa}) = \mathcal{Q}_{p-1}$ under the integral-preserving mapping, $W_h(\kappa) = \det(J_\kappa)^{-1} W_h(\widehat{\kappa})$. Note that the space W_h is equal to the range of divergence operator acting on $\boldsymbol{V}_h$

The variational system (6) can be written in matrix form as

$$\begin{pmatrix} \mathsf{M}_\beta & \mathsf{B}_\alpha^\mathsf{T} \\ \mathsf{B}_\alpha & -\mathsf{W}_\alpha \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathsf{q} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}, \tag{7}$$

where the matrices M, B, and W are defined by

$$\begin{aligned} \mathbf{v}^\mathsf{T} \mathsf{M}_\beta \mathbf{u} &= (\beta \boldsymbol{u}, \boldsymbol{v}), \\ \mathsf{q}^\mathsf{T} \mathsf{B}_\alpha \mathbf{u} &= (\alpha \nabla \cdot \boldsymbol{u}, q), \\ \mathsf{r}^\mathsf{T} \mathsf{W}_\alpha \mathsf{q} &= (\alpha q, r). \end{aligned} \tag{8}$$

It is straightforward to see that eliminating the scalar unknown q from (5) results in the original grad–div system (1). Likewise, the Schur complement of the saddle-point matrix with respect to the $(2,2)$-block $\mathsf{M}_\beta + \mathsf{B}_\alpha^\mathsf{T} \mathsf{W}_\alpha^{-1} \mathsf{B}_\alpha$ is equal to the original $\boldsymbol{H}(\mathrm{div})$ matrix A.

Note that the block system (7) is also closely related to the discretized system corresponding to the Darcy problem (2). In the case of the mixed finite element problem, the resulting system takes the form

$$\begin{pmatrix} \mathsf{M}_{1/\varepsilon} & \mathsf{B}^\mathsf{T} \\ \mathsf{B} & -\mathsf{W}_\gamma \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathsf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathsf{g} \end{pmatrix}, \tag{9}$$

where $\mathsf{M}_{1/\varepsilon}$ is the $\boldsymbol{H}(\mathrm{div})$ mass matrix weighted by ε^{-1}, B corresponds to the unweighted divergence form, and W_γ is the L^2 mass matrix weighted by γ (cf. (8)). In contrast to the variational form corresponding to the grad-div problem, in this context the coefficient γ may be zero (corresponding to the Poisson problem with no reaction term), leading to the saddle-point system with zero $(2,2)$-block

$$\begin{pmatrix} \mathsf{M}_{1/\varepsilon} & \mathsf{B}^\mathsf{T} \\ \mathsf{B} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathsf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathsf{g} \end{pmatrix}. \tag{10}$$

In what follows, we focus on the solution of the 2×2 saddle-point systems (7) and (9). While the original grad-div system is symmetric and positive-definite, and so may be solved using the conjugate gradient method, the saddle-point system is symmetric and indefinite, and so we use the MINRES Krylov method (or GMRES, when using a non-symmetric preconditioner). To precondition this system, we make use of block preconditioners developed in Section 2.2. The preconditioners for the grad-div saddle-point system (7) apply equally well to the Darcy block systems (9) and (10), including the case of zero $(2,2)$-block.

2.1. **Interpolation-histopolation basis.** In order to construct efficient preconditioners for the block systems, we will use properties of a particular choice of bases for the high-order Raviart–Thomas space $\boldsymbol{V}_h$ and L^2 space W_h. These spaces will use the "interpolation-histopolation" basis (see [37] and also [18, 29, 36]). To define this basis, the d-dimensional reference element $[0,1]^d$ is subdivided into p^d subelements (where p is the polynomial degree) whose vertices are given by the d-fold Cartesian product of the $p+1$ Gauss–Lobatto points (see Figure 1). Then, the basis is induced by the following choice of degrees of freedom:

- The degrees of freedom of the Raviart–Thomas space $\boldsymbol{V}_h$ are given by integrals of the normal component over each face of the subelement mesh. The basis functions are induced by the tensor-product of one-dimensional interpolation and histopolation operators.
- The degrees of freedom of the L^2 space W_h are given by integrals over each subelement volume. The basis functions are induced by the tensor-product of one-dimensional histopolation operators.

Given the above definitions, there is a natural identification of Raviart–Thomas degrees of freedom with subelement faces, and L^2 degrees of freedom with subelement volumes. For our purposes, this choice of basis results in several important properties, the proofs of which can be found in [37].

- The Raviart–Thomas mass matrix M defined on $\boldsymbol{V}_h$ is spectrally equivalent to its diagonal, independent of mesh size and polynomial degree.

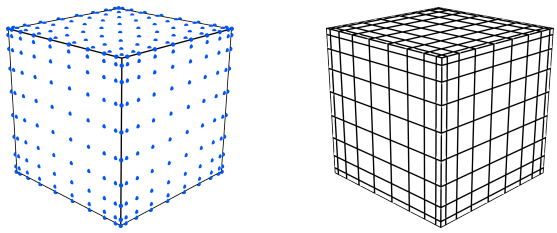


FIGURE 1. Left: high-order ($p = 9$) hexahedral element with 10 Gauss–Lobatto nodes in each dimension. Right: subelement mesh with vertices at the Gauss–Lobatto points.

- The L^2 mass matrix W defined on W_h is spectrally to its diagonal, independent of mesh size and polynomial degree.
- The discrete divergence matrix $\mathsf{D} : \boldsymbol{V}_h \to W_h$ is sparse, with two nonzeros per column (taking values 1 and -1), regardless of polynomial degree. The matrix is entirely determined by the mesh topology (its entries are independent of mesh geometry or distortion), and it coincides exactly with the lowest-order discrete divergence matrix defined on the refined mesh (see the right panel of Figure 1).

Because of the sparsity afforded by the discrete divergence matrix, it is advantageous to consider the transformed saddle-point matrix

$$\mathscr{A} := \begin{pmatrix} \mathsf{M}_\beta & \mathsf{D}^\mathsf{T} \\ \mathsf{D} & -\mathsf{W}_\alpha^{-1} \end{pmatrix} = \begin{pmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{W}_\alpha^{-1} \end{pmatrix} \begin{pmatrix} \mathsf{M}_\beta & \mathsf{B}_\alpha^\mathsf{T} \\ \mathsf{B}_\alpha & -\mathsf{W}_\alpha \end{pmatrix} \begin{pmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{W}_\alpha^{-1} \end{pmatrix} \tag{11}$$

using that $\mathsf{D} = \mathsf{W}_\alpha^{-1}\mathsf{B}_\alpha$ by the definitions in (8). In contrast to the untransformed saddle-point system introduced in (7), the transformed system $\mathscr{A}$ has much sparser off-diagonal blocks (whose sparsity, critically, is independent of the polynomial degree). However, a downside of this transformation is that the matrix $\mathscr{A}$ involves the *inverse* of the L^2 mass matrix W_α. In the context of an iterative solver such as MINRES, this means that the action of W_α^{-1} will need to be computed at every iteration. Because W_h is a discontinuous space, W_α is block-diagonal, and, at least for low orders, the blocks can be factorized efficiently element-by-element. For higher orders, it is not feasible to factorize (or even assemble) the blocks of W_α, and so we turn to iterative matrix-free methods to apply the action of W_α^{-1}. These methods are described in greater detail in Section 2.5, and their performance is studied in the numerical results in Section 3.1. The sparsity of the off-diagonal blocks and the diagonal spectral equivalence properties enumerated above allow for the sparse approximation of the Schur complement and efficient algebraic multigrid preconditioning, enabling the block preconditioning techniques described in the following section.

2.2. **Block preconditioners for saddle-point systems.** At this point, we make a brief digression to state some results concerning block preconditioning of 2×2 saddle-point systems. We do not claim novelty of these results, however we collect them here in a form that will be convenient to reference later (see, e.g., [8, 39, 43] for related results concerning block preconditioning of saddle-point systems). Consider the saddle-point system

$$\mathscr{A} = \begin{pmatrix} \mathsf{A} & \mathsf{B}^\mathsf{T} \\ \mathsf{B} & -\mathsf{C} \end{pmatrix}. \tag{12}$$

While the particular saddle-point system of interest is that given by (11), in this section we only make the assumption that the blocks A and C in (12) are symmetric and positive-definite (but make no other assumptions about the particular form of the blocks). The convergence of MINRES applied to this system can be estimated using bounds on the positive and negative parts of the spectrum,

$$\sigma(\mathscr{A}) \subseteq [\mu_1, \mu_2] \cup [\nu_1, \nu_2] \quad \text{with} \quad \mu_1, \mu_2 < 0, \quad \nu_1, \nu_2 > 0.$$

In particular, we are interested in the ratio C/c, where

$$C = |\lambda|_{\max} = \max\{|\lambda| : \lambda \in \sigma(\mathscr{A})\}, \qquad c = |\lambda|_{\min} = \min\{|\lambda| : \lambda \in \sigma(\mathscr{A})\},$$

resulting in

$$c(\boldsymbol{x}, \boldsymbol{x}) \leq |(\mathscr{A}\boldsymbol{x}, \boldsymbol{x})| \leq C(\boldsymbol{x}, \boldsymbol{x}).$$

Given a symmetric positive-definite preconditioner $\mathcal{B}$, the relevant bounds are those that satisfy

$$c(\mathcal{B}x, x) \leq |(\mathcal{A}x, x)| \leq C(\mathcal{B}x, x), \tag{13}$$

which can then be used to estimate the condition number, $\kappa(\mathcal{B}^{-1}\mathcal{A}) \leq C/c$.

Proposition 1. *Let $\mathcal{B}$ be the block-diagonal preconditioner*

$$\mathcal{B} = \begin{pmatrix} \mathsf{A} & 0 \\ 0 & \mathsf{S} \end{pmatrix},$$

where $\mathsf{S} = \mathsf{C} + \mathsf{D}\mathsf{A}^{-1}\mathsf{D}^{\mathsf{T}}$ is the (negative) Schur complement of $\mathcal{A}$ with respect to the $(1,1)$-block. This preconditioner is optimal in the sense that

$$\sigma(\mathcal{B}^{-1}\mathcal{A}) \subseteq \left[-1, (1-\sqrt{5})/2\right) \cup \left[1, (1+\sqrt{5})/2\right) \text{ and } \kappa(\mathcal{B}^{-1}\mathcal{A}) \leq \frac{\sqrt{5}+1}{\sqrt{5}-1} \approx 2.618\ldots$$

Proof. Let λ be an eigenvalue of $\mathcal{B}^{-1}\mathcal{A}$, i.e.

$$\begin{aligned} \mathsf{A}\mathsf{x} + \mathsf{B}^{\mathsf{T}}\mathsf{y} &= \lambda \mathsf{A}\mathsf{x}, \\ \mathsf{B}\mathsf{x} - \mathsf{C}\mathsf{y} &= \lambda \mathsf{S}\mathsf{y}, \end{aligned}$$

for some $(\mathsf{x}, \mathsf{y}) \neq 0$. Multiplying the first equation by $\mathsf{B}\mathsf{A}^{-1}$ and eliminating x, we obtain

$$\left(\lambda \mathsf{C} + (\lambda^2 - \lambda - 1)\mathsf{S}\right)\mathsf{y} = 0. \tag{14}$$

Considering the case $\lambda \neq 1$, we have $\mathsf{y} \neq 0$ and so $(\mathsf{S}\mathsf{y}, \mathsf{y}) \geq (\mathsf{C}\mathsf{y}, \mathsf{y}) > 0$. Equation (14) then implies that

$$\lambda(\mathsf{C}\mathsf{y}, \mathsf{y}) + (\lambda^2 - \lambda - 1)(\mathsf{S}\mathsf{y}, \mathsf{y}) = 0,$$

and so λ and $(\lambda^2 - \lambda - 1)$ must have opposite signs. We consider the two cases:

- If $\lambda > 0$ then this implies $\lambda^2 - \lambda - 1 < 0$ and so $\lambda < (\sqrt{5}+1)/2$. On the other hand, $(\mathsf{C}\mathsf{y}, \mathsf{y}) \leq (\mathsf{S}\mathsf{y}, \mathsf{y})$ implies $\lambda(\mathsf{S}\mathsf{y}, \mathsf{y}) + (\lambda^2 - \lambda - 1)(\mathsf{S}\mathsf{y}, \mathsf{y}) \geq 0$, and so $\lambda^2 \geq 1$, and $\lambda \in [1, (\sqrt{5}+1)/2)$.
- If $\lambda < 0$ then $\lambda^2 - \lambda - 1 > 0$ and so $\lambda < (1-\sqrt{5})/2$. By the same reasoning as above, $\lambda^2 \leq 1$, so $\lambda \geq -1$, and $\lambda \in [-1, (1-\sqrt{5})/2)$. □

The condition number in Proposition 1 can be improved slightly by using a different scaling of the blocks.

Proposition 2. *The block-diagonal preconditioner*

$$\mathcal{B} = \begin{pmatrix} 2\mathsf{A} & 0 \\ 0 & \mathsf{S} \end{pmatrix},$$

is optimal in the sense that

$$\sigma(\mathcal{B}^{-1}\mathcal{A}) \subseteq [-1, -1/2) \cup [1/2, 1) \qquad \text{and} \qquad \kappa(\mathcal{B}^{-1}\mathcal{A}) \leq 2.$$

The constant 2 is the optimal relative scaling of A in terms of minimizing the bound on $\kappa(\mathcal{B}^{-1}\mathcal{A})$.

Proof. Let $\tau > 0$ be a scaling constant, and consider $\mathcal{B}$ given by

$$\mathcal{B} = \begin{pmatrix} \tau\mathsf{A} & 0 \\ 0 & \mathsf{S} \end{pmatrix}.$$

Repeating the argument from the proof of Proposition 1, we see that either $\lambda = 1/\tau$ or

$$\left(\tau\lambda \mathsf{C} + (\tau\lambda^2 - \lambda - 1)\mathsf{S}\right)\mathsf{y} = 0.$$

Therefore, the eigenvalues are contained in the intervals where the quadratic polynomials $\tau\lambda^2 - \lambda - 1$ and $\tau\lambda^2 + (\tau - 1)\lambda - 1$ have opposite signs. The roots of these polynomials are $\lambda_1 = (1 - \sqrt{1+4\tau})/(2\tau)$, $\lambda_2 = (1 + \sqrt{1+4\tau})/(2\tau)$, $\lambda_1^* = 1/\tau$, and $\lambda_2^* = -1$, respectively. Note that $\lambda_2^* < \lambda_1 < 0 < \lambda_1^* < \lambda_2$, and so

$$\kappa(\mathcal{B}^{-1}\mathcal{A}) \leq \frac{\max(|\lambda_2|, |\lambda_2^*|)}{\min(|\lambda_1|, |\lambda_1^*|)}.$$

We consider three cases depending on the value of τ:

- $\tau < 2$. Then, $|\lambda_2| > |\lambda_2^*|$ and $|\lambda_1| < |\lambda_1^*|$, resulting in the bound

$$\kappa(\mathcal{B}^{-1}\mathcal{A}) \leq \frac{|\lambda_2|}{|\lambda_1|} = \frac{|1 + \sqrt{1+4\tau}|}{|1 - \sqrt{1+4\tau}|} > 2.$$

- $\tau > 2$. Then, $|\lambda_2| < |\lambda_2^*|$ and $|\lambda_1| > |\lambda_1^*|$, resulting in the bound
$$\kappa(\mathcal{B}^{-1}\mathcal{A}) \leq \frac{|\lambda_2^*|}{|\lambda_1^*|} = \frac{1}{1/\tau} > 2.$$
- $\tau = 2$. Then $|\lambda_2| = |\lambda_2^*| = 1$ and $|\lambda_1| = |\lambda_1^*| = 1/2$, resulting in the best bound among these three cases,
$$\kappa(\mathcal{B}^{-1}\mathcal{A}) \leq \frac{|\lambda_2|}{|\lambda_1|} = \frac{|\lambda_2^*|}{|\lambda_1^*|} = 2.$$ □

In the above, we are assuming that the applying the preconditioner $\mathcal{B}$ involves exactly inverting the matrix A and Schur complement S. In practice this is typically infeasible, and so the inverses of these blocks are themselves approximated by spectrally equivalent preconditioners. The following proposition shows that this does not impact the optimality of the preconditioners.

Proposition 3. *Given a saddle-point system $\mathcal{A}$ and preconditioner $\mathcal{B}$, suppose that the inequalities* (13) *hold with constants c and C. Suppose further that $\widetilde{\mathcal{B}}$ is another preconditioner, spectrally equivalent to $\mathcal{B}$, i.e.*

$$\tilde{c}(\widetilde{\mathcal{B}}\boldsymbol{x}, \boldsymbol{x}) \leq (\mathcal{B}\boldsymbol{x}, \boldsymbol{x}) \leq \tilde{C}(\widetilde{\mathcal{B}}\boldsymbol{x}, \boldsymbol{x}). \tag{15}$$

Then, $\widetilde{\mathcal{B}}^{-1}\mathcal{A}$ is uniformly well-conditioned, with condition number bound

$$\kappa(\widetilde{\mathcal{B}}^{-1}\mathcal{A}) \leq C\tilde{C}/(c\tilde{c}).$$

Proof. This follows immediately by combining (13) and (15). □

Remark 1. As a consequence of the above results, a block-diagonal preconditioner of the form

$$\mathcal{B} = \begin{pmatrix} \tau\widetilde{\mathsf{M}} & 0 \\ 0 & \widetilde{\mathsf{S}} \end{pmatrix},$$

where $\widetilde{\mathsf{M}}$ is spectrally equivalent to M and $\widetilde{\mathsf{S}}$ is spectrally equivalent to the Schur complement S, will result optimal (i.e. discretization parameter independent) convergence for MINRES when applied to the saddle-point system $\mathcal{A}$. The relative scaling of the blocks determined by the coefficient τ should be chosen in accordance with Proposition 2

Remark 2. Block-triangular preconditioners of the form

$$\mathcal{B} = \begin{pmatrix} \mathsf{M} & \mathsf{B}^\mathsf{T} \\ 0 & \mathsf{S} \end{pmatrix}$$

result in a preconditioned system with exactly one eigenvalue, $\sigma(\mathcal{B}^{-1}\mathcal{A}) = \{1\}$, and whose minimal polynomial has degree two, guaranteeing convergence of GMRES in at most two iterations [8]. In practice, the diagonal blocks must be replaced with spectrally equivalent approximations $\widetilde{\mathsf{M}}$ and $\widetilde{\mathsf{S}}$. In this setting, convergence estimates for GMRES may be obtained using a field of values analysis, where the resulting bounds depend on the inf-sup constant of the corresponding finite element problem [7]. However, because the triangular preconditioners are non-symmetric, GMRES instead of MINRES must be used, increasing the memory requirements. Whether the reduced number of iterations warrants sacrificing the short-term recurrence is typically problem-specific; in the numerical results we consider only the block-diagonal preconditioner.

Remark 3. Using the so-called Bramble–Pasciak (BP) transformation, it is possible to use a block-triangular preconditioner with the conjugate gradient method in a transformed inner product [10]. This inner product requires a scaled approximation of the $(1,1)$-block of the matrix, typically necessitating the solution of an eigenvalue problem with the matrix $\widetilde{\mathsf{M}}^{-1}\mathsf{M}$. The total work per iteration of the BP–CG method is similar to that of block-diagonally preconditioned MINRES. Preliminary investigations into the performance of the BP–CG method applied to the problems considered presently indicate largely similar performance to that of MINRES with block-diagonal preconditioning (but with the added cost of estimating the extremal eigenvalue of $\widetilde{\mathsf{M}}^{-1}\mathsf{M}$).

2.3. **Application to the grad-div system.** We now apply the block preconditioning techniques developed in Section 2.2 to the mixed ($\boldsymbol{H}(\mathrm{div})$ and L^2) saddle-point system $\mathcal{A}$ in the form given by (11). Using the properties of the interpolation-histopolation basis (see Section 2.1), the $\boldsymbol{H}(\mathrm{div})$ mass matrix M_β is spectrally equivalent to its diagonal, and so we can take $\widetilde{\mathsf{M}} = \mathrm{diag}(\mathsf{M}_\beta)$. We now turn to approximating the Schur complement $\mathsf{S} = \mathsf{W}_\alpha + \mathsf{D}\mathsf{M}_\beta^{-1}\mathsf{D}^\mathsf{T}$. Using the spectral equivalence of W_α to its diagonal, we take $\widetilde{\mathsf{W}} = \mathrm{diag}(\mathsf{W}_\alpha)$, and then define

$$\widetilde{\mathsf{S}} = \widetilde{\mathsf{W}}^{-1} + \mathsf{D}\widetilde{\mathsf{M}}^{-1}\mathsf{D}^\mathsf{T}. \tag{16}$$

Because of the properties of the basis, the matrix $\widetilde{\mathsf{S}}$ takes a particularly simple structure. Each degree of freedom in the space $\boldsymbol{V}_h$ corresponds to a *subelement face*, and each degree of freedom in W_h corresponds to a *subelement volume*. The connectivity pattern of the $\widetilde{\mathsf{S}}$ is determined by the topological connectivity of the subelement mesh. In particular, let $\mathcal{F}(i)$ denote the set of Raviart–Thomas degrees of freedom k corresponding to subelement faces belonging to the subelement volume i. Given two subelement volumes (i.e., two L^2 degrees of freedom) sharing a common face, let k_{ij} denote the degree of freedom corresponding to the shared face. Then, the entries of $\widetilde{\mathsf{S}}$ are given by

$$\widetilde{\mathsf{S}}_{ij} = \begin{cases} \widetilde{\mathsf{W}}_{ii}^{-1} + \sum_{k\in\mathcal{F}(i)} \widetilde{\mathsf{M}}_{kk}^{-1} & \text{if } i = j, \\ -\widetilde{\mathsf{M}}_{kk}^{-1} & \text{if } i \text{ and } j \text{ share common face } k = k_{ij}, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Consequently, the matrix $\widetilde{\mathsf{S}}$ is sparse, since the number of off-diagonal entries is bounded by the number of faces per element (4 for a quadrilateral mesh in 2D, 6 for a hexahedral mesh in 3D), independent of the polynomial degree of the finite element spaces. Additionally, $\mathsf{D}\widetilde{\mathsf{M}}^{-1}\mathsf{D}^\mathsf{T}$ can be viewed as a simple discretization of the Laplacian, and the following arguments show that S is well-preconditioned by methods such as algebraic multigrid.

Proposition 4. *The approximate Schur complement $\widetilde{\mathsf{S}}$ is an M-matrix.*

Proof. This follows from the explicit form (17) using that W_α and M_β are positive-definite, and so $\widetilde{\mathsf{W}}_{ii}^{-1} > 0$ and $\widetilde{\mathsf{M}}_{kk}^{-1} > 0$. □

Furthermore, $\widetilde{\mathsf{S}}$ can be interpreted as a diagonal (approximate) mass matrix $\widetilde{W}$ plus a *weighted graph Laplacian* term $\mathsf{D}\widetilde{\mathsf{M}}^{-1}\mathsf{D}^\mathsf{T}$. The graph Laplacian L of an edge-weighted graph is defined by

$$\mathsf{L}_{ij} = \begin{cases} w_i & \text{if } i = j, \\ -w_{ij} & \text{if } i \sim j, \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

where we write $i \sim j$ if the graph vertices i and j are connected by an edge (with weight w_{ij}). The vertex weight w_i is defined by $w_i = \sum_{i\sim j} w_{ij}$. Comparing (17) to (18), it is straightforward to see that $\mathsf{D}\widetilde{\mathsf{M}}^{-1}\mathsf{D}^\mathsf{T}$ is the graph Laplacian corresponding to the weighted face-neighbor connectivity graph of the subelement mesh, with weights corresponding to the reciprocal of the diagonal entries of M_β (the diagonal entries of M_β scale like area of the corresponding subelement face). Note that this system is very closely related to the lowest-order discontinuous Galerkin discretization described in [37]. Algebraic multigrid preconditioners for linear systems of this form are well-studied, and can be expected to result uniform convergence independent of problem size. The following proposition summarizes the application of block-diagonal preconditioning to the grad-div system.

Proposition 5. *Let $\widetilde{\mathsf{M}}$ denote the diagonal of M_β and let $\widehat{S}$ denote a matrix spectrally equivalent to S (for example, $\widehat{S}^{-1}$ could be given by one AMG V-cycle applied to $\widetilde{\mathsf{S}}$). Then, $\mathcal{B}^{-1}\mathcal{A}$ is uniformly well conditioned, where $\mathcal{B}$ is the block diagonal preconditioner*

$$\mathcal{B} = \begin{pmatrix} \widetilde{\mathsf{M}} & 0 \\ 0 & \widehat{\mathsf{S}} \end{pmatrix}.$$

2.4. **Application to the Darcy system.** The treatment of the Darcy system (9) is largely similar to that of the grad-div system (7). The relevant difference is that in the case of the Darcy system, the off-diagonal blocks B and B^T are not weighted by the coefficient γ. As a result, the transformed system takes the slightly modified form

$$\mathscr{A}' = \begin{pmatrix} \mathsf{M}_{1/\varepsilon} & \mathsf{D}^\mathsf{T} \\ \mathsf{D} & -\mathsf{W}^{-1}\mathsf{W}_\gamma\mathsf{W}^{-1} \end{pmatrix} = \begin{pmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{W}^{-1} \end{pmatrix} \begin{pmatrix} \mathsf{M}_{1/\varepsilon} & \mathsf{B}^\mathsf{T} \\ \mathsf{B} & -\mathsf{W}_\gamma \end{pmatrix} \begin{pmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{W}^{-1} \end{pmatrix} \tag{19}$$

In the case that $\gamma v_h \in W_h$ for all $v_h \in W_h$, the $(2,2)$-block of $\mathscr{A}'$ can be simplified to $-\mathsf{W}_{1/\gamma}^{-1}$. This can be seen by noting that $\mathsf{W}^{-1}\mathsf{W}_\gamma\mathsf{w}$ (where the vector w corresponds to $w_h \in W_h$) is the unique element $u_h \in W_h$ satisfying

$$(u_h, v_h) = (\gamma w_h, v_h)$$

for all $v_h \in W_h$. Likewise, $\mathsf{W}_{1/\gamma}^{-1}\mathsf{W}\mathsf{w}$ is the unique element $u_h' \in W_h$ satisfying

$$(\gamma^{-1}u_h', v_h) = (w_h, v_h)$$

for all $v_h \in W_h$. When $\gamma v_h \in W_h$, we have

$$(u_h, v_h) = (\gamma w_h, v_h) = (w_h, \gamma v_h) = (w_h, \gamma v_h) = (\gamma^{-1}u_h', \gamma v_h) = (u_h', v_h),$$

and hence $\mathsf{W}^{-1}\mathsf{W}_\gamma = \mathsf{W}_{1/\gamma}^{-1}\mathsf{W}$, and so $\mathsf{W}^{-1}\mathsf{W}_\gamma\mathsf{W}^{-1} = \mathsf{W}_{1/\gamma}^{-1}\mathsf{W}\mathsf{W}^{-1} = \mathsf{W}_{1/\gamma}^{-1}$. In particular, this holds when the coefficient γ is piecewise constant (i.e. $\gamma|_\kappa$ is constant for each $\kappa \in \mathcal{T}$). In the general case, this simplification does not hold, and the full form of the $(2,2)$-block is used. In the case of the Darcy problem with zero $(2,2)$-block (corresponding to the Poisson problem $-\nabla\cdot(\varepsilon\nabla p) = g$), the matrix W^{-1} is not required to compute the action of $\mathscr{A}'$, reducing the computational cost of the MINRES iterations.

As before, the $\boldsymbol{H}(\mathrm{div})$ mass matrix $\mathsf{M}_{1/\varepsilon}$ can be approximated by its diagonal, and $\mathsf{W}^{-1}\mathsf{W}_\gamma\mathsf{W}^{-1}$ can be approximated be the product of (the reciprocals of) the diagonals of the terms in the product. The approximate Schur complement $\widetilde{\mathsf{S}}$ takes the same form (17) as in the preceding section, and can be effectively preconditioned with algebraic multigrid.

2.5. L^2 **mass inverse.** Solving the transformed saddle-point system $\mathscr{A}$ given by (11) by a Krylov subspace method such as MINRES (or GMRES in the case of a non-symmetric preconditioner) requires, at every iteration, the action of the operator W^{-1}. The matrix W_α corresponds to the α-weighted L^2 inner product on the space W_h, i.e.,

$$\mathsf{r}^\mathsf{T}\mathsf{W}_\alpha\mathsf{q} = (\alpha q, r), \quad q, r \in W_h.$$

Since the space W_h is discontinuous, W is block-diagonal, with blocks W_i corresponding to the ith element of the mesh κ_i. The block W_i represents the L^2 inner product over κ_i,

$$\mathsf{r}_i^\mathsf{T}\mathsf{W}_i\mathsf{q}_i = (\alpha q, r)_{\kappa_i},$$

where q_i and r_i are subvectors of q and r consisting of those degrees of freedom belonging to the element κ_i. Hence, each block W_i is symmetric and positive-definite, and has size $p^d \times p^d$.

For modest values of p, assembling and factorizing each of the blocks W_i (e.g. using Cholesky factorization or spectral decomposition) is an appropriate strategy, allowing for the application of W_α^{-1} with roughly the same cost as standard operator evaluations or matrix-vector products. However, for higher-order problems, it becomes infeasible to assemble, store, and factorize the blocks W_i. Each block requires $\mathcal{O}(p^{2d})$ storage, and the assembly and factorization require $\mathcal{O}(p^{3d})$ operations. In this case, it is preferable to compute the action of W_α^{-1} matrix-free using an iterative method.

In the matrix-free setting, the action of W_α^{-1} is computed block-by-block using a local conjugate gradient (CG) iteration. Because of properties of the basis for the space W_h, each block W_i is spectrally equivalent to its diagonal, and so using $\mathrm{diag}(\mathsf{W}_i)$ as a preconditioner will result in a number of CG iterations that is independent of the polynomial degree p. Furthermore, the action of W_i can be computed efficiently with matrix assembly using a sum factorization approach [34, 38]. Although this method is asymptotically optimal in terms of iterations (the cost of computing W_α^{-1} is proportional to the cost of evaluating W_α, independent of discretization parameters), the constants in the condition number estimates can be improved by using a change of basis.

In Figure 2 we show the condition number of the 2D and 3D L^2 mass matrix on a single highly skewed element. In each case, we consider the diagonally scaled mass matrix using a different basis. In all cases, the

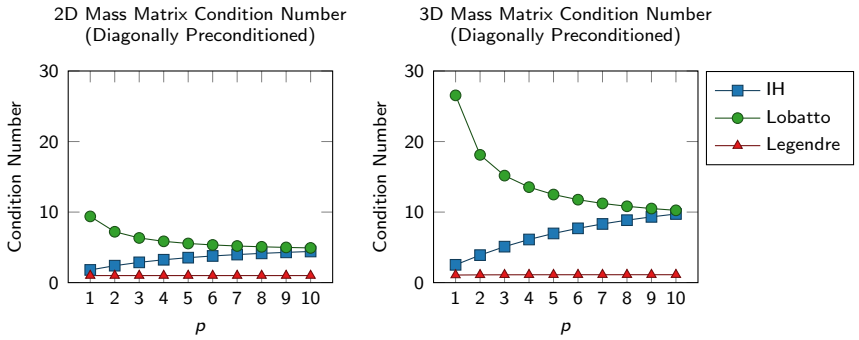


FIGURE 2. Condition number of the L^2 mass matrix preconditioned by diagonal scaling for different choices of basis. Left: 2D case on a skewed quadrilateral. Right: 3D case on a skewed hexahedron.

condition number is bounded with respect to p. As predicted by analytical estimates (cf. [26, 41]) the condition number of the Gauss–Lobatto mass matrix *improves* with increasing p. The condition number obtained using the interpolation–histopolation basis displays a preasymptotic increase in condition number, and for larger p results in largely similar condition numbers as with the Gauss–Lobatto basis. The Gauss–Legendre nodal basis results in the best conditioned system, with maximal condition number of approximately 1.12 in these tests. Note that on elements whose transformation has constant Jacobian determinant (i.e. parallelepipeds) the Gauss–Legendre mass matrix is diagonal (and so in this special case, diagonal preconditioning is exact). However, in the cases we consider here, the elements are skewed, resulting in non-constant Jacobian determinants, and so the mass matrix is not diagonal with any of the bases considered. Even in this case, the diagonally preconditioned matrix remains extremely well conditioned, with condition number only slightly larger than unity.

Motivated by these results, faster CG convergence can be obtained by first performing a change of basis, and then iteratively solving the system in the Gauss–Legendre basis with diagonal preconditioning. Performing the change of basis necessitates the transformation of the right-hand side and the obtained solution. If a non-zero initial guess is used, one additional transformation must also be performed. Each of these transformations is generally less expensive than one operator evaluation, and so this will result in an overall efficiency gain if the total iteration count is reduced by at least two or three. Numerical experiments studying these considerations are presented in Section 3.1.

2.6. **Notes on the untransformed system.** Instead of considering the transformed system $\mathcal{A}$ defined by (11), it is also possible to apply block preconditioners directly to the saddle-point system (7). This has the advantage that each MINRES iteration would require computing the action of W_α rather than W_α^{-1}, potentially leading to computational savings. However, this approach has two drawbacks. Firstly, the off-diagonal blocks B_α and $\mathsf{B}_\alpha^\mathsf{T}$ do not enjoy the same favorable sparsity as D and D^T, and therefore are more expensive to compute. Secondly, and more importantly, the Schur complement of the original system takes the form

$$\mathsf{S}' = \mathsf{W}_\alpha + \mathsf{B}_\alpha \mathsf{M}_\beta^{-1} B_\alpha^\mathsf{T} = \mathsf{W}_\alpha + \mathsf{W}_\alpha \mathsf{D} \mathsf{M}_\beta^{-1} \mathsf{D}^\mathsf{T} \mathsf{W}_\alpha. \tag{20}$$

In order to construct block preconditioners, approximations of the inverse of S' are required. As in the construction of the approximate Schur complement $\widetilde{\mathsf{S}}$ in (16), it is possible to replace W_α and M_β by their diagonals, to obtain a matrix $\widetilde{\mathsf{S}}'$ spectrally equivalent to S'. This approximation of the second term on the right-hand side of (20) is significantly worse that the approximation of the transformed Schur complement S by (16), since it involves the product of three approximations rather than one. A very rough estimate would suggest that, while still asymptotically optimal, the condition number of $\widetilde{\mathsf{S}}'^{-1}\mathsf{S}'$ could be very roughly bounded by the cube of the condition number of $\widetilde{\mathsf{S}}^{-1}\mathsf{S}$.

In Figure 3, we display computed condition numbers of the transformed preconditioned Schur complement $\widetilde{\mathsf{S}}^{-1}\mathsf{S}$ and the untransformed version $\widetilde{\mathsf{S}}'^{-1}\mathsf{S}'$ for increasing polynomial degree. We consider both the unit cube $[0,1]^d$, and a skewed single-element mesh (the same mesh used in the results shown in Figure 2).

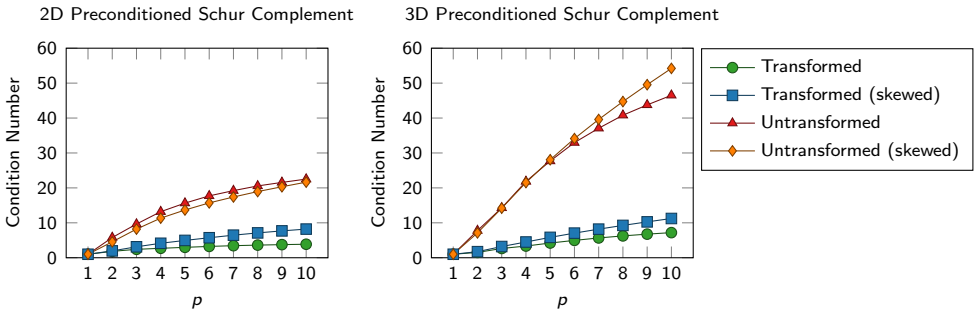


FIGURE 3. Computed condition numbers of preconditioner Schur complement. The transformed system is given by $\widetilde{\mathsf{S}}^{-1}\mathsf{S}$, and the untransformed system is given by $\widetilde{\mathsf{S}}'^{-1}\mathsf{S}'$.

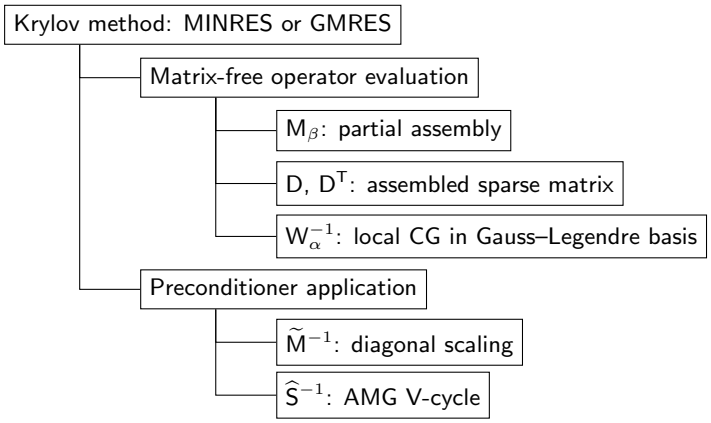


FIGURE 4. Solver diagram for block preconditioners applied to the transformed saddle-point system (21).

Although both versions of the preconditioned Schur complement have $\mathcal{O}(1)$ condition number with respect to polynomial degree, the empirical results show that the transformed Schur complement results in significantly smaller condition number, in accordance with arguments described above. These results justify the use of the transformed system (11), despite the increased cost per iteration incurred by the inverse L^2 mass matrix.

3. Algorithmic details and GPU acceleration

In this section, we give a brief overview of the solution algorithm and describe the strategies used for GPU acceleration. To solve the variational problem (6), we solve the saddle-point system

$$\begin{pmatrix} \mathsf{M}_\beta & \mathsf{D}^\mathsf{T} \\ \mathsf{D} & -\mathsf{W}_\alpha^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \widetilde{\mathsf{q}} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}, \tag{21}$$

such that $(\mathbf{u}, \mathsf{q})$ for $\mathsf{q} = \mathsf{W}_\alpha^{-1}\widetilde{\mathsf{q}}$ gives the solution to the untransformed system (7). A schematic of the solution procedure is shown in Figure 4. This system is solved using MINRES (with the block-diagonal preconditioner) or GMRES (with block-triangular preconditioners). Each iteration requires application of the operators M_β, D, D^T and W_α^{-1}, and application of the preconditioners $\widetilde{\mathsf{M}}^{-1}$ and $\widehat{\mathsf{S}}^{-1}$. The high-order mass matrix M_β can be applied efficiently using matrix-free algorithms with sum factorization (cf. [1, 21]). The discrete divergence operator D can be represented explicitly as a sparse matrix (where the number of nonzeros per row remains bounded, independent of the polynomial degree). The efficient application of W_α^{-1} with GPU acceleration is discussed in Section 3.1. The preconditioner $\widetilde{\mathsf{M}}^{-1}$ for the $(1,1)$-block is simply diagonal scaling, and is trivially parallelizable. In this work, we use algebraic multigrid for the approximate Schur complement preconditioner; details are discussed in Section 3.2.

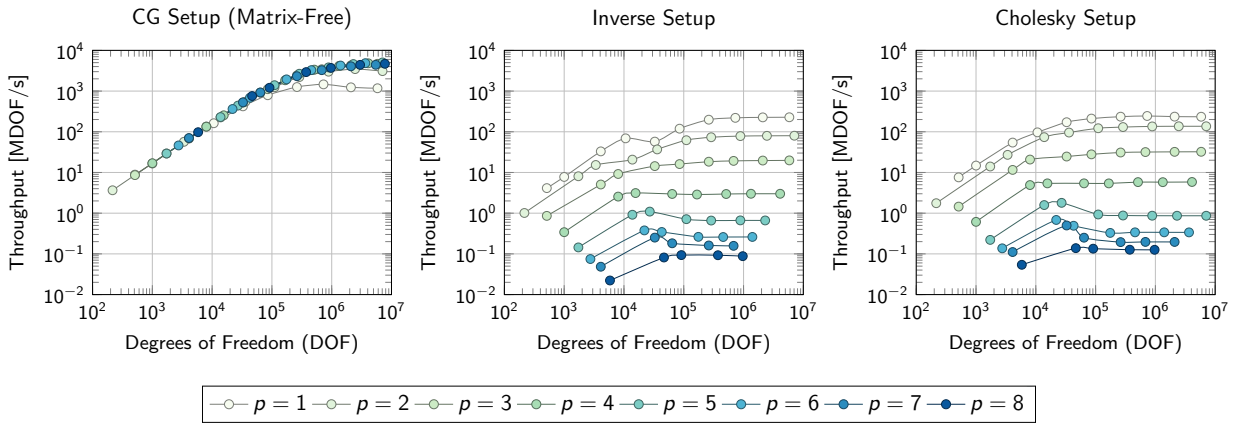


FIGURE 5. GPU DG mass inverse setup phase.

3.1. **GPU acceleration of the DG mass inverse.** Because each iteration involving the system (21) requires computing the action of W_α^{-1}, it is imperative to develop fast algorithms for the action of the inverse of the DG mass matrix. As described in Section 2.5, we consider several approaches, depending on the polynomial degree and spatial dimension. For relatively small polynomial degrees, it is feasible to assemble and store the L^2 mass matrix, allowing for the use of direct methods; for higher polynomial degrees, the memory requirements and computational time required to assemble the matrix render this option infeasible, necessitating the use of iterative solvers.

Since this matrix is block-diagonal, in the case where the matrix may be assembled, each block is stored as a dense matrix, which can then be factorized. These operations are particularly well-suited for use with batched linear algebra libraries [19]. The performance of batched LU solvers and matrix-matrix multiplication operations on GPUs has been well-studied [2, 42]. Presently, we consider two approaches. The first approach is to perform a batched Cholesky factorization (`potrf`) during the setup, and then each application of the mass inverse will perform a triangular solve (`potrs`). Batched versions of these routines are available on Nvidia and AMD GPU hardware through the cuSOLVER and rocSOLVER libraries. The second approach computes the explicit inverse of each block (`getrf` followed by `getri`), and then each application is computed using a batched matrix-vector product (`gemv`). While the computation of the inverse matrix necessitates additional overhead and is less numerically stable than solving the system using the Cholesky factorization, it has the advantage that the application of the inverse is computed using a matrix-vector product rather than a triangular solve, which is better-suited for fine-grained parallelism. For symmetric positive-definite matrices of small sizes, matrix inversion typically possesses satisfactory stability bounds [15]. Since direct methods are only relevant for small-sized matrices in this application, the numerical stability of explicit matrix inverses is not a primary concern.

For larger polynomial degrees, direct methods become impractical or infeasible. In this case, we perform conjugate gradient iterations locally on each element in parallel. Because the system is block-diagonal, each conjugate gradient solver is independent, and no global reductions are required. In our implementation, the entire CG iteration is performed in a single (fused) kernel, with one block of threads per element. The number of threads per block is chosen to facilitate the application of the local mass operator. This action corresponds to the "*CEED BP1*" benchmark problem, and highly optimized GPU implementations for this operation are available [2, 17]. This kernel also performs the initial change of basis operations, allowing the CG iteration to solve the better-conditioned Gauss-Legendre mass matrix, regardless of the user's choice basis for the discretization (see Figure 2). The vector operations (`axpy`) in the CG method are local to each element, and are trivially parallelizable. The Dot products required by CG are performed using parallel reductions in each block.

In Figure 5, we display GPU kernel throughput for the mass solver setup operations using these approaches. The matrix-free CG setup consists of precomputing computing data (such as geometric factors) at quadrature points and assembling the diagonal of the matrix to be used for Jacobi preconditioning. The setup for the direct methods requires assembling the local mass matrices and performing the dense factorizations.

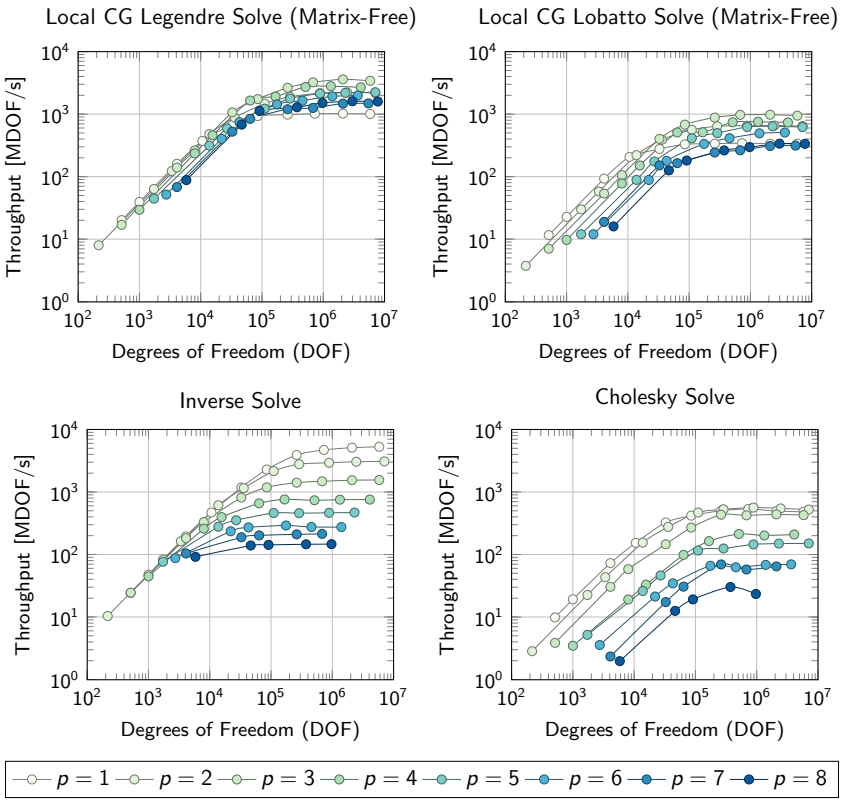


FIGURE 6. GPU DG mass inverse solve phase.

The Cholesky and explicit inverse methods exhibit largely similar performance: the highest performance is achieved for the lowest-order discretizations (i.e. larger batches of smaller matrices). Performance decreases monotonically as the polynomial degree (and matrix size) increase; large problems at $p \geq 5$ could not be run to completion because of memory limitations. On the other hand, the CG setup, including assembly of the diagonal, lends itself to sum factorization, requiring $\mathcal{O}(p^{d+1})$ operations and $\mathcal{O}(p^d)$ memory, leading to significantly improved performance for higher polynomial degrees.

In the context of the iterative solvers considered presently, the setup time is of lesser concern than the operator application time: the setup is performed once, while the action of W^{-1} must be computed at least once every iteration (and twice every iteration in the case of variable-coefficient Darcy problems, see Section 2.4). Figure 6 displays GPU kernel throughput for computing the action of W^{-1} using the methods described in this section. These results make clear that significantly higher performance is achieved in the CG solvers by first changing basis to the Gauss–Legendre basis, and then solving the better-conditioned system (cf. Figure 2). The matrix-free methods using the Gauss–Legendre basis consistently achieve over 2×10^9 degrees of freedom per second, with a peak performance of 3.6×10^9 at $p = 3$. In contrast, a standard CG iteration with global reductions and without fused kernels achieves peak performance about 2–3 orders of magnitude slower than the local CG approach described here.

Comparing the direct solvers, it is clear that using the precomputed explicit inverse matrix results in significantly faster performance than the Cholesky factorization. This is likely attributable to the increased fine-grain parallelism possible when performing batched matrix-vector products compared with triangular solves. For $p = 1$ and $p = 2$, the explicit inverse results in the fastest solve time of all methods considered; the highest throughput observed was 5.2×10^9 degrees of freedom per second, achieved for the largest problem size with $p = 1$. For $p = 3$, the explicit inverse has throughput roughly equal to that of the Gauss–Legendre CG iteration (with significantly more expensive setup costs), and for $p \geq 4$, the Gauss–Legendre CG iteration is the most performant method considered.

TABLE 1. DG mass inverse setup and solve run times. Solve time is reported for 100 applications of the inverse. The fastest method for each polynomial degree is shown in bold. For $p = 8$, there was insufficient GPU memory to compute the explicit inverse, indicated by "—".

		$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	$p=7$	$p=8$
CG	Setup	**0.0016**	**0.0007**	**0.0004**	**0.0004**	**0.0004**	**0.0004**	**0.0005**	**0.0004**
	Solve ×100	0.1937	0.0841	**0.0556**	**0.0598**	**0.0697**	**0.0770**	**0.0981**	**0.0845**
	Total	0.1953	0.0847	**0.0561**	**0.0603**	**0.0701**	**0.0774**	**0.0986**	**0.0850**
Inverse	Setup	0.0071	0.0197	0.0782	0.4544	2.2950	5.8911	10.0749	—
	Solve ×100	**0.0383**	**0.0598**	0.1149	0.2296	0.3690	0.6095	0.8069	—
	Total	**0.0454**	**0.0796**	0.1932	0.6840	2.6640	6.5007	10.8818	—
Cholesky	Setup	0.0076	0.0128	0.0539	0.3077	1.9638	4.9905	8.8000	12.4510
	Solve ×100	0.3466	0.3182	0.3996	0.8135	1.1570	2.3312	2.5846	5.9352
	Total	0.3542	0.3310	0.4535	1.1212	3.1208	7.3216	11.3846	18.3862

We conclude with a performance comparison on a series of problems with roughly 1.7×10^6 DOFs, with polynomial degree $p = 1, 2, \ldots, 8$. In order to benchmark the solvers in a situation relevant to the saddle-point solvers considered in this paper, on each problem we apply the inverse mass matrix 100 times. In Table 1, we present the time required to construct the inverse (or set up the matrix-free CG), the time required for 100 inverse applications, and the total time. The shortest runtime for each polynomial degree is shown in bold. The matrix-free CG method has the fastest setup time among the methods considered, with roughly constant time required for all polynomial degrees (the longest setup time was required for $p = 1$, which is consistent with the lower throughput results shown in Figure 5). The explicit inverse resulted in the fastest overall runtimes for $p = 1$ (4.3× faster than CG) and $p = 2$ (1.1× faster than CG). For polynomial degrees $p \geq 3$, the matrix-free CG resulted in the fastest runtime; the observed speed-up increases dramatically with the polynomial degree. For $p = 8$, there was insufficient memory to compute the explicit inverse. For this test case, the CG setup time was 30,000× faster than the Cholesky factorization, the solve time was 70× faster than the triangular solve, resulting in an overall speed-up of 216×. The Cholesky factorization was the slowest of the three methods considered; the slightly faster factorization time compared with the explicit inverse was outweighed by the slower inverse applications.

Based on these results, in the contexts where the inverse of a fixed mass matrix will be applied many times (such as the iterative solvers considered presently, or in the case of time stepping on a fixed mesh), it would be efficient to use explicit inverses for $p \leq 2$, and matrix-free CG for $p \geq 3$. If, on the other hand, the inverse of the mass matrix will be applied only once (for example, in situations involving a moving mesh, necessitating the recomputation of the mass matrix at every time step), the faster setup time suggests that the matrix-free CG iteration should be used even for $p \leq 2$.

3.2. **Schur complement preconditioning.** We now consider GPU-accelerated preconditioning of the approximate Schur complement $\widetilde{\mathsf{S}} = \widetilde{\mathsf{W}}^{-1} + \mathsf{D}\widetilde{M}^{-1}D^{\mathsf{T}}$, where we recall that $\widetilde{\mathsf{W}}$ is the diagonal of the L^2 mass matrix, $\widetilde{M}$ is the diagonal of the $\boldsymbol{H}(\mathrm{div})$ mass matrix, and D is the discrete divergence matrix. The approach taken presently is algebraic multigrid preconditioning, and so the first step is the assembly of the matrix. The diagonals $\widetilde{\mathsf{W}}$ and $\widetilde{\mathsf{M}}$ can be assembled efficiently using sum factorization [40]. Because of the properties of the interpolation–histopolation basis, the discrete divergence matrix D has the same structure as the lowest-order divergence matrix on a refined mesh. In particular, the entries D_{ij} are given by

$$\mathsf{D}_{ij} = \begin{cases} \sigma_{ij} & \text{if } j \in \mathscr{F}(i), \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathscr{F}(i)$ denotes the set of subelement faces adjacent to the subelement volume i (cf. Section 2.3), and $\sigma_{ij} = \pm 1$ denotes the orientation of the Raviart-Thomas basis function on subelement face j relative to subelement volume i. This matrix can be constructed directly in CSR format efficiently and in parallel on

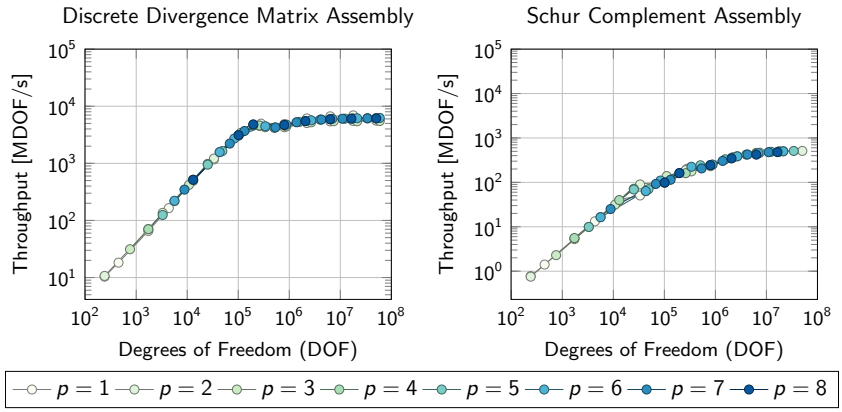


FIGURE 7. Throughput for discrete divergence and approximate Schur complement assembly.

the GPU. The algorithm used to construct this matrix is given in Algorithm 1. The kernel throughput of this algorithm is shown in the left panel of Figure 7.

Algorithm 1 Construction of the discrete divergence matrix D in CSR format.

$N \leftarrow \#$ of L^2 DOFs
parallel for $i \in \{0, \dots, N-1\}$ **do** ▷ For each L^2 DOF i (subelement volume)
  $I[i] \leftarrow 2di$ ▷ $2d$ nonzeros per row
  $e \leftarrow$ `element_map`$[i]$ ▷ e is the mesh element containing DOF i
  $i_{loc} \leftarrow$ `l2_global_to_local`$[i]$ ▷ i_{loc} is the local L^2 DOF index of i within e
  for $k \in \{1, \dots, 2d\}$ **do** ▷ For each subelement face adjacent to i
    $j_{loc} \leftarrow$ `volume_to_face`$[k, i_{loc}]$ ▷ j_{loc} is the local RT DOF index of the subelement face k
    $\sigma_{loc} \leftarrow$ `local_orientation`$[j_{loc}]$ ▷ σ_{loc} is the orientation of DOF j_{loc} in the reference element
    $j \leftarrow$ `rt_local_to_global`$[e, j_{loc}]$ ▷ j is the global DOF index
    $\sigma_{glob} \leftarrow$ `global_orientation`$[j]$ ▷ σ_{loc} is the orientation of the j relative to j_{loc}
    $J[2di + k] \leftarrow j$
    $A[2di + k] \leftarrow \sigma_{loc}\sigma_{glob}$
  end
end
$I[N] \leftarrow 2dN$

After constructing the discrete divergence, the approximate Schur complement can be constructed by computing the sparse matrix-matrix product $\mathsf{D}\widetilde{\mathsf{M}}^{-1}\mathsf{D}^\mathsf{T}$. In the current implementation, this product is computed using the sparse triple-product algorithms provided by the *hypre* solvers library [20]. The throughput for the triple product operation is shown in the right panel of Figure 7. Given the assembled Schur complement in CSR format, the approximate inverse $\widehat{\mathsf{S}}^{-1}$ is given by one V-cycle of an algebraic multigrid preconditioner; in this work, we use the BoomerAMG preconditioner from *hypre* [24]. Detailed GPU performance studies for AMG preconditioners applied to similar problems are given in [36].

4. Numerical results

In the following sections, we test the performance of the saddle-point solvers on a number of grad–div and Darcy test problems. GPU results were performed on LLNL's *Lassen* supercomputer, with four Nvidia V100 GPUs per node. The methods were implemented in the framework of the open-source MFEM finite element library [4], see also `https://mfem.org`. A relative tolerance of 10^{-12} was used as the stopping criterion for all iterative solvers, unless specifically noted otherwise. The BoomerAMG algebraic multigrid preconditioner from the *hypre* library is used to precondition the approximate Schur complement [24], and *hypre*'s ADS preconditioner applied to the low-order-refined system is used as a comparison [28]. The AMG

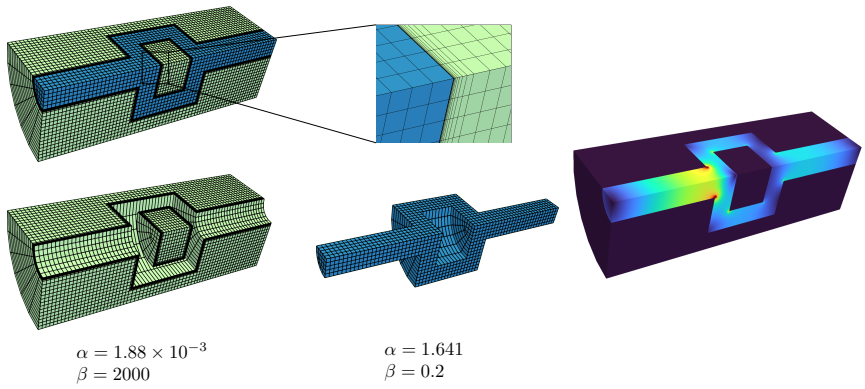


FIGURE 8. Crooked pipe grad-div problem (figure reproduced from [37]). Left panel: mesh with inset showing anisotropic elements and definition of piecewise constant coefficients on subdomains. Right panel: illustration of solution with constant forcing term.

preconditioner for the Schur complement uses parallel maximal independent set (PMIS) coarsening with no aggressive coarsening levels.

4.1. **Grad-div problem: crooked pipe.** As a first numerical example, we consider the "crooked-pipe" grad-div problem. This problem, which serves as a model problem related radiation diffusion simulations, is posed on a cylindrical sector. The elements are divided into two materials; the interface between the materials consists of highly stretched, anisotropic elements, increasing the problem difficulty. We solve the problem

$$\nabla(\alpha \nabla \cdot \boldsymbol{u}) - \beta \boldsymbol{u} = \boldsymbol{f},$$

where the coefficients α and β are piecewise constants determined by the materials. A schematic of this problem is shown in Figure 8. In the outer region (colored green in Figure 8), we have $\alpha = 1.88 \times 10^{-3}$ and $\beta = 2000$, and in the inner region (colored blue in Figure 8), we have $\alpha = 1.641$ and $\beta = 0.2$. This problem, first proposed in [23], was considered as a benchmark problem using the ADS solver in [28], using algebraic hybridization in [16], and with low-order-refined preconditioning in [37].

We use the same problem setup as in [37], and solve this problem with increasing polynomial degree. We compare the performance of the saddle-point solvers discussed presently to applying the ADS preconditioner directly to the high-order assembled matrix, and to ADS applied to the spectrally equivalent low-order-refined system. We first compare CPU results computed using 144 MPI ranks on four nodes of LLNL's *Quartz* supercomputer, which is the same configuration considered in [37]. The results are shown in Table 2. Both methods display a pre-asymptotic increase in the number of iterations with increasing polynomial degree; no increase in the number of iterations for the saddle-point solver was observed when increasing the polynomial degree from $p = 5$ to $p = 6$, suggesting that the asymptotic regime has been reached. At $p = 6$, the saddle-point solver required $1.9\times$ as many iterations as at $p = 2$, and the LOR–ADS solver required $2.5\times$ as many iterations. Overall, the saddle-point solver required approximately $1.5\times$ as many iterations as the LOR–ADS solver, however it was faster overall by a factor of 6–9×. The reason for this speed-up (despite a larger number of iterations) is that the saddle-point preconditioner is much less computationally expensive than the ADS preconditioner: each application of the saddle-point preconditioner requires just one diagonal scaling and one scalar AMG V-cycle (applied to the smaller L^2 system), whereas each application of ADS requires (depending on the specific solver settings) around 11 AMG V-cycles.

These tests are repeated on the GPU using one node of LLNL's *Lassen* with 4 Nvidia V100 GPUs. The results are shown in Table 3. The iteration counts for the GPU test are larger than those obtained on the CPU. This is because different BoomerAMG settings are used when *hypre* is run on the CPU or GPU [20]. In particular, hybrid Gauss-Seidel smoothing is used on the CPU, while ℓ_1-Jacobi smoothing is used on the GPU. This appears to have a more significant detrimental effect on the ADS solver; the iteration counts for the saddle-point solver are largely similar to the CPU case. The saddle-point solver consistently achieves 9–15× speed-up compared with the LOR–ADS solver, with greater speed-up at higher orders. At $p = 6$, the

TABLE 2. CPU results from the crooked pipe test problem (144 CPU cores on LLNL's *Quartz*).

	Saddle-Point		LOR–ADS			
p	Its.	Time (s)	Its.	Time (s)	Speed-up	# DOFs
2	168	0.26	96	1.73	6.65×	356,500
3	231	0.87	143	5.73	6.59×	1,190,115
4	269	2.11	172	14.91	7.07×	2,805,520
5	299	4.45	206	36.90	8.29×	5,461,375
6	323	8.73	241	80.84	9.26×	9,416,340

TABLE 3. GPU results from the crooked pipe test problem (4 Nvidia V100 GPUs on LLNL's *Lassen*).

	Saddle-Point		LOR–ADS			
p	Its.	Time (s)	Its.	Time (s)	Speed-up	# DOFs
2	193	0.85	143	7.73	9.09×	356,500
3	251	1.52	206	14.97	9.85×	1,190,115
4	298	2.19	269	23.72	10.83×	2,805,520
5	335	3.49	332	37.60	10.77×	5,461,375
6	360	4.01	392	61.94	15.45×	9,416,340

TABLE 4. Matrix-based methods: CPU results from the crooked pipe test problem (144 CPU cores on LLNL's Quartz).

	Hybridization		ADS		
p	Its.	Time (s)	Its.	Time (s)	# DOFs
2	28	0.16	74	2.49	356,500
3	35	0.86	85	19.20	1,190,115
4	36	5.73	92	100.77	2,805,520
5	31	29.82	102	390.32	5,461,375
6	36	245.54	112	1293.12	9,416,340

total solve time using the saddle-point solver using the GPU on one node of *Lassen* is over twice as fast as on the CPU using 4 nodes of *Quartz*.

For comparison, we also include CPU results for the matrix-based algebraic hybridization and ADS solvers in Table 4; GPU results are not reported because GPU-accelerated versions of these algorithms are not available. Consistent with the results reported in [16], the algebraic hybridization solver greatly outperforms the ADS solver on all cases considered. Similarly, the LOR–ADS solver outperforms the ADS solver for $p \geq 2$, as reported in [37], and consequently the saddle-point solver also compares favorably to matrix-based ADS for these test cases. The comparison with algebraic hybridization is more interesting. The hybridization solver (which performs a conjugate gradient iteration directly on the hybridized Schur complement system) requires significantly fewer iterations than either ADS or the saddle-point solver. For $p = 2$, hybridization results in the fastest overall runtime, and for $p = 3$, we obtained roughly equal runtimes with the hybridization and saddle-point solvers. However, for $p > 3$, the time required to assemble and set up the hybridized solver becomes prohibitive, and the saddle-point solver results in significant speed-ups. For $p = 4$, the saddle-point solver is almost twice as fast, and for $p = 6$, the saddle-point solver results in an almost 30× speedup. We conclude that for relatively low orders, when assembling and forming the hybridized Schur complement system is not prohibitively expensive, algebraic hybridization is a highly competitive option. For higher orders, matrix-free methods such as the saddle-point solver discussed presently become necessary, and we note that the benefits of the matrix-free methods, such as reduced memory footprint, are more pronounced on GPU-based architectures.

4.2. **Darcy flow: SPE10 benchmark.** In this section, we consider the SPE10 benchmark problem from the reservoir simulation community [14]. This problem has been widely used as a test case for solvers in

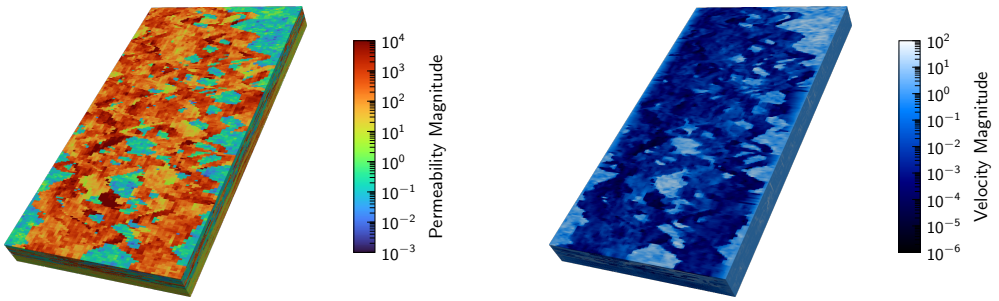


FIGURE 9. SPE10 problem, logarithmic color scale (cf. [14]). Left: magnitude of permeability coefficient. Right: solution velocity magnitude.

TABLE 5. Scalability study for the SPE10 benchmark problem.

$p = 4$				$p = 6$			
# GPUs	# DOFs	Its.	Time (s)	# GPUs	# DOFs	Its.	Time (s)
1	3.36×10^6	147	3.25	1	3.30×10^6	165	3.14
2	5.76×10^6	155	4.31	2	5.87×10^6	166	4.36
4	1.20×10^7	157	5.15	4	1.13×10^7	166	4.82
8	2.68×10^7	160	6.45	8	2.63×10^7	165	6.41
16	4.58×10^7	159	6.35	16	4.67×10^7	165	6.19
32	9.53×10^7	164	6.94	32	9.01×10^7	171	7.20
64	2.13×10^8	164	8.62	64	2.10×10^8	176	8.97
128	3.66×10^8	166	8.55	128	3.73×10^8	173	8.79
256	7.61×10^8	165	10.44	256	7.20×10^8	175	10.53

the literature, see e.g. [16, 25]. The problem geometry is $\Omega = [0, 1200] \times [0, 2200] \times [0, 170]$, represented as a Cartesian mesh with $60 \times 220 \times 85$ elements. The permeability tensor is defined as a piecewise constant on each element. This coefficient is highly heterogeneous, exhibiting contrast of over 10^7; see Figure 9. We solve Darcy's equations for the velocity and pressure,

$$\begin{aligned} \begin{cases} \kappa \boldsymbol{u} + \nabla p = 0, \\ \nabla \cdot \boldsymbol{u} = 0, \end{cases} & \quad \text{in } \Omega. \\ \boldsymbol{u} \cdot \boldsymbol{n} = (1, 0, 0)^{\mathsf{T}} \cdot \boldsymbol{n} & \quad \text{on } \partial\Omega, \end{aligned}$$

where κ is inverse permeability matrix. Since pure Neumann boundary conditions are enforced for this problem, the Schur complement is singular, with a nullspace consisting of all constant functions. To ensure a convergent solver, every application of the approximate inverse $\widehat{\mathsf{S}}^{-1}$ is followed by an orthogonalization step.

We use this problem to perform a weak scaling study using $p = 4$ and $p = 6$ elements. With each refinement, the number of degrees of freedom is roughly doubled by solving on a finer mesh, and the problem is solved using twice the number of GPUs. With increasing mesh refinement the iteration counts remain roughly constant. The largest problem has roughly 7×10^8 unknowns, and is solved in about 10 seconds on 256 GPUs. For each of the cases considered, the solve time and iterations are similar between the $p = 4$ and $p = 6$ cases, illustrating the uniform convergence of the solver with respect to polynomial degree. The parallel scalability of the solver is largely consistent with similar results reported in [27, 36]. For the largest problem, the weak parallel efficiency of the solver is 31%, comparing favorably to the parallel efficiency of 25% for the algebraic hybridization solver and 7.5% for the ADS solver reported for similarly sized SPE10 problems in [16].

4.3. **Nonlinear radiation diffusion.** As a final test case, we consider the solution of time-dependent grey (one-group) nonlinear radiation diffusion equations. The governing equations are

$$\rho \frac{\partial e}{\partial t} = -c\sigma(aT_{\text{mat}}^4 - E) + Q, \tag{22}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot F = c\sigma(aT_{\text{mat}}^4 - E) + S, \tag{23}$$

$$F = -\frac{c}{3\sigma}\nabla E, \tag{24}$$

where e is the material specific internal energy, E is the radiation energy density, and F is the radiation flux, σ is the absorption opacity, c is the speed of light, a is the black body constant, and ρ is the material density. Q and S are given source terms. The material temperature T_{mat} is related to the internal energy e through the specific heat C_v. The energy unknowns e and E are taken in the L^2 finite element space W_h, and the radiation flux is taken in the $\boldsymbol{H}(\text{div})$ space $\boldsymbol{V}_h$.

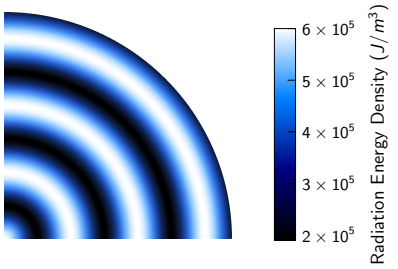
For this test case, we use the manufactured solution proposed in [12]. The equations are integrated in time implicitly; we use the three-stage L-stable singly diagonal implicit Runge–Kutta (SDIRK) method [3]. The resulting nonlinear algebraic equations are solved using an iterative scheme as follows. First, the radiation flux F is lagged, and a coupled nonlinear system for the material energy e and radiation energy E is solved. Then, the material energy e is fixed, and a Darcy-type linear system is solved for the radiation energy E and radiation flux F. This procedure is repeated until the full nonlinear residual has converged to the desired tolerance; for the time-accurate time steps considered presently, this method typically requires only one or two outer iterations to reach a relative tolerance of 10^{-12}.

For fixed radiation flux F, the (inner) nonlinear equations corresponding to (22) and (23) are solved element-by-element using Newton's method. The resulting Jacobian matrix takes the form of a 2×2 element-wise block diagonal system, which is well-preconditioned by a block Jacobi preconditioner with blocks of size 2×2. After solving for the energies e and E, the material energy e is fixed, and the coupled linear equations for E and F corresponding to (23) and (24) are solved; the solution of this system is often a bottleneck in radiation hydrodynamics simulations, emphasizing the importance of the development of efficient preconditioners [22, 31]. These equations take the form of a Darcy-type problem, which can be preconditioned using the block preconditioners described in this work.

The test problem in [12] is posed in spherical coordinates; presently, we solve this problem in Cartesian coordinates on a three-dimensional mesh with 884,736 hexahedral elements using polynomial degree $p = 4$. This problem configuration has 1.7×10^8 Raviart–Thomas degrees of freedom and 5.7×10^7 L^2 degrees of freedom, and is solved in parallel with 16 Nvidia V100 GPUs on four nodes of LLNL's *Lassen*. The equations are integrated until a final time of $0.1/\tau \approx 4.39\times 10^{-11}$, where $\tau \approx 2.28\times 10^9$ is the decay constant of the test problem. The time step is chosen to be $\Delta t = h/\tau$, and 10 implicit time steps (each with three Runge–Kutta stages) are taken. Each stage requires an outer nonlinear solve (with relative tolerance 10^{-12}), an inner nonlinear solve (with relative tolerance 10^{-8}), and a linear solver (with relative tolerance 10^{-8}). At the final time, the relative L^2 error was 7.72×10^{-8} in the material energy and 5.25×10^{-7} in the radiation energy. Each Runge–Kutta stage required two outer solver iterations and four inner Newton iterations. In total, 60 saddle-point linear solves were performed, and on average the linear solver converged in 84 MINRES iterations. The timing results and iteration counts are presented in the right panel of Figure 10. Consistent with results reported in [22], approximately 92% of the total runtime for this problem was spent in the linear solver.

5. Conclusions

In this work, we have developed solvers for grad-div and Darcy-type problems in $\boldsymbol{H}(\text{div})$ based on block preconditioning for the associated saddle-point system. Properties of the interpolation–histopolation high-order basis result in off-diagonal blocks with enhanced sparsity. The Schur complement of the block system can be approximated by an M-matrix with sparsity independent of the polynomial degree; such systems are amenable to preconditioning with scalable algebraic multigrid methods. The $(1,1)$-block can be well-approximated by a diagonal matrix. The resulting block preconditioners result in uniform convergence, independent of mesh size and polynomial degree. These solvers are amenable to GPU acceleration; high-throughput algorithms are developed and benchmarked. The performance of the solvers is demonstrated on



Alg. Component	Time (s)	%
Nonlinear residual	8.22	4.45%
Inner Newton solver (2 iterations per solve)	7.15	3.87%
Linear solver (84 iterations per solve)	169.39	91.68%

FIGURE 10. Left: snapshot of radiation energy density from radiation diffusion test case. Right: timing results and iteration counts on 16 V100 GPUs.

several benchmark problems, including the "crooked-pipe" grad-div problem, the SPE10 benchmark from reservoir simulation, and nonlinear radiation-diffusion.

6. Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-JRNL-848035). W. Pazner and Tz. Kolev were partially supported by the LLNL-LDRD Program under Project No. 20-ERD-002.

References

[1] A. Abdelfattah et al. "GPU algorithms for efficient exascale discretizations". In: *Parallel Computing* 108 (2021), p. 102841. DOI: 10.1016/j.parco.2021.102841.

[2] A. Abdelfattah et al. "Performance, Design, and Autotuning of Batched GEMM for GPUs". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 21–38.

[3] R. Alexander. "Diagonally Implicit Runge–Kutta Methods for Stiff ODEs". In: *SIAM Journal on Numerical Analysis* 14.6 (1977), pp. 1006–1021. DOI: 10.1137/0714068.

[4] R. Anderson et al. "MFEM: a modular finite element methods library". In: *Computers & Mathematics with Applications* (2020). DOI: 10.1016/j.camwa.2020.06.009.

[5] D. N. Arnold, R. S. Falk, and R. Winther. "Preconditioning in $H(\mathrm{div})$ and applications". In: *Mathematics of Computation* 66.219 (1997), pp. 957–985. DOI: 10.1090/s0025-5718-97-00826-0.

[6] D. N. Arnold, R. S. Falk, and R. Winther. "Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$". In: *Numerische Mathematik* 85.2 (2000), pp. 197–217. DOI: 10.1007/pl00005386.

[7] E. Aulisa, S. Calandrini, and G. Capodaglio. "FOV-equivalent block triangular preconditioners for generalized saddle-point problems". In: *Applied Mathematics Letters* 75 (2018), pp. 43–49.

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

[9] D. Boffi, F. Brezzi, and M. Fortin. *Mixed Finite Element Methods and Applications*. Vol. 44. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2013.

[10] J. H. Bramble and J. E. Pasciak. "A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems". In: *Mathematics of Computation* 50.181 (1988), pp. 1–17.

[11] P. D. Brubeck and P. E. Farrell. "Multigrid solvers for the de Rham complex with optimal complexity in polynomial degree". arXiv eprint 2211.14284. 2022.

[12] T. A. Brunner. *Development of a grey nonlinear thermal radiation diffusion verification problem*. Tech. rep. SAND2006-4030C. Sandia National Laboratories, 2006.

[13] T. A. Brunner et al. "Preserving spherical symmetry in axisymmetric coordinates for diffusion problems". In: *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013)*. 2013.

[14] M. A. Christie and M. J. Blunt. "Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques". In: *SPE Reservoir Evaluation & Engineering* 4.04 (2001), pp. 308–317. DOI: 10.2118/72469-pa.

[15] J. J. D. Croz and N. J. Higham. "Stability of methods for matrix inversion". In: *IMA Journal of Numerical Analysis* 12.1 (1992), pp. 1–19. DOI: 10.1093/imanum/12.1.1.

[16] V. Dobrev et al. "Algebraic hybridization and static condensation with application to scalable $H(\mathrm{div})$ preconditioning". In: *SIAM Journal on Scientific Computing* 41.3 (2019), B425–B447. DOI: 10.1137/17M1132562.

[17] V. Dobrev et al. *ECP Milestone Report: Identify initial kernels, bake-off problems (benchmarks) and miniapps (CEED-MS6)*. Tech. rep. LLNL-TR-734169. U.S. Department of Energy, 2017. DOI: 10.5281/zenodo.2542333.

[18] C. R. Dohrmann. "Spectral equivalence of low-order discretizations for high-order $H(\mathrm{curl})$ and $H(\mathrm{div})$ spaces". In: *SIAM Journal on Scientific Computing* 43.6 (2021), A3992–A4014.

[19] J. Dongarra et al. "The Design and Performance of Batched BLAS on Modern High-Performance Computing Systems". In: *Procedia Computer Science* 108 (2017), pp. 495–504. DOI: 10.1016/j.procs.2017.05.138.

[20] R. D. Falgout et al. "Porting *hypre* to heterogeneous computer architectures: strategies and experiences". In: *Parallel Computing* 108 (2021), p. 102840. DOI: 10.1016/j.parco.2021.102840.

[21] P. Fischer et al. "Scalability of high-performance PDE solvers". In: *The International Journal of High Performance Computing Applications* 34.5 (2020), pp. 562–586. DOI: 10.1177/1094342020915762.

[22] M. González et al. "Multigroup radiation hydrodynamics with flux-limited diffusion and adaptive mesh refinement". In: *Astronomy & Astrophysics* 578 (2015), A12. DOI: 10.1051/0004-6361/201525971.

[23] F. Graziani and J. LeBlanc. *The crooked pipe test problem*. Tech. rep. UCRL-MI-143393. Lawrence Livermore National Laboratory, 2000.

[24] V. E. Henson and U. M. Yang. "BoomerAMG: a parallel algebraic multigrid solver and preconditioner". In: *Applied Numerical Mathematics* 41.1 (2002), pp. 155–177. DOI: 10.1016/s0168-9274(01)00115-5.

[25] G. Kanschat, R. Lazarov, and Y. Mao. "Geometric multigrid for Darcy and Brinkman models of flows in highly heterogeneous porous media: a numerical study". In: *Journal of Computational and Applied Mathematics* 310 (2017), pp. 174–185. DOI: 10.1016/j.cam.2016.05.016.

[26] T. Kolev and W. Pazner. "Conservative and accurate solution transfer between high-order and low-order refined finite element spaces". In: *SIAM Journal on Scientific Computing* 44.1 (2022), A1–A27. DOI: 10.1137/21M1403916.

[27] T. Kolev et al. "Efficient exascale discretizations: high-order finite element methods". In: *The International Journal of High Performance Computing Applications* (2021), pp. 527–552. DOI: 10.1177/10943420211020803.

[28] T. V. Kolev and P. S. Vassilevski. "Parallel auxiliary space AMG solver for $H(\mathrm{div})$ problems". In: *SIAM Journal on Scientific Computing* 34.6 (2012), A3079–A3098.

[29] J. Kreeft, A. Palha, and M. Gerritsma. "Mimetic framework on curvilinear quadrilaterals of arbitrary order". arXiv:1111.4304. 2011.

[30] M. Kronbichler and K. Ljungkvist. "Multigrid for matrix-free high-order finite element computations on graphics processors". In: *ACM Transactions on Parallel Computing* 6.1 (2019), pp. 1–32. DOI: 10.1145/3322813.

[31] S. H. Langer, I. Karlin, and M. M. Marinak. "Performance Characteristics of HYDRA — A Multi-physics Simulation Code from LLNL". In: *High Performance Computing for Computational Science — VECPAR 2014*. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 173–181.

[32] J. W. Lottes and P. F. Fischer. "Hybrid multigrid/Schwarz algorithms for the spectral element method". In: *Journal of Scientific Computing* 24.1 (2005), pp. 45–78. DOI: 10.1007/s10915-004-4787-3.

[33] J. Melenk. "On condition numbers in *hp*-FEM with Gauss–Lobatto-based shape functions". In: *Journal of Computational and Applied Mathematics* 139.1 (2002), pp. 21–48. DOI: 10.1016/s0377-0427(01)00391-0.

[34] S. A. Orszag. "Spectral methods for problems in complex geometries". In: *Journal of Computational Physics* 37.1 (1980), pp. 70–92.

[35] W. Pazner. "Efficient low-order refined preconditioners for high-order matrix-free continuous and discontinuous Galerkin methods". In: *SIAM Journal on Scientific Computing* 42.5 (2020), A3055–A3083. DOI: 10.1137/19m1282052.

[36] W. Pazner, T. Kolev, and J.-S. Camier. "End-to-end GPU acceleration of low-order-refined preconditioning for high-order finite element discretizations". arXiv eprint 2210.12253. (Submitted for publication). 2022.

[37] W. Pazner, T. Kolev, and C. R. Dohrmann. "Low-order preconditioning for the high-order finite element de Rham complex". In: *SIAM Journal on Scientific Computing* 45.2 (2023), A675–A702. DOI: 10.1137/22M1486534.

[38] W. Pazner and P.-O. Persson. "Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods". In: *Journal of Computational Physics* 354 (2018), pp. 344–369.

[39] J. Pestana and A. J. Wathen. "Natural preconditioning and iterative methods for saddle point systems". In: *SIAM Review* 57.1 (2015), pp. 71–91.

[40] E. M. Rønquist and A. T. Patera. "Spectral element multigrid. I. Formulation and numerical results". In: *Journal of Scientific Computing* 2.4 (1987), pp. 389–406. DOI: 10.1007/bf01061297.

[41] S. A. Teukolsky. "Short note on the mass matrix for Gauss–Lobatto grid points". In: *Journal of Computational Physics* 283 (2015), pp. 408–413.

[42] O. Villa et al. "Power/Performance Trade-Offs of Small Batched LU Based Solvers on GPUs". In: *Euro-Par 2013 Parallel Processing*. Springer Berlin Heidelberg, 2013, pp. 813–825.

[43] W. Zulehner. "Nonstandard norms and robust estimates for saddle point problems". In: *SIAM Journal on Matrix Analysis and Applications* 32.2 (2011), pp. 536–560.