

Preconditioning and Reduced-Order Modeling of Navier-Stokes Equations in Complex Porous Microstructures

Kangan Li^a, Yashar Mehmani^a

^aEnergy and Mineral Engineering Department, The Pennsylvania State University, University Park, Pennsylvania 16802

Abstract

We aim to solve the incompressible Navier-Stokes equations within the complex microstructure of a porous material. Discretizing the equations on a fine grid using a staggered (e.g., marker-and-cell, mixed FEM) scheme results in a nonlinear residual. Adopting the Newton method, a linear system must be solved at each iteration, which is large, ill-conditioned, and has a saddle-point structure. This demands an iterative (e.g., Krylov) solver, that requires preconditioning to ensure rapid convergence. We propose two monolithic *algebraic* preconditioners, $a\text{PLMM}_{\text{NS}}$ and $a\text{PNM}_{\text{NS}}$, that are generalizations of previously proposed forms by the authors for the Stokes equations ($a\text{PLMM}_S$ and $a\text{PNM}_S$). The former is based on the pore-level multiscale method (PLMM) and the latter on the pore network model (PNM), both successful approximate solvers. We also formulate faster-converging geometric preconditioners $g\text{PLMM}$ and $g\text{PNM}$, which impose $\partial_n \mathbf{u} = 0$ (zero normal-gradient of velocity) exactly at subdomain interfaces. Finally, we propose an accurate coarse-scale solver for the steady-state Navier-Stokes equations based on $g\text{PLMM}$, capable of computing approximate solutions orders of magnitude faster. We benchmark our preconditioners against state-of-the-art block preconditioners and show $g\text{PLMM}$ is the best-performing one, followed closely by $a\text{PLMM}_S$ for steady-state flow and $a\text{PLMM}_{\text{NS}}$ for transient flow. All preconditioners can be built and applied on parallel machines.

Keywords: Navier-Stokes equations, Porous media, Multiscale method, Preconditioning, Krylov solver

1. Introduction

Porous media span a wide range of natural (e.g., soil, rock) and engineered (e.g., foamed metal, ceramic) materials with extreme microstructural complexity [1, 2]. High-resolution X-ray μCT imaging [3] is often used to capture this complexity, for subsequent use in pore-scale modeling. Here, we concern ourselves with the Navier-Stokes equations:

$$\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) - \mu \Delta \mathbf{u} + \nabla p = \rho \mathbf{b} \quad (1a)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (1b)$$

describing the flow of a single-phase, incompressible, and Newtonian fluid through the void space Ω of a porous sample. Eqs. 1a and 1b denote momentum and mass conservation, respectively. Variables $\mathbf{u}$, p , ρ , μ , and $\mathbf{b}$ represent velocity, pressure, density, viscosity, and body force per unit mass, respectively. The symbols ∇ , Δ , and $\nabla \cdot$ are the gradient, Laplacian, and divergence operators. Direct numerical simulation (DNS) methods discretize Eq. 1 on a fine grid over Ω . Popular choices are staggered finite volume (FVM) [4], marker and cell (MAC) finite difference [5], and mixed finite element (MFEM) [6, 7], due to their flux conservation properties essential to subsequent solute transport and two-phase flow simulations [8]. High-fidelity solutions obtained from these methods help characterize geologic sites for CO_2 sequestration [9], underground H_2 storage [10], and geothermal energy extraction [11], and design new porous components for fuel cells [12], electrolyzers [13], and heat exchangers and thermal insulators [14].

*Corresponding author: Yashar Mehmani. Email: yzm5192@psu.edu

Email addresses: kb15610@psu.edu (Kangan Li), yzm5192@psu.edu (Yashar Mehmani)

Algebraic systems arising from the above DNS methods are nonlinear, due to the inertia term $\mathbf{u} \cdot \nabla \mathbf{u}$, and are often solved with Newton's method. Linearization leads to the following saddle-point structure for the Newton update:

$$J^k \delta^k = -r^k \quad \Rightarrow \quad \begin{bmatrix} F^k & G^\top \\ G & O \end{bmatrix} \begin{bmatrix} \delta_u^k \\ \delta_p^k \end{bmatrix} = - \begin{bmatrix} r_u^k \\ r_p^k \end{bmatrix} \quad \Rightarrow \quad \hat{A} \hat{x} = \hat{b} \quad (2)$$

where r^k is the residual vector at iteration k and J^k is the corresponding Jacobian. The Newton update δ^k consists of a velocity δ_u^k and a pressure δ_p^k component. The next iterate is obtained from $\mathbf{u}^{k+1} = \mathbf{u}^k + \delta_u^k$ and $p^{k+1} = p^k + \delta_p^k$. Residual vectors r_u^k and r_p^k correspond to discretized momentum and mass balance in Eq.1, respectively. For simplicity, we denote the linearized system by $\hat{A} \hat{x} = \hat{b}$, which is often very large. Solving it requires iterative (e.g., Krylov) solvers, that depend on effective preconditioning to converge rapidly [15]. Our main goal is to develop such preconditioners.

The most common preconditioners are block-triangular [16] of the following form (dropping the index k):

$$M_B = \begin{bmatrix} F & G^\top \\ O & X \end{bmatrix} \quad \Rightarrow \quad M_B^{-1} = \begin{bmatrix} F^{-1} & -F^{-1}G^\top X^{-1} \\ O & X^{-1} \end{bmatrix} \quad \Rightarrow \quad \hat{A} M_B^{-1} \hat{x} = \hat{b} \quad (3)$$

applied here as a right-preconditioner, M_B . If X equals the Schur complement matrix $-GF^{-1}G^\top$, convergence is in one step. Note G is the discrete divergence operator, and its transpose the discrete gradient. The idea behind all block preconditioners is to *approximate* X^{-1} , or more precisely its action on a given vector, because direct computation is equivalent to solving the system. Thus, left-multiplying a vector by M_B^{-1} requires solving two smaller systems, one involving the coefficient matrix F and another involving X . Since these are elliptic, algebraic multigrid (AMG) [17, 18] is a preferred choice [19]. Approximate candidates for X^{-1} abound in the literature [7, 16, 20–23], among which the *scaled*-BFBt approximation of [23] was found to exhibit robust performance in complex geometries [24]:

$$X^{-1} = -(GD_F^{-1}G^\top)^{-1}GD_F^{-1}FD_F^{-1}G^\top(GD_F^{-1}G^\top)^{-1} \quad (4)$$

where D_F is the diagonal matrix comprised of the diagonal entries of F .

Unfortunately, using AMG to solve block systems associated with F and X , henceforth referred to as *bAMG*, was found by the authors [24] to result in GMRES stagnation for Stokes flow through complex porous geometries. The prefix *b* here stands for “block” preconditioner. A far more robust alternative to AMG was proposed by [24] based on the pore-level multiscale method (PLMM) [25] and the pore-network model [26], equipped with better prolongation/restriction matrices. We call these *bPLMM* and *bPNM*, which exhibit faster convergence and no stagnation. Even with these improvements, block-preconditioners remain prohibitively slow for practical pore-scale simulation [24].

To address this gap, monolithic preconditioners based on PLMM and PNM were formulated in [24] for the Stokes equations, which are direct algebraic translations of the geometric models these algorithms represent [25, 26]. We denote them by *aPLMM_S* and *aPNM_S*, where subscript “S” emphasizes the preconditioners are built from the discrete Stokes system. The term “monolithic” implies that velocity and pressure unknowns are solved simultaneously, without splitting the system into blocks. Both *aPLMM_S* and *aPNM_S* are two-level, consisting of a coarse preconditioner M_G and a smoother M_L . The former encapsulates the physical insights behind the original PLMM and PNM, and the latter is an additive-Schwarz smoother to wipe out high-frequency errors. A key feature of M_G is that it doubles as an approximate Stokes solver, where $\hat{x}_{approx} = M_G^{-1} \hat{b}$ is usable in many applications. L_2 -errors with the M_G of *aPLMM_S* are $< 2\%$ for velocity and $< 1\%$ for pressure, and with *aPNM_S* they are $< 6\%$ for velocity and $< 5\%$ for pressure [24].

Our first goal here is to extend *aPLMM_S* and *aPNM_S* to the Navier-Stokes equations (Eq.1), referring to the resulting preconditioners as *aPLMM_{NS}* and *aPNM_{NS}*. In PLMM and PNM, Ω is decomposed into subdomains called *primary* and *dual grids*. In PLMM, primary grids coincide with geometric enlargements of Ω (or pores), and dual grids (covering primary-grid interfaces) coincide with geometric constrictions (or throats). In PNM, the reverse is true: primary grids are throats and dual grids (and interfaces) are pores. Decompositions are achieved via the watershed segmentation algorithm [27]. Shape and correction vectors are built on each primary grid by solving local systems subject to closure boundary conditions (BCs) at interfaces between primary grids. These BCs are $p = \text{const}$ and $\partial_n \mathbf{u} = \mathbf{0}$, with ∂_n denoting differentiation normal to the interface. For Stokes flow, the BCs are known to be very accurate for PLMM [25], due to converging-diverging flow patterns near throats. In constructing *aPLMM_S* and *aPNM_S* for Stokes equations, closure BCs are easily imposed via purely algebraic operations [24]. We show this is not the case for the

Navier-Stokes equations, because $\partial_n \mathbf{u} = \mathbf{0}$ can only be imposed approximately in building $a\text{PLMM}_{\text{NS}}$ and $a\text{PNM}_{\text{NS}}$.

To impose $\partial_n \mathbf{u} = \mathbf{0}$ exactly in a preconditioner, we show the residual r^k in Eq.2 must be modified by removing the inertia term $\mathbf{u} \cdot \nabla \mathbf{u}$ in a small neighborhood of the primary-grid interfaces. This yields the geometric (not algebraic) preconditioners $g\text{PLMM}$ and $g\text{PNM}$, where access to the code assembling r^k is required. Despite this need for mildly intrusive code access, we find $g\text{PLMM}$ and $g\text{PNM}$ outperform $a\text{PLMM}_{\text{NS}}$ and $a\text{PNM}_{\text{NS}}$. We test all preconditioners ($a\text{PLMM}_{\text{NS}}$, $a\text{PNM}_{\text{NS}}$, $g\text{PLMM}$, $g\text{PNM}$) in GMRES for a range of 2D/3D porous geometries and benchmark them against state-of-the-art block ($b\text{AMG}$, $b\text{PLMM}$, $b\text{PNM}$) and Stokes-based ($a\text{PLMM}_S$, and $a\text{PNM}_S$) preconditioners. In all cases, $g\text{PLMM}$ emerges as the best-performing preconditioner. If algebraic preconditioning is desired, $a\text{PLMM}_S$ and $a\text{PLMM}_{\text{NS}}$ are close seconds for the steady-state and unsteady-state Navier-Stokes equations, respectively.

Other monolithic preconditioners for Navier-Stokes equations exist, notably the generalized Dryja-Smith-Widlund (GDSW) [28, 29], but place no emphasis on the importance of decomposition (by watershed) and closure BCs. We show, by comparing $a\text{PLMM}_{\text{NS}}$ and $a\text{PNM}_{\text{NS}}$ for example, that these choices play outside roles in Krylov convergence. A further demonstration of this fact was made in [30] for simpler elliptic equations arising from linear elasticity.

Our second contribution is a coarse-scale solver for the steady-state Navier-Stokes equations based on $g\text{PLMM}$. The algorithm yields an approximate solution iteratively, unlike the single-step $\hat{x}_{\text{apr}} = \mathbf{M}_G^{-1} \hat{b}$ for Stokes flow. Iterations consist of Newton updates on coarse (pressure) unknowns only, with *no* recourse to the fine grid. The coarse solution is mapped onto the fine grid using a prolongation operator, which is not feasible with competing reduced-order models based on PNM [31–33]. Computations are an order of magnitude faster than required for the exact solution.

The paper’s outline follows: Section 2 describes the problem we aim to solve and the fine-grid solver employed. Section 3 details the formulations of the algebraic preconditioners $a\text{PLMM}_{\text{NS}}$ and $a\text{PNM}_{\text{NS}}$. Section 4 formulates the geometric preconditioners $g\text{PLMM}$ and $g\text{PNM}$. The coarse solver for steady-state Navier Stokes is detailed in Section 5. In Section 6, we describe the set of 2D/3D domains considered for testing the preconditioners and the coarse solver. Results are presented in Sections 7–8 and discussed in Section 9. The paper concludes with takeaways in Section 10.

2. Problem and fine-grid solver

We aim to solve Eq.1 on the complex void space Ω of a porous sample (white region in Fig.1a), which is often captured by a pore-scale image (e.g., X-ray μCT). Eq.1 requires an initial condition (IC) $\mathbf{u}(t = 0) = \mathbf{u}_0$ and several boundary conditions (BCs). We impose no-slip ($\mathbf{u} = \mathbf{0}$ and $\partial p_n = 0$) on the fluid-solid interface Γ_w (white-black boundary in Fig.1a) and inlet/outlet BCs (constant pressure and $\partial_n \mathbf{u} = 0$) on the left/right external boundaries of Ω , respectively. The latter results in flow from left to right. Here, ∂_n denotes differentiation in the direction normal to the corresponding boundary. At moderate to low Reynolds numbers $Re = \rho U l / \mu$ (where U and l are characteristic velocity and length), the equations admit a steady-state laminar solution without flow instabilities, allowing one to drop the $\mathbf{u}_t$ term. At even lower Re , flow becomes creeping, reducing Eq.1 to the Stokes equations by further dropping $\mathbf{u} \cdot \nabla \mathbf{u}$.

We discretize Eq.1 on a Cartesian *fine grid* that conforms to, or is an integer fraction of, the image pixels comprising Ω . We use a marker and cell (MAC) finite difference method, with velocity unknowns defined at cell faces and pressure unknowns at cell centers, as shown in Fig.2c. We remark that our proposed preconditioners in later sections are by no means limited to Cartesian grids or the MAC scheme. Other discretizations discussed in Section 1 (e.g., FVM, MFEM), where velocity and pressure are arranged in a spatially staggered fashion, are equally possible and would benefit from our preconditioners. The resulting discrete system, given by Eq.2, is nonlinear and is solved with Newton’s method. At each iteration of Newton, a linearized system of the form $\hat{\mathbf{A}} \hat{\mathbf{x}} = \hat{\mathbf{b}}$ is solved.

The blocks $\mathbf{F}^k$, $\mathbf{G}^\top$, $\mathbf{G}$ of $\hat{\mathbf{A}}$ in Eq.2 are discrete forms of the $\rho \partial_t + \rho \mathcal{L}(\mathbf{u} \cdot \nabla \mathbf{u}) - \mu \Delta$, gradient (∇), and divergence ($\nabla \cdot$) operators, respectively. The symbol $\mathcal{L}(\cdot)$ denotes the argument is linearized around the previous iterate, which is why only $\mathbf{F}^k$ has superscript k . For Stokes flow, $\mathbf{F}$ is symmetric and the residual r^k is linear, allowing Newton to converge in one step. For Navier-Stokes flow, neither symmetry nor linearity are present. For practical domain sizes of real porous microstructures, $\hat{\mathbf{A}}$ is large and poorly conditioned, caused by the geometric complexity of Ω and the small fine-grids needed to resolve it. This demands iterative solvers like GMRES, which require preconditioning.

3. Algebraic monolithic preconditioner

We extend the fully-algebraic monolithic preconditioners $a\text{PLMM}_S$ and $a\text{PNM}_S$ originally designed for the Stokes equations [24] to the Navier-Stokes equations. They are based on the pore-level multiscale method (PLMM) and the

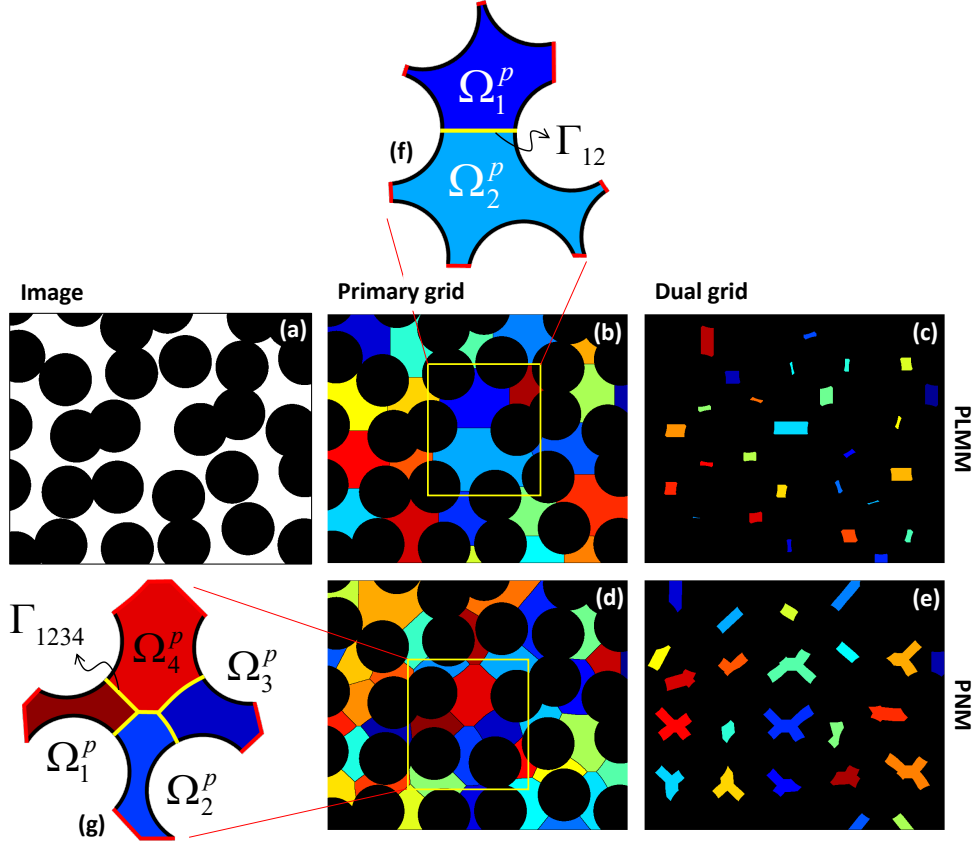


Figure 1: Schematic of a pore-scale image and its decomposition into primary grids and dual grids in the PLMM and PNM preconditioner. (a) The white regions of the image represent the void space and black regions represent the solid phase. The Navier-Stokes equations are solved in the void space. (b, d) Primary grids (or subdomains) obtained from PLMM (b) and PNM (d) decompositions. (c, e) Corresponding dual grids that overlap the contact interfaces shared between primary grids. (f, g) Each basis function is associated with one contact interface (yellow). In PLMM, each contact interface is shared by only two adjacent primary grids. While in PNM, each contact interface is shared among multiple primary grids.

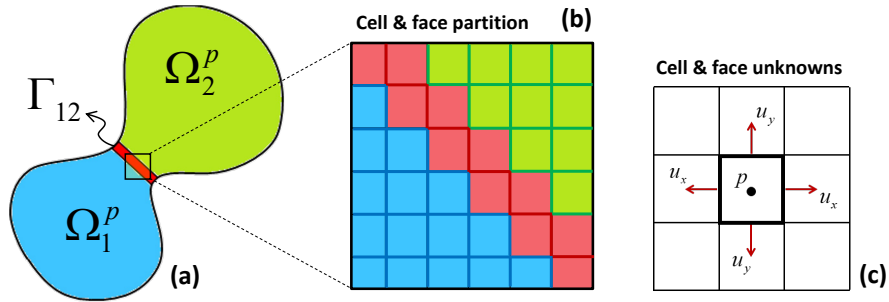


Figure 2: (a) Two primary grids, Ω_1^p and Ω_2^p , separated by a contact interface, Γ_{12} . (b) Partitioning of the cells and faces of the fine grid among the two primary grids and the contact interface is shown. Light/dark blue cells/faces belong to Ω_1^p , light/dark green cells/faces belong to Ω_2^p , and light/dark red cells/faces belong to Γ_{12} . (c) The cell unknowns correspond to pressure, p , and the face unknowns correspond to velocity, $\mathbf{u}$.

the pore network model (PNM). We refer to the extended preconditioners as $a\text{PLMM}_{\text{NS}}$ and $a\text{PNM}_{\text{NS}}$, where prefix “a” denotes “algebraic.” Both are two-level with a global (coarse) preconditioner M_G and a local (fine) smoother M_L . Section 3.1 outlines the overall structure of the two-level preconditioners, combining M_G and M_L . Since the mathematical structures of $a\text{PLMM}_{\text{NS}}$ and $a\text{PNM}_{\text{NS}}$ are identical, we detail $a\text{PLMM}_{\text{NS}}$ first in Section 3.2 then highlight

key differences of $aPNM_{NS}$ in Section 3.3. In Section 3.2.1, we describe the domain decomposition central to M_G and M_L . Section 3.2.2 formulates M_G , and Section 3.2.3 reviews the additive-Schwarz smoother M_L by [24]. Since $aPLMM_{NS}$ and $aPNM_{NS}$ reduce to $aPLMM_S$ and $aPNM_S$ for Stokes flow, we drop subscripts “NS” and “S” below.

3.1. Overall structure

The monolithic $aPLMM$ and $aPNM$ preconditioners M are formulated as:

$$M^{-1} = M_G^{-1} + M_L^{-1}(I - \hat{A}M_G^{-1}) \quad (5a)$$

$$M_L^{-1} = \sum_{i=1}^{n_{st}} M_L^{-1}(I - \hat{A}M_L^{-1})^{i-1} \quad (5b)$$

where the global preconditioner M_G and the smoother M_L are combined multiplicatively, with the former attenuating low-frequency errors and the latter high-frequency errors. The local smoother itself is a multiplicative (n_{st} -stage) application of a *base smoother* M_L , which can be any black-box preconditioner (e.g., block Gauss-Seidel) or a custom smoother compatible with M_G . Our numerical tests show that the additive-Schwarz smoother M_{pd} from [24] (reviewed in Section 3.2.3) provides superior performance. Therefore, we adopt $M_L = M_{pd}$ with $n_{st} = 1$ throughout this work.

3.2. Monolithic PLMM preconditioner

3.2.1. Domain decomposition

To build M_G and M_L , the domain Ω is first decomposed into N^p non-overlapping subdomains Ω_i^p , called *primary grids*. The decomposition uses a modified watershed segmentation algorithm [34] that operates on a binary-image representation of Ω , as shown in Fig.1b for the pore-scale image in Fig.1a. Primary grids appear as randomly colored regions. The interfaces between adjacent primary grids, Γ_{ij} , are called *contact interfaces* (thick yellow line in Fig.1f). Watershed segmentation ensures that Ω_i^p correspond to local geometric enlargements of Ω and Γ_{ij} to local constrictions. We also construct N^d complementary *dual grids* Ω_i^d that overlap contact interfaces and cover a thin region around them (see Fig.1c). Dual grids are built by successive morphological dilations of the pixels comprising each contact interface [24, 25]. By construction, N^d equals the number of contact interfaces N^c . Dual grids are used only by M_L to remove high-frequency errors concentrated near Γ_{ij} after each application of M_G . Notice the union of Ω_i^d is much smaller than Ω . Unlike the PLMM algorithm of [25], where overlapping Ω_i^d are merged (sometimes creating large sample-spanning regions with costly local problems), here Ω_i^d are allowed to overlap, avoiding this computational drawback. Primary and dual grids are collectively termed *coarse grids* when the distinction is unimportant.

3.2.2. Global preconditioner

In iterative solvers, preconditioning means solving systems like $\hat{A}\hat{x} = \hat{b}$ *approximately* for $\hat{x}$ given $\hat{b}$. The proposed M_G is presented first as a recipe for computing an approximate solution, involving three operations called *permutation*, *reduction*, and *prolongation*. These operations parallel those of [24] for the Stokes equations, with key differences highlighted below. We then provide a compact representation of M_G^{-1} as the composition of matrices and operators.

Permutation. The goal here is to reshuffle the rows and columns of $\hat{A}$ such that the fine-grid entires associated with each primary grid, Ω_i^p , and each contact interface Γ_{ij} , are grouped together by applying a permutation matrix W to the linear system in Eq.2. Consider the simple domain Ω depicted in Fig.2a, which is decomposed into two primary grids Ω_1^p and Ω_2^p , sharing the contact interface Γ_{12} . With reference to Fig.2b, fine-grid *cells* comprising Ω are partitioned into those that belong to: (1) Ω_1^p (light blue); (2) Ω_2^p (light green); and (3) Γ_{12} (light red). Here, Γ_{12} is defined by the thinnest 4-connected collection of pixels in 2D (and 6-connected in 3D) that topologically separate the two primary grids. Other definitions of Γ_{12} may be used without loss of generality. A similar partitioning of the fine-grid *faces* is performed as follows: (1) faces flanked by at least one cell from Ω_1^p belong to Ω_1^p (dark blue); (2) faces flanked by at least one cell from Ω_2^p belong to Ω_2^p (dark green); and (3) faces flanked by cells neither in Ω_1^p nor Ω_2^p belong to Γ_{12} (dark red). Given this labeling of cells and faces, the matrix W is constructed as follows:

$$\underbrace{W^T \hat{A} W}_A \underbrace{W^T \hat{x}}_x = \underbrace{W^T \hat{b}}_b \quad \Rightarrow \quad Ax = b \quad (6a)$$

with the block structure below for A , b , and x :

$$A = \begin{bmatrix} A_p^p & A_c^p & A_f^p \\ A_p^c & A_c^c & A_f^c \\ A_p^f & A_c^f & A_f^f \end{bmatrix} \quad b = \begin{bmatrix} b^p \\ b^c \\ b^f \end{bmatrix} \quad x = \begin{bmatrix} x^p \\ x^c \\ x^f \end{bmatrix} \quad (6b)$$

We remark that the matrix W is unitary and satisfies $WW^\top = I$.

The super/subscript p in each block denotes the *cell and face* unknowns/residuals belonging to the primary grids. The super/subscript c and f denote the *cell only* and *face only* unknowns/residuals belonging to the contact interfaces, respectively. Recall that the cell and face unknowns correspond to pressure and velocity, respectively (Fig.2c).

Reduction. The goal of reduction is to introduce *closure or localization assumptions* that simplify and effectively decouple the permuted system in Eq.6. For the Stokes equations, the steps outlined below enforce the closure BCs $p = \text{const}$ and $\partial_n \mathbf{u} = \mathbf{0}$ on each contact interface. In [25], a local analysis was performed, demonstrating these BCs are nearly optimal for Stokes flow based on arguments that streamlines exhibit a converging-diverging pattern at geometric constrictions (Γ_{12} in Fig.1b). For the Navier-Stokes equations, we adopt the same reduction steps but with two caveats. The velocity closure no longer corresponds to $\partial_n \mathbf{u} = \mathbf{0}$ due to the asymmetric upwind discretization of the inertia term in Eq.1a. Additionally, the suitability of these closures for Navier-Stokes flow is diminished given the more complex streamline patterns near geometric constrictions compared to Stokes flow, as we illustrate later.

To impose $\partial \mathbf{u} / \partial n \approx 0$ (where “ $\approx$ ” means “inexact”) at contact interfaces, we first approximate A as follows:

$$A \approx \begin{bmatrix} \tilde{A}_p^p & A_c^p & \mathbf{0} \\ A_p^c & \tilde{A}_c^c & \mathbf{0} \\ A_p^f & A_c^f & A_f^f \end{bmatrix} \quad \begin{aligned} \tilde{A}_p^p &= A_p^p + \text{diag}(\text{csum}(A_f^p)) \\ \tilde{A}_c^c &= A_c^c + \text{diag}(\text{csum}(A_f^c)) \end{aligned} \quad (7)$$

where the operation $\text{csum}(\cdot)$ sums all columns of its argument matrix to yield a single column vector. The operation $\text{diag}(\cdot)$ takes an input vector and creates a diagonal matrix whose diagonal entries coincide with those of the vector. Notice by imposing $\partial_n \mathbf{u} \approx \mathbf{0}$ in Eq.7, the face-only unknowns x^f defined at contact interfaces are now decoupled from the system. Therefore, we may henceforth focus on solving the smaller system $\tilde{A}\tilde{x} = \tilde{b}$ with:

$$\tilde{A} = \begin{bmatrix} \tilde{A}_p^p & A_c^p \\ A_p^c & \tilde{A}_c^c \end{bmatrix} \quad \tilde{b} = \begin{bmatrix} b^p \\ b^c \end{bmatrix} \quad \tilde{x} = \begin{bmatrix} x^p \\ x^c \end{bmatrix} \quad (8)$$

after which x^f can be obtained by back substitution (this is fast because A_f^f is a very small):

$$x^f = (A_f^f)^{-1} (b^f - A_p^f x^p - A_c^f x^c) \quad (9)$$

Focusing next on Eq.8, each block consists of the following sub-blocks:

$$\tilde{A}_p^p = \begin{bmatrix} \tilde{A}_{p_1}^{p_1} & \cdots & \tilde{A}_{p_n}^{p_1} \\ \vdots & \ddots & \vdots \\ \tilde{A}_{p_1}^{p_n} & \cdots & \tilde{A}_{p_n}^{p_n} \end{bmatrix} \quad \tilde{A}_c^c = \begin{bmatrix} \tilde{A}_{c_1}^{c_1} & \cdots & \tilde{A}_{c_m}^{c_1} \\ \vdots & \ddots & \vdots \\ \tilde{A}_{c_1}^{c_m} & \cdots & \tilde{A}_{c_m}^{c_m} \end{bmatrix} \quad A_c^p = \begin{bmatrix} A_{c_1}^{p_1} & \cdots & A_{c_m}^{p_1} \\ \vdots & \ddots & \vdots \\ A_{c_1}^{p_n} & \cdots & A_{c_m}^{p_n} \end{bmatrix} \quad A_p^c = \begin{bmatrix} A_{p_1}^{c_1} & \cdots & A_{p_n}^{c_1} \\ \vdots & \ddots & \vdots \\ A_{p_1}^{c_m} & \cdots & A_{p_n}^{c_m} \end{bmatrix} \quad (10a)$$

$$b^p = \begin{bmatrix} b^{p_1} \\ \vdots \\ b^{p_n} \end{bmatrix} \quad b^c = \begin{bmatrix} b^{c_1} \\ \vdots \\ b^{c_m} \end{bmatrix} \quad x^p = \begin{bmatrix} x^{p_1} \\ \vdots \\ x^{p_n} \end{bmatrix} \quad x^c = \begin{bmatrix} x^{c_1} \\ \vdots \\ x^{c_m} \end{bmatrix} \quad (10b)$$

where $n = N^p$ and $m = N^c$, with N^p and N^c denoting the total number of primary grids and contact interfaces in Ω . Each sub-block corresponds to entries associated with either a primary grid, a contact interface, or a coupling between

the two. To complete the imposition of $\partial_n \mathbf{u} \approx \mathbf{0}$, we further simplify the $\tilde{\mathbf{A}}_p^p$ sub-block as follows:

$$\tilde{\mathbf{A}}_p^p \approx \bar{\mathbf{A}}_p^p = \begin{bmatrix} \bar{\mathbf{A}}_{p_1}^{p_1} & & \mathbf{O} \\ & \ddots & \\ \mathbf{O} & & \bar{\mathbf{A}}_{p_n}^{p_n} \end{bmatrix} \quad \bar{\mathbf{A}}_{p_i}^{p_i} = \tilde{\mathbf{A}}_{p_i}^{p_i} + \text{diag} \left(\sum_{j \neq i} \text{csum}(\tilde{\mathbf{A}}_{p_j}^{p_i}) \right) \quad (11)$$

where $\bar{\mathbf{A}}_p^p$ is block-diagonal matrix with the entries of each primary grid fully decoupled from those of other primary grids. Similar to Eq.7, the off-diagonal blocks of each block-row in $\tilde{\mathbf{A}}_p^p$ are column-summed and added to its main diagonal entries. We are now left with solving the simpler system $\bar{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$, with:

$$\bar{\mathbf{A}} = \begin{bmatrix} \bar{\mathbf{A}}_p^p & \mathbf{A}_c^p \\ \mathbf{A}_p^c & \tilde{\mathbf{A}}_c^c \end{bmatrix} \quad (12)$$

To enforce $p = \text{const}$ at contact interfaces, we introduce a *reduction matrix*, $\mathbf{Q}$:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_{N_f^p \times N_f^p} & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}^o \end{bmatrix} \quad \mathbf{Q}^o = \begin{bmatrix} \mathbf{1}^{c_1} & & \mathbf{O} \\ & \ddots & \\ \mathbf{O} & & \mathbf{1}^{c_m} \end{bmatrix}_{N_f^c \times N_f^c} \quad \mathbf{1}^{c_i} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N_f^{c_i} \times 1} \quad (13a)$$

such that when applied, yields the following *reduced system*:

$$\bar{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \quad \tilde{\mathbf{x}} \simeq \mathbf{Q}x_M \Rightarrow \underbrace{\mathbf{Q}^\top \bar{\mathbf{A}} \mathbf{Q}}_{\mathbf{A}_M} x_M = \underbrace{\mathbf{Q}^\top \tilde{\mathbf{b}}}_{b_M} \Rightarrow \mathbf{A}_M x_M = b_M \quad (13b)$$

In Eq.13a, N_f^p , N_f^c , and $N_f^{c_i}$ denote the number of fine-scale unknowns belonging to all primary grids, all contact interfaces, and the contact interface labeled c_i , respectively. We remark that N_f^p includes both velocity and pressure unknowns, whereas N_f^c and $N_f^{c_i}$ include only pressure unknowns. The reduction matrix $\mathbf{Q}$ is block-diagonal and is comprised of only 0 and 1 entries. Left-multiplying (or right-multiplying) a matrix by $\mathbf{Q}^\top$ (or $\mathbf{Q}$) has the effect of performing a row-sum (or column-sum) of all fine-scale entries associated with each contact interface separately. As the entries belonging to the interface c_i in $\bar{\mathbf{A}}$ correspond to only pressure unknowns, the closure $p = \text{const}$ over c_i is imposed by the column-sum. The row-sum ensures overall (“weak”) mass conservation across each interface.

The reduced system has now the following block structure:

$$\mathbf{A}_M = \begin{bmatrix} \bar{\mathbf{A}}_p^p & \bar{\mathbf{A}}_c^p \\ \bar{\mathbf{A}}_p^c & \bar{\mathbf{A}}_c^c \end{bmatrix} \quad b_M = \begin{bmatrix} b^p \\ b^c \end{bmatrix} \quad x_M = \begin{bmatrix} x^p \\ x^o \end{bmatrix} \quad (14)$$

where x^o (with length $N^c \times 1$) contains *coarse-scale* pressure unknowns defined at contact interfaces (one per interface).

Prolongation. The goal here is to build a *prolongation matrix*, $\mathbf{P}$, and a *restriction matrix*, $\mathbf{R}$, such that a coarse-scale problem can be formulated to solve for the interface pressures x^o . This is done as follows:

$$\mathbf{A}_M x_M = b_M, \quad x_M = \mathbf{P} \begin{bmatrix} x^o \\ y^o \end{bmatrix} \Rightarrow \underbrace{\mathbf{R} \mathbf{A}_M \mathbf{P}}_{\mathbf{A}^o} \begin{bmatrix} x^o \\ y^o \end{bmatrix} = \underbrace{\mathbf{R} b_M}_{b^o} \quad (15)$$

where y^o are auxiliary coarse-scale unknowns associated with the primary grids.

The prolongation matrix $\mathbf{P}$ has the following block structure:

$$\mathbf{P} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{I} & \mathbf{O} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} p_{c_1}^{p_1} & p_{c_2}^{p_1} & \cdots & p_{c_m}^{p_1} \\ p_{c_1}^{p_2} & p_{c_2}^{p_2} & \cdots & p_{c_m}^{p_2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{c_1}^{p_n} & p_{c_2}^{p_n} & \cdots & p_{c_m}^{p_n} \end{bmatrix}_{N_f^p \times N_f^c} \quad \mathbf{C} = \begin{bmatrix} c^{p_1} & & & \mathbf{O} \\ & c^{p_2} & & \\ & & \ddots & \\ \mathbf{O} & & & c^{p_n} \end{bmatrix}_{N_f^p \times N^p} \quad (16a)$$

where $p_{c_j}^{p_i}$ and c^{p_i} are called *shape* and *correction* vectors, respectively.

The shape vectors are defined as follows:

$$p_{c_j}^{p_i} = \begin{cases} -(\bar{A}_{p_i}^{p_i})^{-1} R_p^{p_i} \bar{A}_c^p e_c^{c_j}, & c_j \in C^{p_i} \\ \mathbf{O}, & c_j \notin C^{p_i} \end{cases} \quad (16b)$$

$$e_c^{c_j} = [\delta_{1j}, \delta_{2j}, \dots, \delta_{m_j}]_{N^c \times 1}^\top \quad (16c)$$

The index set C^{p_i} contains all contact interfaces that intersect the boundary of the primary grid p_i . The Kronecker delta δ_{ab} is used to define the unit vector $e_c^{c_j}$, with 1 occupying its j^{th} entry and 0 elsewhere. The matrix $R_p^{p_i}$ is called a *contraction matrix* that restricts a fine-scale $N_f^p \times 1$ vector defined over the union of all primary grids (excluding the contact interfaces) to a fine-scale $N_f^{p_i} \times 1$ vector defined over the primary grid p_i only. More precisely:

$$R_p^{p_i} = [\Delta_{p_1}^{p_i}, \Delta_{p_2}^{p_i}, \dots, \Delta_{p_n}^{p_i}]_{N_f^{p_i} \times N_f^p} \quad \Delta_{p_j}^{p_i} = \begin{cases} I_{N_f^{p_i} \times N_f^{p_i}} & \text{if } i = j \\ \mathbf{O}_{N_f^{p_i} \times N_f^{p_j}} & \text{if } i \neq j \end{cases} \quad (16d)$$

Similar to N_f^p defined earlier, $N_f^{p_i}$ is the number of fine-scale unknowns that belong to primary grid p_i . From Eq.16b, it is obvious that the matrix B is very sparse, because only two shape vectors per column contain non-zero entries. These two shape vectors belong to the two primary grids sandwiching the contact interface associated with each column (Fig.1f). As we shall discuss in Section 3.3, this does not hold for the aPNM preconditioner (Fig.1g).

Returning back to Eq.16a, the correction vectors are defined as follows:

$$c^{p_i} = (\bar{A}_{p_i}^{p_i})^{-1} b_B^{p_i} \quad (16e)$$

where $b_B^{p_i}$ is a right-hand side (RHS) vector defined on primary grid p_i that accounts for any non-homogeneous BCs and source terms ($b \neq 0$) in Eq.1. This does *not* equal b^{p_i} in Eq.10b, which also includes non-zero entries associated with the PDE residual in the interior of Ω_i^p , specifically $b^{p_i} = b_B^{p_i} + b_I^{p_i}$. To compute $b_B^{p_i}$, we evaluate the residual of the steady-state form of Eq.1 (dropping the time derivative, or equivalently using zero velocity at the previous time step) at the origin (i.e., $r(\mathbf{u} = \mathbf{0}, p = 0)$), then subject the result to the same permutation in Eq.6. We remark that for the Stokes equations, where Eq.1 is linear, the discrete system $\hat{A}\hat{x} = \hat{b}$ yields a RHS vector $\hat{b}$ that contains only the effect of non-homogeneous BCs, resulting in $b^{p_i} = b_B^{p_i}$. Hence, no distinction between these two vectors was made in [24]. Here, we use $b_B^{p_i}$ instead of b^{p_i} in Eq.16e because extensive numerical tests revealed the latter leads to poor GMRES convergence by introducing high-frequency errors in c^{p_i} over primary grids.

Having completed the definition of P , we now define the restriction matrix in Eq.15 as:

$$R = \begin{bmatrix} \mathbf{O} & \mathbf{I} \\ \Pi(C^\top) & \mathbf{O} \end{bmatrix} \quad (17)$$

where $\mathbf{I}$ is the $N^c \times N^c$ identity matrix, and $\Pi(C^\top)$ is a matrix with 0 and 1 entries that reflects the sparsity pattern of $C^\top$ in Eq.16a (i.e., entries are 1 where $C^\top$ is non-zero, and 0 elsewhere). Left-multiplying A_M by R performs a row-sum over each primary grid while leaving the rows associated with contact interfaces intact. Combined with the earlier row-sum performed through the application of Q in Eq.13b over entries of each contact interface, mass conservation is imposed in a weak (integrated) sense within each primary grid and across each contact interface. Note that $R \neq P^\top$. The choice of the restriction matrix in Eq.17 is motivated by the fact that a Galerkin restriction ($R = P^\top$) leads to poor solver performance for (Navier-)Stokes equations discretized with MAC finite differences or similar schemes [24].

We note that constructing R is trivial, and constructing P is embarrassingly parallelizable. The latter follows because computing shape and correction vectors in Eqs.16b and 16e involves solving fully decoupled local systems on primary grids. Since only RHS vectors differ between systems, local LU-decomposition can accelerate computations.

Summary. The above steps constitute a recipe for computing the action of the global preconditioner, M_G^{-1} , on a

RHS vector $\hat{b}$. For notational simplicity, and ease of reference later, we express this action as:

$$\mathbf{M}_G^{-1} = \hat{\mathbf{P}}(\hat{\mathbf{R}}\hat{\mathbf{A}}\hat{\mathbf{P}})^{-1}\hat{\mathbf{R}} \quad (18)$$

where $\hat{\mathbf{P}}$ and $\hat{\mathbf{R}}$ are *effective* prolongation and restriction matrices, respectively, and $\hat{\mathbf{A}}$ a transformation of $\hat{\mathbf{A}}$:

$$\hat{\mathbf{P}} = \mathbf{W}\mathbf{Q}\mathbf{P} \quad (19a)$$

$$\hat{\mathbf{R}} = \mathbf{R}\mathbf{Q}^\top\mathbf{W}^\top \quad (19b)$$

$$\hat{\mathbf{A}} = \mathbf{W}\mathcal{C}(\mathbf{W}^\top\hat{\mathbf{A}}\mathbf{W})\mathbf{W}^\top \quad (19c)$$

Recall $\mathbf{W}$, $\mathbf{Q}$, $\mathbf{P}$, and $\mathbf{R}$ are the permutation (Eq.6a), reduction (Eq.13a), prolongation (Eq.16a), and restriction (Eq.17) matrices introduced earlier. The operator $\mathcal{C}(\cdot)$ is *linear* and encapsulates the column-sums performed in Eqs.7 and 11 to impose the velocity closure BC. The $\text{csum}(\cdot)$ operation in Eqs.7 and 11 does not admit a simple matrix representation. Left-multiplying a coarse-grid vector, like $[x^o, y^o]^\top$ in Eq.15, by $\hat{\mathbf{P}}$ maps it onto the fine grid. Left-multiplication by $\hat{\mathbf{R}}$ maps in the inverse direction. The above representation of $\mathbf{M}_G$ makes one of its key properties immediately clear:

Proposition 3.1. *The matrix $\mathbf{M}_G^{-1}\hat{\mathbf{A}}$ is not a projection, where $\mathbf{M}_G^{-1}$ is given by Eq.18 for the saddle-point Stokes or Navier-Stokes equations. This lies in contrast to $\mathbf{M}_G^{-1} = \hat{\mathbf{P}}(\hat{\mathbf{R}}\hat{\mathbf{A}}\hat{\mathbf{P}})^{-1}\hat{\mathbf{R}}$ for elliptic equations, where $\mathbf{M}_G^{-1}\hat{\mathbf{A}}$ is a projection.*

Proof. A simple substitution of Eqs.18-19 into $(\mathbf{M}_G^{-1}\hat{\mathbf{A}})^2$ shows that the product does not equal $\mathbf{M}_G^{-1}\hat{\mathbf{A}}$, unless $\mathcal{C}(\cdot)$ is the identity matrix. For elliptic equations, $\mathcal{C}(\cdot)$ is identity [35], but not for saddle-point equations studied herein. $\square$

Remark 1. We found empirically that $\mathbf{M}_G^{-1} = \hat{\mathbf{P}}(\hat{\mathbf{R}}\hat{\mathbf{A}}\hat{\mathbf{P}})^{-1}\hat{\mathbf{R}}$ performs almost identically to Eq.18 (notice the difference in using $\hat{\mathbf{A}}$ versus $\hat{\mathbf{A}}$). In other words, the column-sum operator $\mathcal{C}(\cdot)$ in Eq.19c can be replaced by the identity matrix. We adopt this formulation to produce all results in Section 8, because it saves the extra column-sum operation.

3.2.3. Local smoother

Application of the global preconditioner $\mathbf{M}_G$ leaves behind high-frequency errors concentrated near contact interfaces [24]. Therefore, a compatible local smoother $\mathbf{M}_L$ is needed to attenuate these errors. We adopt the *dual-primary* smoother¹ $\mathbf{M}_L = \mathbf{M}_{dp}$ of [24], which has exhibited superior performance over other black-box alternatives. The dual-primary smoother combines two additive-Schwarz (block-Jacobi) preconditioners multiplicatively:

$$\mathbf{M}_{dp}^{-1} = \mathbf{M}_d^{-1} + \mathbf{M}_p^{-1}(\mathbf{I} - \hat{\mathbf{A}}\mathbf{M}_d^{-1}) \quad (20)$$

where $\mathbf{M}_d^{-1}$ is applied first to eliminate errors within dual grids, then $\mathbf{M}_p^{-1}$ is applied to remove errors within primary grids. Both $\mathbf{M}_d$ and $\mathbf{M}_p$ have standard additive-Schwarz formulations [15]:

$$\mathbf{M}_p^{-1} = \sum_{i=1}^{N^p} \mathbf{E}_f^{p_i} \underbrace{(\mathbf{R}_f^{p_i} \hat{\mathbf{A}} \mathbf{E}_f^{p_i})^{-1}}_{\hat{\mathbf{A}}_{p_i}} \mathbf{R}_f^{p_i} \quad \mathbf{M}_d^{-1} = \sum_{i=1}^{N^c} \mathbf{E}_f^{d_i} \underbrace{(\mathbf{R}_f^{d_i} \hat{\mathbf{A}} \mathbf{E}_f^{d_i})^{-1}}_{\hat{\mathbf{A}}_{d_i}} \mathbf{R}_f^{d_i} \quad (21)$$

The matrices $\mathbf{R}_f^{p_i}$ and $\mathbf{E}_f^{p_i}$ restrict and extend vectors between Ω_i^p and Ω , respectively, while $\mathbf{R}_f^{d_i}$ and $\mathbf{E}_f^{d_i}$ operate between Ω_i^d and Ω . These restriction and extension matrices are transposes of each other and contain only 0 and 1 entries. For details, see [24]. Applying $\mathbf{M}_d$ requires solving $N^d (= N^c)$ decoupled local problems on dual grids, and applying $\mathbf{M}_p$ requires solving N^p decoupled problems on primary grids. Both operations are embarrassingly parallelizable.

3.3. Monolithic PNM preconditioner

The overall structures of $\mathbf{M}_G$ and $\mathbf{M}_L$ for the monolithic *a*PNM preconditioner are identical to those of the monolithic *a*PLMM. The only difference lies in the definitions of “primary grid” and “dual grid,” which is reflected in the

¹ A term we coin here, mirroring the *contact-grain* smoother defined in [36] for linear elasticity.

permutation matrix W in Eq.6. In $aPNM$, the domain Ω is decomposed into primary grids Ω_i^p in a different way, still utilizing watershed segmentation. Unlike $aPLMM$, primary grids in $aPNM$ correspond to local geometric constrictions of Ω , and contact interfaces to geometric enlargements. In pore-network modeling parlance, these are the *throats* and *pores* of the void space, respectively. Details of how this decomposition is performed can be found in [24], but briefly, it involves providing interface pixels identified by $aPLMM$ as “seed points” to the watershed algorithm. The result is a decomposition like Fig.1d, where contact interfaces are shared by more than two primary grids and are located within geometric enlargements. For example, the contact interface in Fig.1g (yellow line denoted by Γ_{1234}) is shared among four primary grids: Ω_1^p , Ω_2^p , Ω_3^p , and Ω_4^p . Dual grids are constructed in exactly the same way as before, namely, by morphologically dilating the pixels comprising each contact interface as illustrated by Fig.1e. Unlike $aPLMM$, each column of the matrix B in Eq.16a for $aPNM$ can have more than two non-zero shape vectors $p_{c_j}^{p_i}$.

4. Geometric monolithic preconditioner

As noted in Section 3.2.2, the closure BC $\partial_n \mathbf{u} = \mathbf{0}$ is not imposed exactly at contact interfaces during the reduction step of M_G , except for the Stokes equations. The deviation grows further with increasing Reynolds number, Re , resulting in large errors near interfaces [24] after each application of M_G . These errors then convect downstream due to the upwind scheme used for the inertia term in the Navier-Stokes equations. Since the inertia and viscous terms cannot be separated algebraically in the discretized system, exact closure cannot be imposed. In Stokes flow, inertia is absent, so no such separation is needed. In Navier-Stokes flow, exact closure demands geometric information, i.e., access to the code used to assemble the Jacobian matrix in Eq.2. This would allow defining a modified residual $\tilde{r}_w^k$ (instead of r^k in Eq.2) whose Jacobian $\tilde{J}$ is used (in lieu of J^k) to build M_G . The modified residual we propose is:

$$\tilde{r}_w^k = \left[\begin{array}{c} \rho (\partial_t \mathbf{u}_k + \mathbf{w} \cdot \nabla \mathbf{u}_k) - \mu \Delta \mathbf{u}_k + \nabla p_k - \rho \mathbf{b} \\ \nabla \cdot \mathbf{u}_k \end{array} \right] \quad (22)$$

where the inertia term $\mathbf{u} \cdot \nabla \mathbf{u}$ is linearized around a given velocity $\mathbf{w}$, called “wind.” To compute $\mathbf{u}$ (and p) at some Re , we set $\mathbf{w}$ to the velocity at a nearby Reynolds number. The closer the Re corresponding to $\mathbf{w}$ is to the Re at which the solution is sought, the more accurate M_G will be. If no adjacent $\mathbf{w}$ is available, the Stokes velocity serves as fallback.

Aside from the linearization in Eq.22, we also remove the inertia term entirely from the modified residual of those fine-grid faces that lie within a few layers of each contact interface. Fig.3 illustrates such faces by tagging them with black dots. The layer-thickness of tagged faces depends on the size of the upwind stencil used for the inertia term (here, three-point). Removing the inertia term ensures that the reduction steps used to build M_G in Section 3.2.2 impose $\partial_n \mathbf{u} = \mathbf{0}$ exactly at interfaces. This is highlighted by the Navier-Stokes basis functions depicted in Fig.4, derived from the Jacobian of Eq.22. Each basis is a column of the matrix B in Eq.16a, comprised of two (in PLMM) or four (in PNM) shape vectors associated with the yellow-highlighted interfaces. They belong to primary grids in Figs.1f-g and correspond to a high Re . For comparison, Fig.4 also shows Stokes bases. Both satisfy $\partial_n \mathbf{u} = \mathbf{0}$ at interfaces (yellow).

To summarize, the resulting geometric PLMM and PNM preconditioners, denoted hereafter by $gPLMM$ and $gPNM$, are constructed from $\hat{A}$ set equal to the modified Jacobian $\tilde{J}_w$ of Eq.22 (where inertia is disabled for near-interface faces). The RHS vector $\hat{b}$ is computed as before from the original (not modified) residual in Eq.2 (i.e., $-r^k$). We show in Section 8 that $gPLMM$ and $gPNM$ perform better than $aPLMM$ and $aPNM$. Notice $\tilde{J}_w$ does not depend on the Newton iteration index k , meaning M_G is build once and reused thereafter. While J^k , used to build $aPLMM$ and $aPNM$, does depend on k , we found empirically that building M_G once and reusing it for all k is more efficient.

5. Coarse-scale Navier-Stokes solver

Here, we demonstrate that the global preconditioner M_G for $gPLMM$ and $gPNM$ can also be used as standalone coarse-scale solvers to compute approximate solutions to the *steady-state* Navier-Stokes equations (i.e., pre-turbulent regime). This approach results in significant computational speedup over exact solvers while maintaining acceptable accuracy in many applications, provided M_G corresponds to $gPLMM$ (not $gPNM$). Extension to dynamic (i.e., turbulent) flow is beyond our scope and left as future work. For Stokes flow, governed by the linear system $\hat{A}\hat{x} = \hat{b}$, we

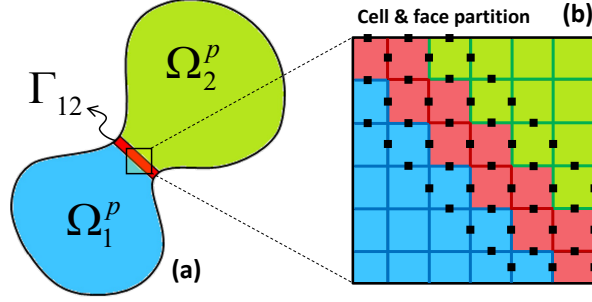


Figure 3: The inertia ($\mathbf{u} \cdot \nabla \mathbf{u}$) term is removed from the modified residual in Eq.22 of the fine-grid faces that lie within a few layers of each contact interface (Γ_{12}). These faces are tagged with black dots. The operation is required to impose the closure BC $\partial_n \mathbf{u} = \mathbf{0}$ exactly at contact interfaces.

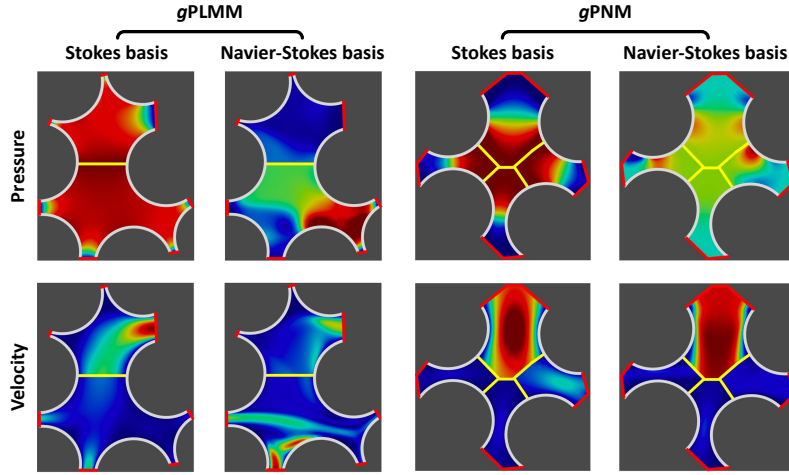


Figure 4: Basis functions (pressure and velocity magnitude) corresponding to the M_G of gPLMM and gPNM. Each basis is a column of the matrix B in Eq.16a, comprised of two (in gPLMM) or four (in gPNM) shape vectors associated with the yellow-highlighted interface. Navier-Stokes bases are built from the Jacobian of the modified residual in Eq.22 at $Re = 1.6 \times 10^4$. The bases belong to primary grids highlighted in Figs.1f-g, with BCs $p=0$ and $\partial \mathbf{u} / \partial n = 0$ on all red lines, and $p=1$ and $\partial \mathbf{u} / \partial n = 0$ on yellow lines. No-slip BCs hold on the void-solid interface (gray lines).

showed in [24] that an accurate approximate solution can be obtained from a single application of M_G^{-1} in Eq.18:

$$\hat{\mathbf{x}}_{apr} = M_G^{-1} \hat{\mathbf{b}} \quad (23)$$

However, a single application is insufficient for the steady-state Navier-Stokes equations due to its nonlinearity, requiring an iterative approach. To achieve speedup, we devise a Newton method that performs iterations only on the coarse-scale (pressure) unknowns, without recourse to the fine grid or any fine-scale problems (local or global):

$$J_c^k \delta_c^k = -r_c^k \quad (24)$$

The coarse-scale residual and Jacobian are defined as follows:

$$r_c^k = \hat{R} \tilde{r}^k \quad (25a)$$

$$J_c^k = \hat{R} \tilde{J}^k \hat{P} \quad (25b)$$

where $\tilde{r}^k$ and $\tilde{J}^k$ are the modified residual and Jacobian computed from Eq.22, respectively (setting $\partial_t \mathbf{u}_k = \mathbf{0}$). However, the omission of subscript w from both implies that we set $w = \mathbf{u}_k$ in Eq.22. In other words, $\tilde{r}^k$ and $\tilde{J}^k$ are the same as r^k and J^k in Eq.2, except the inertia term is removed from fine-grid faces near contact interfaces (Fig.3). $\hat{R}$ and $\hat{P}$ are the

effective restriction and effective prolongation matrices, respectively; defined in Eq.19. We build $\hat{R}$ and $\hat{P}$ by setting $\hat{A}$ in Section 3.2.2 to $\tilde{J}_w$ for a given wind w ($\neq u_k$) at a nearby Re . If no such w is available, the Stokes velocity serves as fallback. To map the course-scale update δ_c^k onto the fine grid and obtain the approximate fine-scale solution at the next Newton iteration, we execute:

$$\hat{x}_{aprx}^{k+1} = \hat{x}_{aprx}^k + \hat{P} \delta_c^k \quad (26)$$

Note that $\hat{R}$ and $\hat{P}$ do *not* depend on Newton iterations and are built only once. If they were built from J^k in Eq.2 instead of $\tilde{J}_w$, coarse iterations would diverge or yield poor approximate solutions. This is because the closure $\partial_n u = 0$ is only strictly valid at interfaces when $\tilde{J}_w$ is used to construct M_G . Thus, only M_G from gPLMM and gPNM can serve as coarse solvers, not aPLMM or aPNM. We show later that the M_G of gPLMM is far more accurate than gPNM.

Algorithm 1 summarizes the above steps for the proposed coarse-scale solver of the steady-state Navier-Stokes equations. The algorithm consists of an outer loop that gradually increments Re from zero to a desired maximum, enveloping the coarse-scale Newton solver as an inner loop. Here, Re is incremented by increasing the inlet-boundary pressure while keeping the outlet pressure constant. The outer loop's job is to provide a good initial guess for Newton to converge (iterations diverge if too far from $\hat{x}_{aprx}$). In computing $\tilde{J}_w$, we set w to the velocity from the previous Re .

Algorithm 1: Coarse-scale solver for steady-state Navier-Stokes equations

```

Initialize  $\hat{x}_{aprx} = 0$  and  $w = 0$ 
Do  $Re = 0$  to  $Re_{max}$  with increment  $\Delta Re$                                  $\triangleright$  Increment Reynolds number
    Set wind  $w$  in Eq.22 to velocity from previous  $Re$  increment
    Build coarse preconditioner  $M_G$  from Jacobian  $\tilde{J}_w$  of Eq.22                 $\triangleright$  This yields  $\hat{R}$  and  $\hat{P}$ 
    Compute coarse-scale residual  $r_c^0$  and Jacobian  $J_c^0$  from Eq.25
    Initialize Newton to  $\hat{x}_{aprx}^0 = \hat{x}_{aprx}$  from previous  $Re$  increment
    Do  $k = 0$  to  $k_{max}$                                                          $\triangleright$  Coarse-scale Newton iteration
        Compute coarse-scale Newton update  $\delta_c^k$  by solving Eq.24
        Update fine-scale approximate solution  $\hat{x}_{aprx}^{k+1}$  using Eq.26
        Update coarse-scale residual  $r_c^{k+1}$  and Jacobian  $J_c^{k+1}$  from Eq.25
        If  $\|r_c^{k+1}\| / \|r_c^0\| < 10^{-6}$ , exit
    End do
    Set  $\hat{x}_{aprx}$  to converged approximate solution
End do

```

Remark 2. For the linear Stokes equations, Algorithm 1 converges in only one Newton iteration and the solution equals Eq.23 but with $M_G^{-1} = \hat{P}(\hat{R}\hat{A}\hat{P})^{-1}\hat{R}$ (not Eq.18). Another way to formulate the coarse-scale Jacobian given by Eq.25b would have been $J_c^k = W\mathcal{C}(W^T \tilde{J}^k W)W^T$, but the presence of the column-sum operator $\mathcal{C}(\cdot)$ prevents Newton from converging in only one step, which is why we opted for the form given by Eq.25b. The simple proof of this fact involves tracing one Newton step of Algorithm 1, revealing that $(\hat{R}\tilde{J}^k\hat{P})^{-1}J_c^k = I$ must hold for one-step convergence.

6. Problem set

In the following sections, we first test the coarse-scale solver in Algorithm 1 against the exact solution of the steady-state Navier-Stokes equations for the 2D and 3D porous geometries in Fig.5. Both gPLMM and gPNM are considered as the basis for building M_G for the solver. The domains include a monodisperse and spatially disordered disk pack [25] (GLD4), a polydisperse and spatially disordered disk pack [25] (PD4), a 2D representation of Berea sandstone [37] (Berea), and a 3D Gyroid whose void-solid interface Γ_w is described by the following equation [38]:

$$\sin(12x)\cos(12y) + \sin(12y)\cos(12z) + \sin(12z)\cos(12x) = 0 \quad (27)$$

After gaining some intuition into the accuracy of gPLMM and gPNM as coarse-scale solvers, we test the algebraic (aPLMM and aPNM) and geometric (gPLMM and gPNM) monolithic preconditioners proposed for both steady-state and unsteady forms of the Navier-Stokes equations. The test domains include GLD4, PD4, Berea, and a 3D packing of non-spherical grains [24] (Granular). Disk positions in PD4 and GLD4 are identical, differing only in the disk

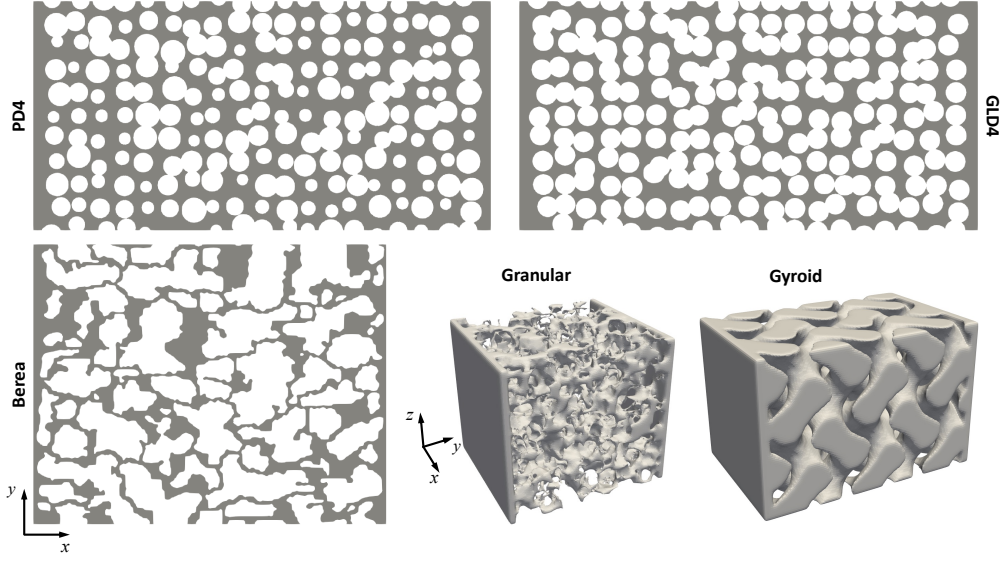


Figure 5: Schematic of 2D and 3D porous geometries considered for testing the proposed preconditioners and the coarse-scale solver.

sizes. We benchmark a PLMM, a PNM, g PLMM and g PNM against each other and block-triangular preconditioners b AMG, b PLMM, and b PNM. The latter were described in Section 1 and detailed in [24]. We utilize the *scaled*-BFBt approximation of the Schur-complement matrix in Eq.4, which exhibits robust performance in complex geometries [24]. The difference between b AMG, b PLMM, and b PNM lies in the choice of prolongation (transposition yields restriction) matrices used to precondition the elliptic sub-systems arising from the Schur-complement approximation. b PLMM and b PNM are built from the same shape and correction vectors used to formulate a PLMM and a PNM. To build b AMG, we use the `amg.m` code from the iFEM GitHub repository [39], with settings identical to those in [24].

Each domain is decomposed into primary and dual grids corresponding to the randomly colored regions in Fig.6. These are used by a PLMM, a PNM, g PLMM, g PNM, b PLMM, and b PNM. Table 1 summarizes each domain’s dimensions and fine/coarse-grid information, including the number of cell-pressure unknowns (n_c), face-velocity unknowns (n_f), primary grids (N^p), and dual grids ($N^d = N^c$) for both PLMM and PNM. The N^p for PLMM roughly equals the N^c for PNM because primary grids in PLMM correspond to contact interfaces (or dual grids) in PNM and vice versa.

To compute Reynolds number $Re = \rho U l / \mu$, we define the characteristic velocity as the interstitial velocity $U = U_D / \phi$, where U_D is the Darcy velocity (= flowrate divided by cross-sectional area) and ϕ is porosity. The characteristic length is $l = V_s / A_w$, where V_s is the total solid volume and A_w is the void-solid interfacial area $|\Gamma_w|$. In all simulations, we set the fluid density to $\rho = 1$ g/cc and the dynamic viscosity to $\mu = 2 \times 10^{-4}$ Poise for PD4, 1.2×10^{-4} Poise for GLD4, 1×10^{-4} Poise for Berea, and 1×10^{-3} Poise for Granular and Gyroid. The BCs imposed were described in Section 2, which include setting the inlet pressure p_{in} at the left boundary and the outlet pressure p_{out} at the right.

To validate our fine-grid Navier-Stokes solver, we simulate the classic problem of flow around a circular or square obstacles at various Reynolds numbers in Appendix A. These simulations show that our solver produces accurate drag coefficients compared to literature values, with comparable pressure, velocity-magnitude, and vorticity fields.

Table 1: Geometric and fine/coarse-grid information of the domains in Fig.5. They include the domain size, number of cell-pressure unknowns (n_c), number of face-velocity unknowns (n_f), and number of primary (N^p) and dual ($N^d = N^c$) grids for PLMM- and PNM-based preconditioners.

	Domain size (cm)	n_c	n_f	N^p , PLMM	N^c , PLMM	N^p , PNM	N^c , PNM
PD4	2×1	610,040	1,230,924	174	294	291	172
GLD4	2×1	717,097	1,447,556	186	257	260	178
Berea	1.45×1.15	931,396	1,877,760	203	255	302	192
Granular	$0.9 \times 1 \times 0.9$	375,953	1,265,199	99	690	647	96
Gyroid	$1 \times 1.5 \times 1$	616,400	1,947,468	21	103	109	16

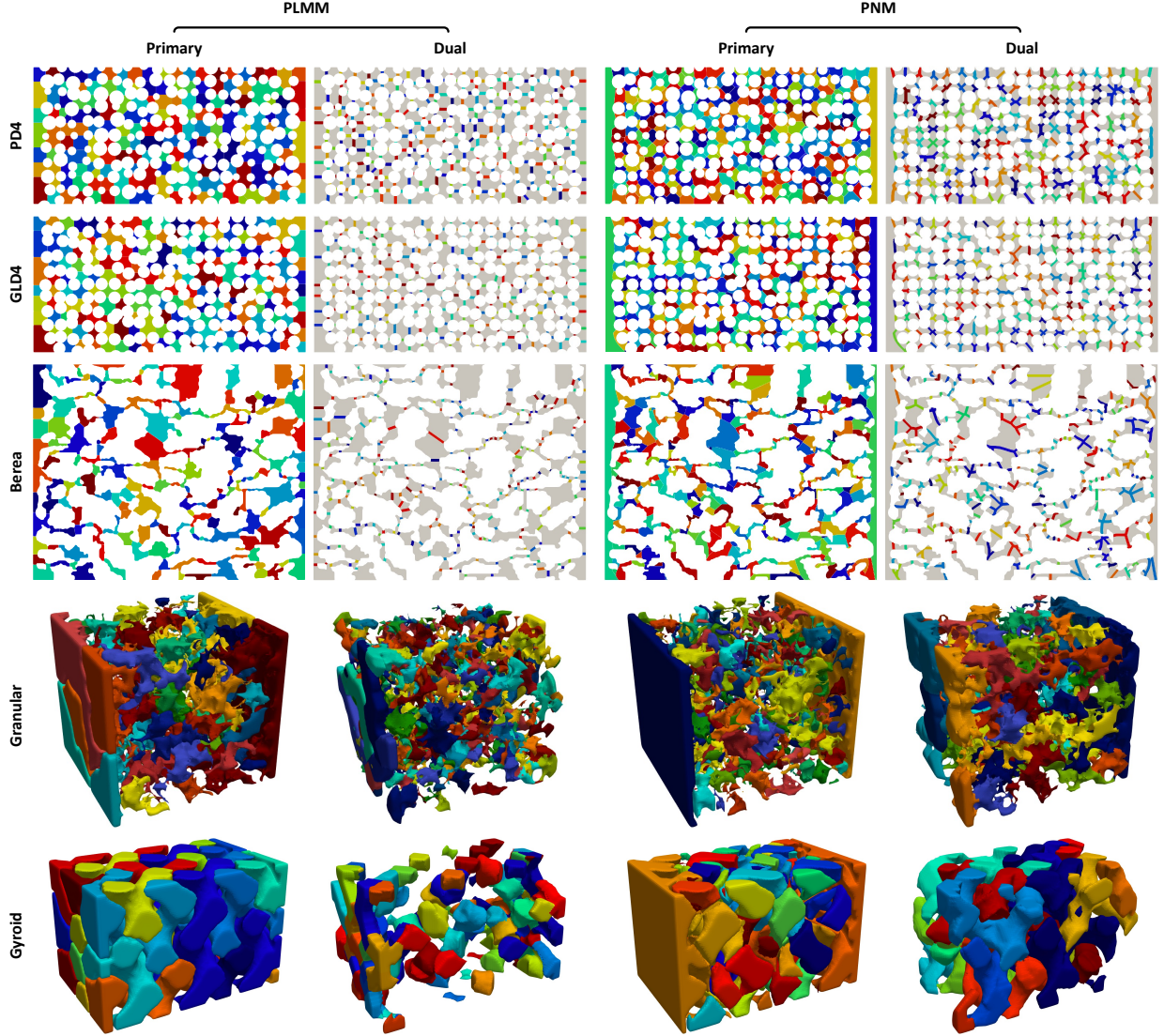


Figure 6: Decomposition of the porous geometries shown in Fig.5 into primary and dual grids used by the coarse-scale solver and the monolithic and block preconditioners considered herein. Each primary grid and dual grid is illustrated by a randomly colored region.

7. Testing the coarse-scale solver

We test the accuracy of the coarse-scale solver outlined in Algorithm 1 for the steady-state Navier-Stokes equations by comparing the pressure and velocity magnitudes associated with the approximate solution $\hat{x}_{apr}$ produced to that of the exact solution. To increment Re in Algorithm 1, we progressively increase the inlet pressure p_{in} by 10% of its previous value while keeping the outlet pressure p_{out} fixed at zero. The initial value of p_{in} is set to 10^{-3} dynes/cm². We increment Re in this way until the coarse-scale Newton loop ceases to converge within 50 iterations, which we interpret as the onset (or vicinity) of turbulent instabilities beyond which a steady-state solution is difficult to compute. Recall Algorithm 1 uses the converged solution from the previous Re increment as the initial guess for Newton in the current Re increment. In addition to pressure and velocity fields, which are fine-scale information, we compute the permeability K of each domain versus Re , which is a bulk property:

$$K = \frac{\mu L U_D}{\Delta p} \quad (28)$$

The U_D , Δp , and L denote Darcy velocity, $p_{in} - p_{out}$, and inlet-to-outlet distance of the domain, respectively.

Figs.7–8 compare approximate solutions from the coarse-scale solver to exact solutions at the highest Re reachable by the coarse-scale solver. Results are shown for the PD4, GLD4, Berea and Gyroid domains. Approximate solutions from both gPLMM and gPNM are included, where the corresponding M_G (or equivalently $\hat{R}$ and $\hat{R}$) is utilized in Algorithm 1. We observe very good agreement between approximate pressure and velocity-magnitudes from gPLMM versus the exact solutions. Approximate solutions via gPNM are significantly inferior in accurately capturing the flow field. These observations are confirmed further by Fig.9, where comparisons of K versus Re are presented.

To quantify errors in the approximate solutions, we use the following metrics by [24, 25]:

$$E_2^\xi \equiv \left(\frac{1}{|\Omega|} \int_{\Omega} (E_p^\xi)^2 d\Omega \right)^{1/2} \quad E_p^\xi \equiv \frac{|\xi_t - \xi_c|}{\sup_{\Omega} |\xi_t|} \times 100 \quad E_K \equiv \frac{|K_t - K_c|}{|K_t|} \times 100 \quad (29)$$

where ξ is a placeholder for either pressure, p , or velocity magnitude, $|\mathbf{u}|$. Subscripts c and t denote the approximate solution obtained from the coarse-scale solver and the “true” (or exact) solution, respectively. From left to right, the metrics in Eq.29 yield the L_2 , pointwise, and permeability errors; all expressed in percentages of the true value.

Fig.9 plots E_2^p , $E_2^{|\mathbf{u}|}$, and E_K versus Re for all domains, where we see these errors increase with Reynolds number. Moreover, errors associated with gPNM are much larger (up to 20 times) than those with gPLMM. For the coarse-solver based on gPLMM, errors are bounded as follows for all domains: $E_2^p < 3\%$, $E_2^{|\mathbf{u}|} < 6\%$ in 2D and $< 18\%$ in 3D, and $E_K < 30\%$, which are acceptable in many engineering applications. We note the approximate velocity field of the coarse-scale solver is conservative in a “weak” sense, i.e., integrated flux along each interface is continuous. Fig.10 depicts pointwise errors of the pressure and velocity-magnitude in Figs.7–8 for the PD4 and Gyroid domains, which

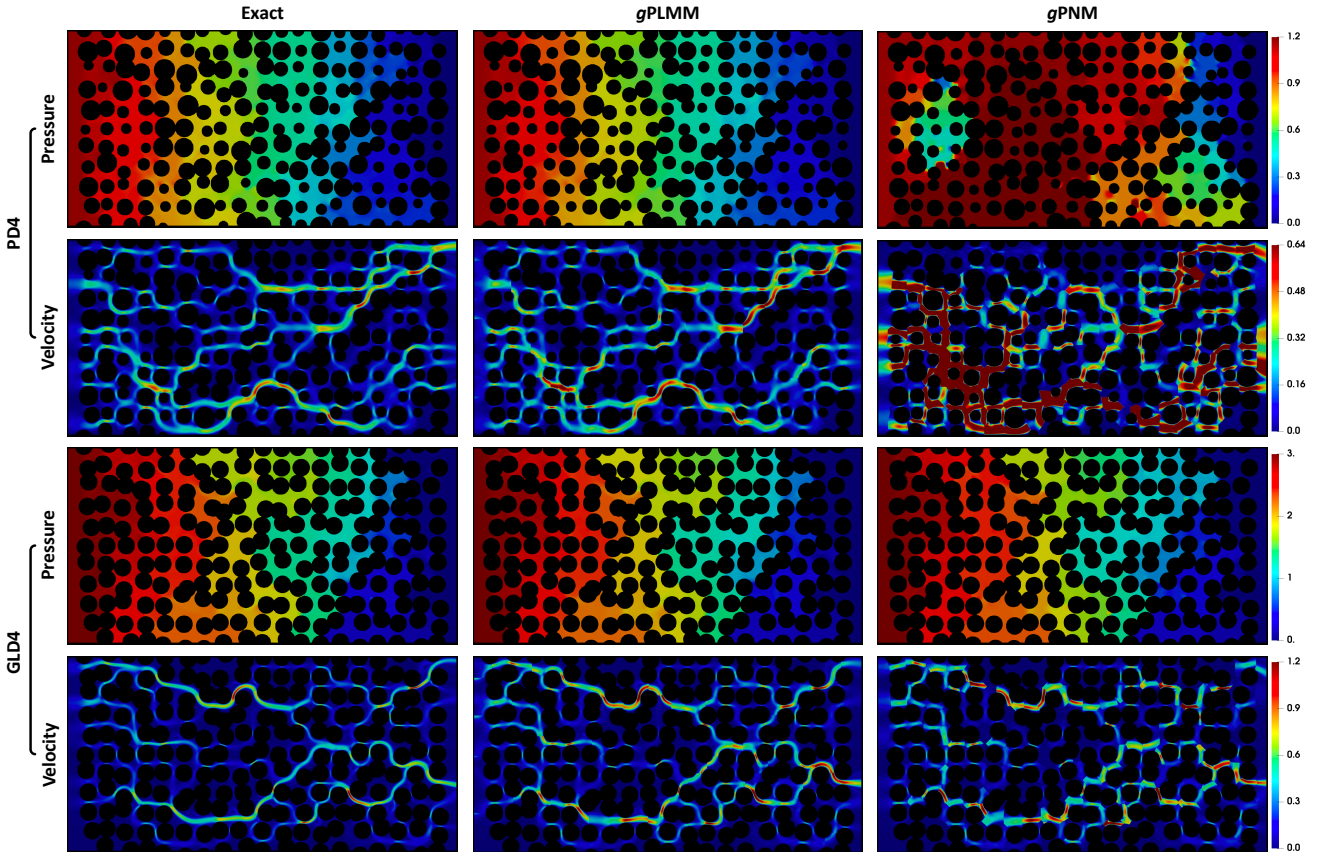


Figure 7: Comparison of approximate pressure and velocity-magnitude fields obtained from the coarse-scale solver in Algorithm 1 against exact solutions. Results correspond to the *steady-state Navier-Stokes equations* on PD4 at $Re = 3.1 \times 10^3$ and GLD4 at $Re = 1.1 \times 10^4$.

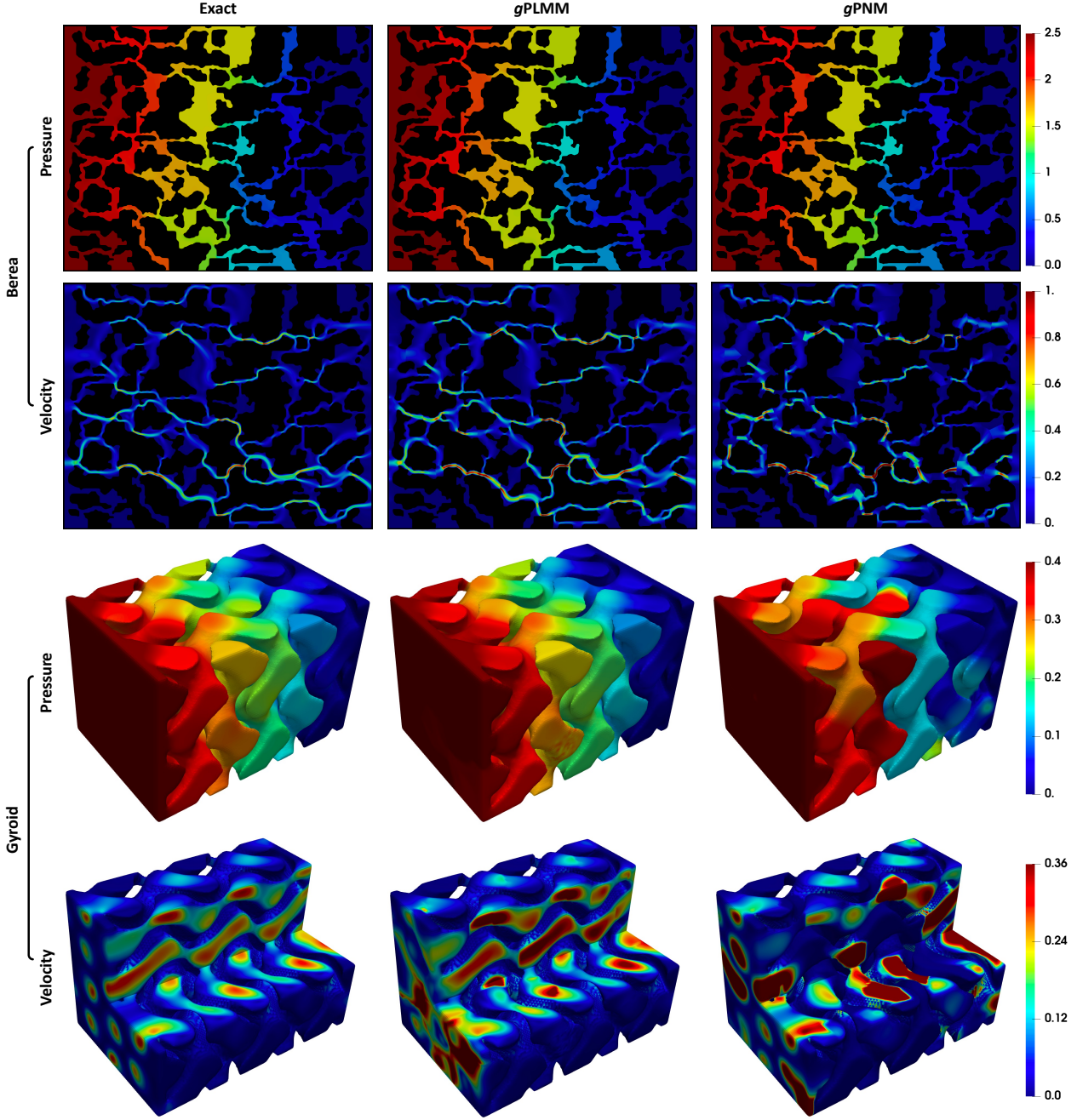


Figure 8: Comparison of approximate pressure and velocity-magnitude fields obtained from the coarse-scale solver in Algorithm 1 against exact solutions. Results correspond to the *steady-state Navier-Stokes equations* on Berea at $Re = 1.6 \times 10^4$ and Gyroid at $Re = 4.7 \times 10^3$.

confirm the observations just described. $gPLMM$ errors are lower than $gPNM$, and under 1% for much of the domain. Other domains behave similarly and omitted for brevity. Unlike Stokes flow, where errors localize more strongly near contact interfaces [24] (high-frequency), Navier-Stokes exhibits errors that are comparatively global (low-frequency).

An interesting observation from Fig.9 is that the permeability predicted by the $gPLMM$ coarse-scale solver is always larger than the true permeability computed from the exact solution. This is because the imposition of the closure BC $\partial_n \mathbf{u} = \mathbf{0}$ suppresses vortex formation, which is a key mechanism for dissipating energy, hence reducing

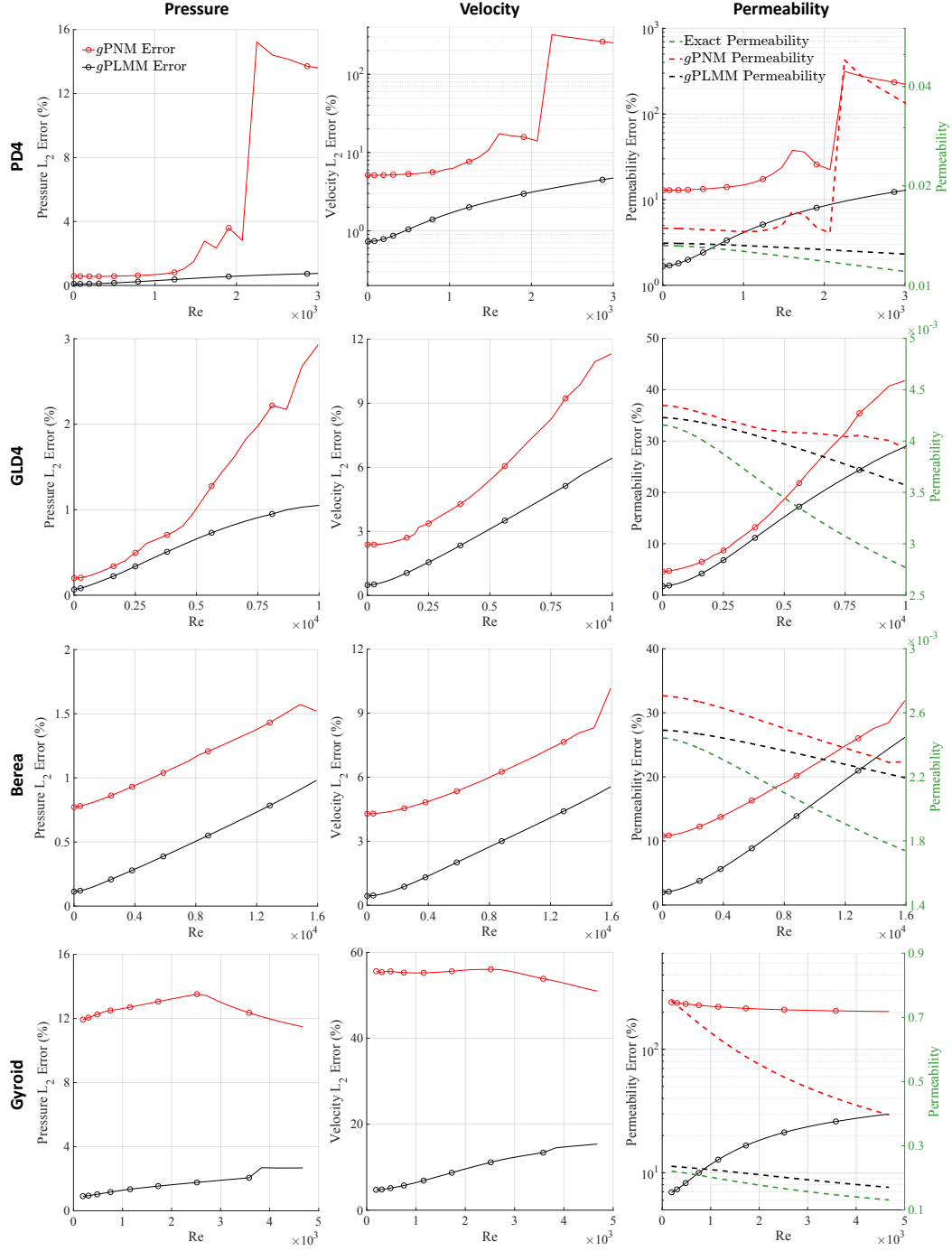


Figure 9: Errors in pressure (E_2^p), velocity magnitude ($E_2^{||u||}$), and permeability (E_K) versus Reynolds number (Re) for approximate solutions from the coarse-scale solver in Algorithm 1 based on gPLMM and gPNM. Results correspond to the PD4, GLD4, Berea, and Gyroid domains. The approximate and exact permeabilities (K) are plotted with dashed lines on the right. The former overestimates K due to vortex suppression.

flowrate. This is illustrated by Fig.11, where streamlines at low and high Re are drawn inside a portion of the domain shown in Fig.1a. The suppression of vortices is, incidentally, the reason why errors in Fig.9 grow with Re .

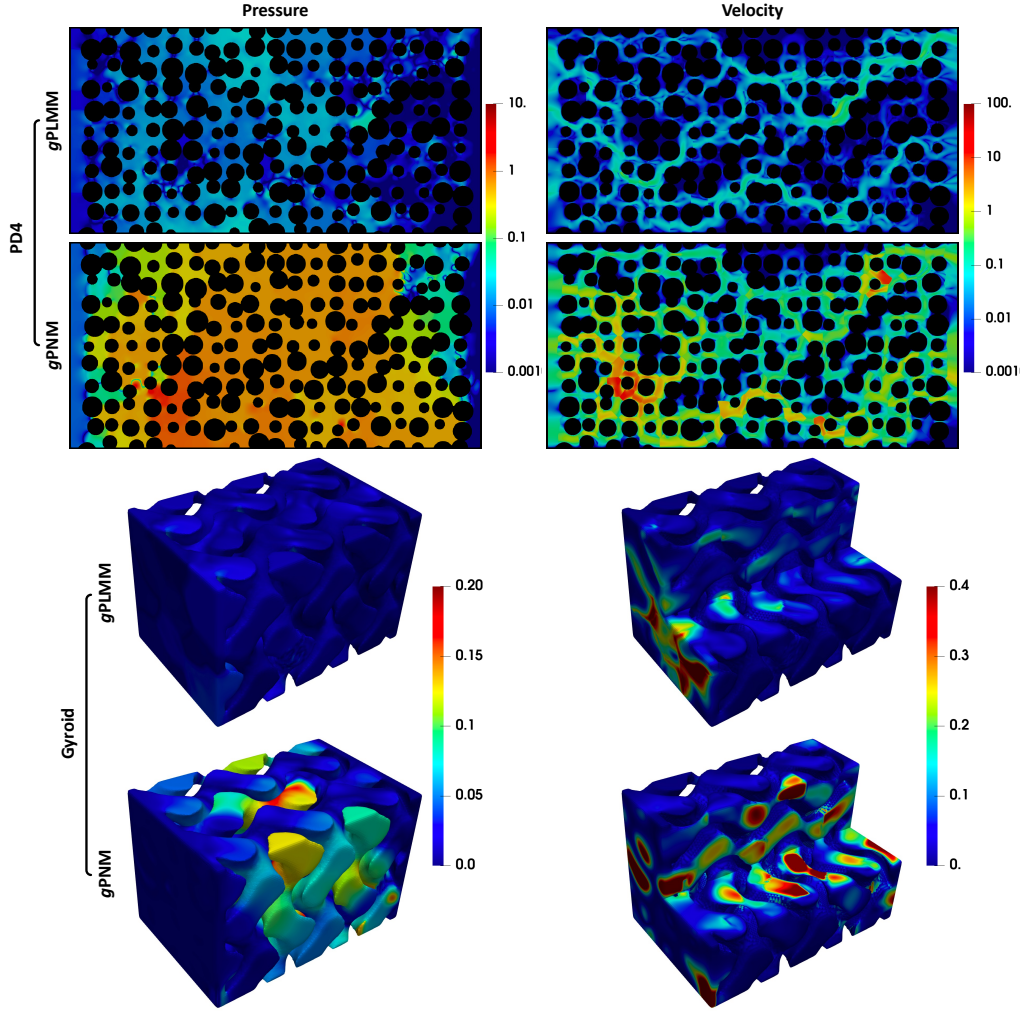


Figure 10: Pointwise errors in pressure (E_p^p) and velocity magnitude ($E_p^{||}$) for approximate solutions from the coarse-scale solver in Algorithm 1 based on gPLMM and gPNM. Results correspond to PD4 at $Re = 3.1 \times 10^3$ and Gyroid at $Re = 4.7 \times 10^3$, whose field plots are shown in Figs.7–8.

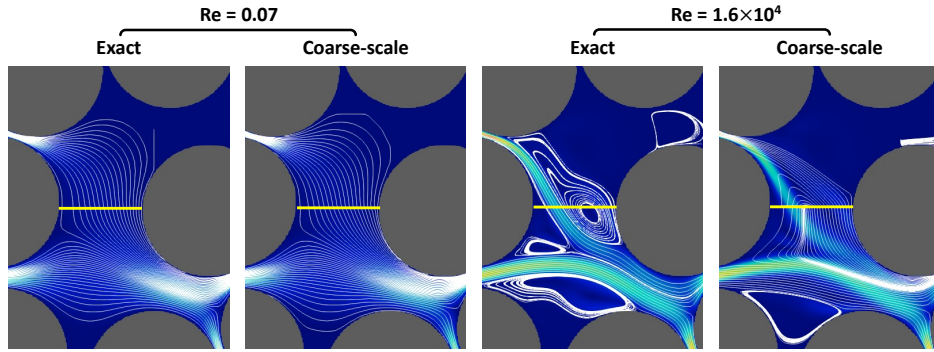


Figure 11: Schematic of flow streamlines (white lines) near a contact interface (yellow line). The velocity field is computed from the gPLMM coarse-scale solver and the exact solution. Results are shown for low and high Re corresponding to a portion of the domain in Fig. 1.

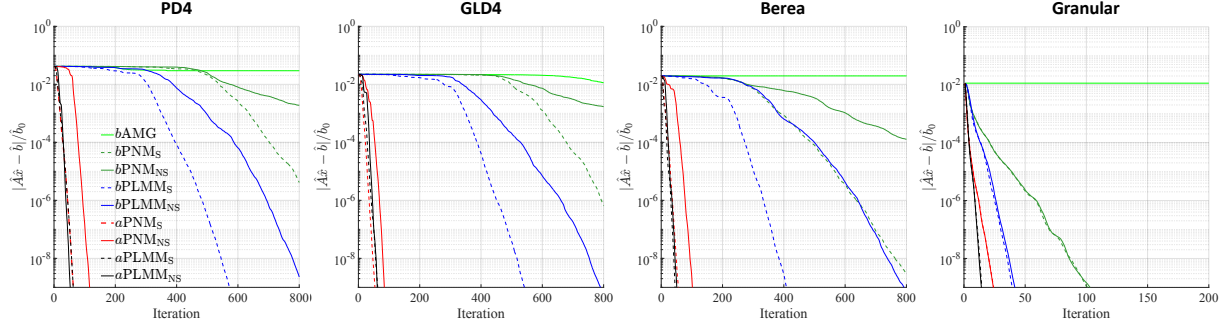


Figure 12: Normalized linear-system residual versus GMRES iterations for the first Newton step of the first Re increment ($Re_1 = 9.0 \times 10^3$). Results correspond to different block and monolithic preconditioners applied to the *steady-state Navier-Stokes equations* on all domains.

8. Testing the preconditioners

We apply all preconditioners, including $aPLMM/aPNM$, $gPLMM/gPNM$, $bPLMM/bPNM$, and $bAMG$, within right-preconditioned GMRES. We declare convergence when the normalized residual satisfies $\|\hat{A}\hat{x} - \hat{b}\|/\|\hat{b}_0\| < 10^{-9}$, where $\hat{b}_0$ is the residual of the steady-state (or time-dependent if velocity at prior time step is set to zero) Navier-Stokes equations at the origin, i.e., $r(\mathbf{u} = \mathbf{0}, p = 0)$. This captures the magnitude of nonhomogeneous BCs. We use $\|\hat{b}_0\| = \|r^0\|$ for normalization instead of $\|\hat{b}\| = \|r^k\|$ because the latter decreases with Newton iterations, leading to excess GMRES iterations at large k . We run simulations in series on an Intel® Xeon® Gold 6248R CPU @ 3.00 GHz machine.

8.1. Steady-state Navier-Stokes equations

Similar to Section 7, we increment Re by progressively increasing the inlet pressure p_{in} from 1 dynes/cm² using constant increments of $\Delta p_{in} = 1$ for PD4, 0.5 for GLD4 and Berea, and 2 for the Granular domain, while keeping the outlet pressure at $p_{out} = 0$. Unlike Section 7, fine-scale (not coarse-scale) solutions are computed at each Re by performing Newton iterations on the fine grid, and GMRES iterations at each Newton iteration. The initial Newton guess at any given Re is the solution from the previous Re increment. To probe the necessity of rebuilding the $aPLMM/aPNM$, $gPLMM/gPNM$, and $bPLMM/bPNM$ preconditioners as Re evolves, we test two variants of each: (1) *Stokes-based*, where the preconditioners are built from the Stokes equations once and reused across all subsequent Re ; and (2) *Navier-Stokes-based*, where the preconditioners are rebuilt from scratch at the start of every Re increment; specifically, before the Newton loop is entered, but not during. Following the notation in Section 1, we denote Stokes-based preconditioners with subscript “S” and Navier-Stokes-based preconditioners with “NS.” For example, $bPLMM_S$ is the block PLMM preconditioner based on the Stokes equations, and $aPNM_{NS}$ is the algebraic monolithic PNM preconditioner that is continuously updated based on the Navier-Stokes equations. Since $gPLMM_S = aPLMM_S$ and $gPNM_S = aPNM_S$, we use $gPLMM$ and $gPNM$ (no subscript) to denote Navier-Stokes-based preconditioners.

First, let us demonstrate that block preconditioners are disqualified from further consideration due to their severely poor performance. Fig.12 compares the GMRES convergence observed for the monolithic and block preconditioners in the first Newton iteration of the first Re increment. This is the simplest system to solve, as higher Re poses further challenges to GMRES (shown later). Compared to $aPLMM$ and $aPNM$, the block preconditioners $bPLMM$, $bPNM$, and $bAMG$ are extremely slow to converge with long periods of initial stagnation. For $bAMG$, stagnation is permanent. By contrast, monolithic preconditioners converge robustly and at least an order magnitude faster. This observation is

Table 2: Total wall-clock times (seconds) associated with the first three Re increments in solving the steady-state Navier-Stokes equations on the PD4 domain. They include the costs of constructing M_G , constructing M_L , and the self-time of GMRES for different preconditioners in GMRES.

	$bPNM_S$	$bPNM_{NS}$	$bPLMM_S$	$bPLMM_{NS}$	$aPNM_S$	$aPNM_{NS}$	$aPLMM_S$	$aPLMM_{NS}$
$Re_1 = 9.0 \times 10^3$	40,088	87,345	22,295	44,478	658	1,158	566	758
$Re_2 = 1.2 \times 10^4$	48,796	110,934	31,011	65,789	674	1,800	563	741
$Re_3 = 1.4 \times 10^4$	62,347	137,529	49,125	102,886	824	2,405	643	769

insensitive to whether block preconditioners are Stokes-based or Navier-Stokes-based. In [24], the authors observed better performance by $bPLMM$ and $bPNM$ for the Stokes equations than what is observed here for the Navier-Stokes equations. However, even there, $aPLMM$ and $aPNM$ were superior, and $bAMG$ was prone to stagnation, consistent with Fig.12. Table 2 lists the total wall-clock times (WCTs) associated with the first three Re increments of the PD4 domain, using different preconditioners. The WCTs include contributions from the costs of building M_G , building M_L , and the self-time of GMRES. We see that block preconditioners take roughly 1-2 orders of magnitude longer than the monolithic preconditioners. Hence, we exclude $bPLMM$, $bPNM$, and $bAMG$ from all subsequent analysis.

Fig.13 shows the number of Newton and GMRES iterations as a function of Re for the PD4, GLD4, Berea, and Granular domains. GMRES iterations correspond to the first Newton step of each Re , where GMRES struggles the most compared to subsequent Newton steps. Notice that the number of Newton iterations (between 2 to 6) is almost constant with respect to Re , which is a direct result of using the converged solution from the prior Re as the initial guess for the current Re . By contrast, GMRES iterations increase steadily with Re , indicative of $\hat{A}\hat{x} = \hat{b}$ becoming more challenging to solve at higher Reynolds numbers. *The best-performing preconditioner is consistently $gPLMM$ across all domains*, outperforming $aPLMM_{S/NS}$ by up to a factor of ~ 2 , and $aPNM_{S/NS}$ by up to a factor of ~ 4 . The only exception is Granular, where $gPLMM$ and $aPLMM_{S/NS}$ are comparable, and $aPNM_{S/NS}$ slightly outperforms $gPLMM$ at high (but not low) Re . Interestingly, $gPNM$ performs as poorly as $aPNM_{S/NS}$, implying that the closure BC $\partial_n \mathbf{u} = \mathbf{0}$ is only meaningful for $gPLMM$, but not $gPNM$. Moreover, comparing $aPLMM_S$ to $aPLMM_{NS}$, we see little difference in performance, implying the Stokes-based preconditioner is more attractive because it is built only once. As a general rule, preconditioners based on PLMM are seen to perform far better than those based on PNM.

Fig.13 also includes the performance of GMRES preconditioned by only the smoothers of PLMM and PNM, i.e., M_L not paired with M_G . We see the PLMM smoother performs better than the PNM smoother, but neither beats their respective monolithic preconditioners, where M_L is paired with M_G . This confirms that such pairing is necessary.

Fig.14 depicts total wall-clock times (WCT), T_{tot} , associated with computations in Fig.13 versus Re . This includes costs corresponding to building M_G (T_{MG}), building M_L (T_{ML} ; here LU-decomposition of local systems in the additive-

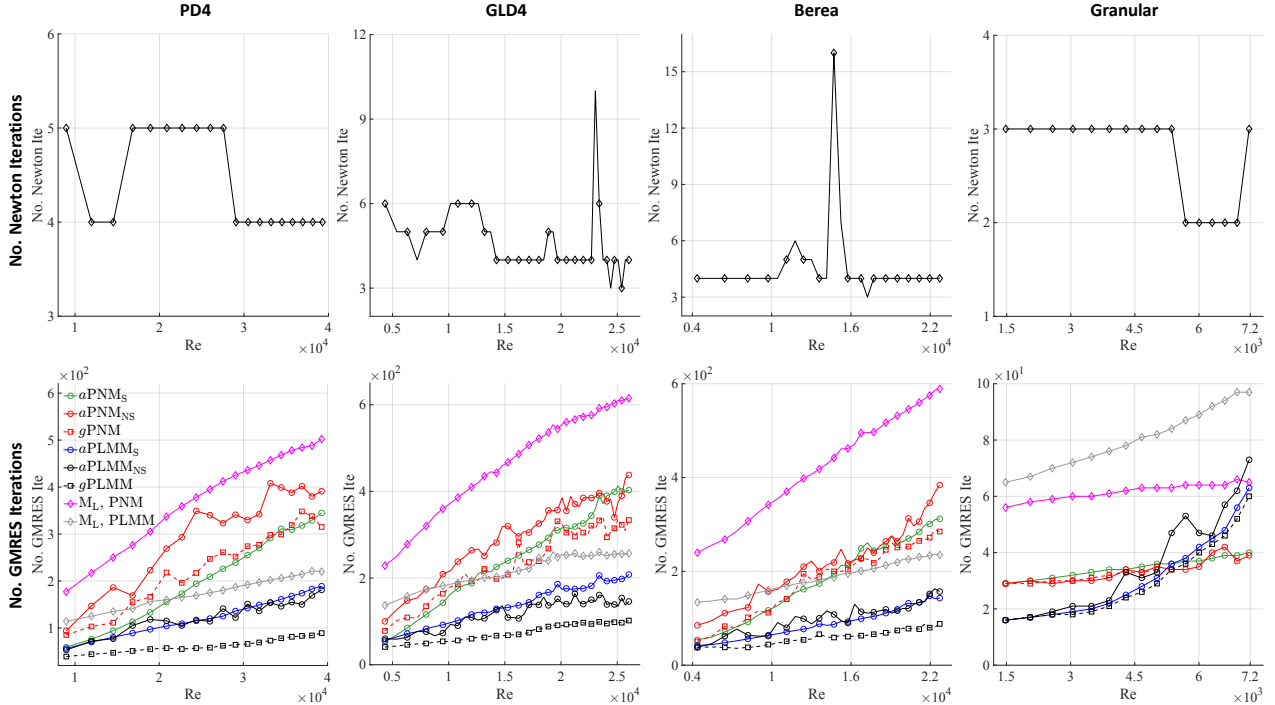


Figure 13: (Top row) Number of Newton iterations versus Re . (Bottom row) Number of GMRES iterations of the first Newton step versus Re . Results belong to different preconditioners based on PLMM and PNM applied to the *steady-state Navier-Stokes equations* in all domains. Prefix a means “algebraic,” and g means “geometric.” Subscripts “S” and “NS” imply preconditioners are built from Stokes or Navier Stokes, respectively.

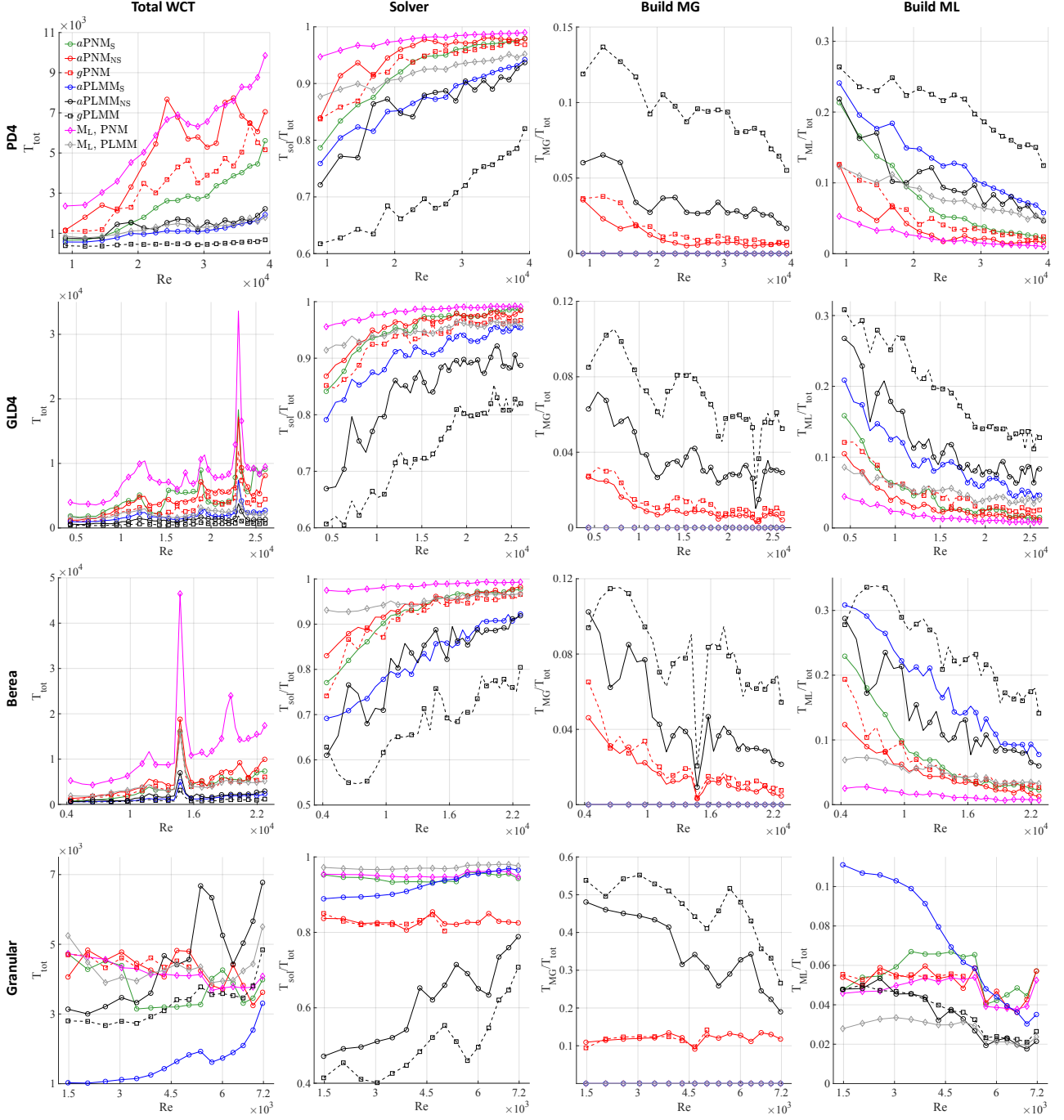


Figure 14: Total wall-clock times (WCT; in seconds) spent solving the *steady-state Navier-Stokes equations* (T_{tot}) versus Re , for different preconditioners based on PLMM and PNM (built from Stokes and Navier Stokes) in all domains. T_{tot} consists of the costs of building M_G (T_{MG} ; if monolithic), building M_L (T_{ML}), and self-time of GMRES (T_{sol}). The breakdown of T_{tot} into T_{MG} , T_{ML} , and T_{sol} is illustrated in fractions of T_{tot} .

Schwarz smoothers), and self-time of GMRES (T_{sol}). Results belong to the $aPLMM$, $aPNM$, $gPLMM$, $gPNM$, and M_L preconditioners. The breakdown of T_{tot} into T_{MG} , T_{ML} , and T_{sol} is shown as fractions of T_{tot} in Fig.14. We see in all cases, T_{sol} dominates T_{tot} , making the relative ranking among preconditioners in terms of T_{tot} similar to that already discussed in Fig.13 in terms of GMRES iterations. To quantify speedup, we normalize the T_{tot} of all preconditioners

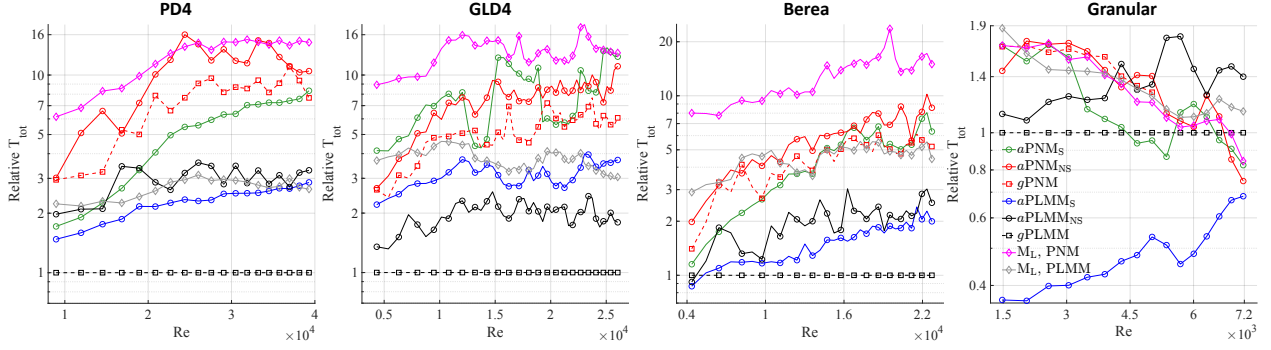


Figure 15: Relative performance of all monolithic and smoother-only preconditioners in Fig.14 measured against *gPLMM* for the *steady-state Navier-Stokes equations*. Relative performance is defined as the ratio of total WCT (T_{tot}) for the preconditioner in question over that of *gPLMM*.

against that of *gPLMM* in Fig.15. Compared to *gPLMM*, *aPLMM*_{S/NS} are 2-3 times slower, *aPNM*_{S/NS} are ~ 10 times slower, the M_L of *PLMM* is 3-5 times slower, and the M_L of *PNM* is 10-20 times slower. The only exception is 3D Granular, whose T_{MG} constitutes a sizable fraction of T_{tot} due to the presence of a few large primary grids, making the continuous updating of M_G versus Re in *gPLMM* more costly than the one-time cost of construction in *aPLMM*_S. Once again, *aPLMM*_S and *aPLMM*_{NS} have comparable T_{tot} across all domains, except Granular for reasons just stated. This makes *aPLMM*_S more attractive than *aPLMM*_{NS} as it is not rebuilt at every Re increment.

To understand why *gPLMM* outperforms its algebraic counterparts *aPLMM*_{S/NS}, and why *PLMM*-based preconditioners outperform *PNM*-based ones, we visualize the spectrum of the error-propagation matrix E :

$$E_G = I - M_G^{-1}\hat{A}, \quad E_L = I - M_L^{-1}\hat{A} \quad \Rightarrow \quad E = E_L E_G \quad (30)$$

for *gPLMM*, *gPNM*, *aPLMM*_{S/NS}, and *aPNM*_{S/NS} in Fig.16. Each spectrum consists of the 500 largest eigenvalues of E for the PD4 domain at $Re = 3.9 \times 10^4$ (other domains yield similar results). We see that the spectrum of *gPLMM* is clustered closest to the origin of the complex plane and lies inside the unit circle, explaining its superior GMRES convergence. Next in ranking are *aPLMM*_{S/NS}, which have similar performance. While *aPLMM*_{NS} is more clustered near the origin than *aPLMM*_S, the former is plagued by rogue eigenvalues outside the unit circle slowing down convergence. Lastly, *aPNM*_{S/NS} and *gPNM* have the largest spectral radii, thus the slowest GMRES convergence.

In addition to E 's spectrum, Fig.16 shows the spectra of E_G and E_L defined by Eq.30. These correspond to the error-propagation matrices of M_G and M_L , respectively. Eigenvalues of E_L lie inside the unit circle for both the *PLMM* and *PNM* smoothers, with many very close to one, confirming the smoothers alone are inadequate as preconditioners. For *aPLMM*_{NS} and *aPNM*_{NS}, Fig.16 shows the eigenvalues of E_G consist of zero or one, because E_G is a projection by Proposition 3.1 and our simplification for M_G in Remark 1. For *gPLMM*, *gPNM*, *aPLMM*_S, and *aPNM*_S, E_G 's eigenvalues lie outside the unit circle, because the $\hat{A}$ used in Eq.30 is different from the Jacobian used to construct M_G . What is interesting is that the spectral radius of E for *gPLMM* and *aPLMM*_S (but not *gPNM* and *aPNM*_S) is smaller than one, despite the spectral radius of E_G being larger than one. The opposite is true for *aPLMM*_{NS} and *aPNM*_{NS}, where the spectral radius of E is larger than one despite both E_G and E_L having spectra contained within the unit disk. The latter is possible because both E_G and E_L are non-normal matrices, with non-orthogonal eigenvectors [40].

8.2. Unsteady-state Navier-Stokes equations

We focus next on the unsteady-state Navier-Stokes equations and the turbulent flow regimes that result from it. To ensure flow is turbulent at all t , we set the IC equal to the steady-state solution from the previous section at the highest simulated Re . The BCs are identical to the steady-state simulations in Section 8.1, where the inlet pressure p_{in} is incrementally increased while the outlet pressure is $p_{out} = 0$. The only difference here is that a timescale is associated with the growth of p_{in} . Specifically, we take one Δp_{in} increment per $\Delta t = 0.2$ sec, where $\Delta p_{in} = 20$ dynes/cm² in all 2D domains and $\Delta p_{in} = 30$ dynes/cm² in the 3D Granular domain. Within each 0.2 sec interval, during which p_{in} is constant, smaller time steps dt are taken to capture the turbulent details of flow. The dt are taken adaptively, based on how fast the residual norm evolves. We test the performance of the *gPLMM*, *aPLMM*_{S/NS}, and *aPNM*_{S/NS}

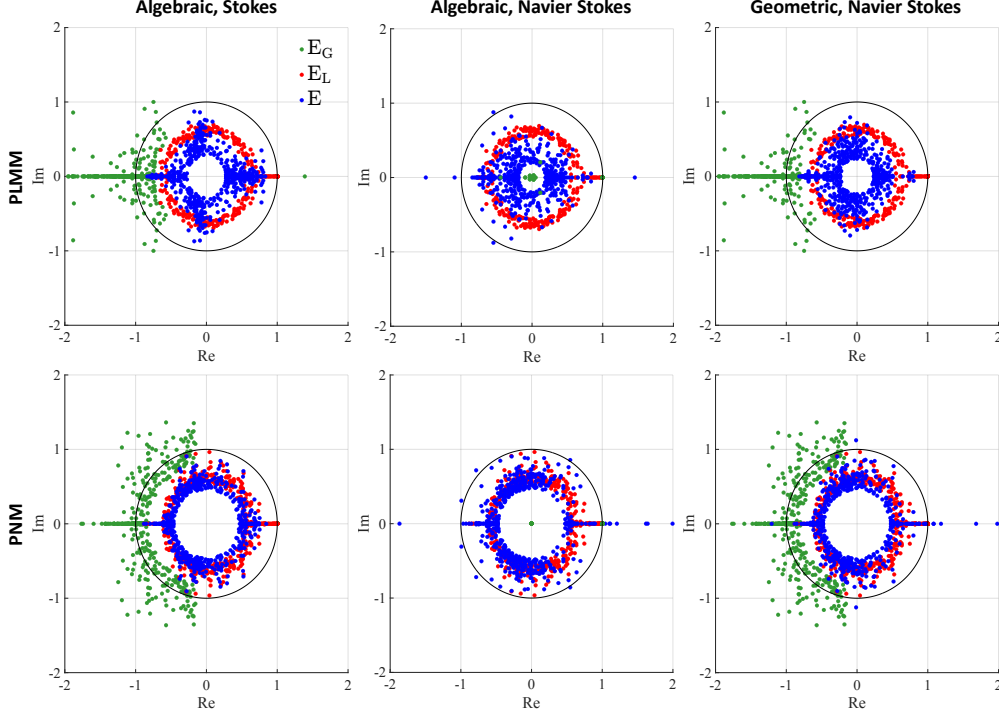


Figure 16: Spectra (500 largest eigenvalues) of the error-propagation matrices E (blue dots), E_G (green dots), and E_L (red dots) in Eq.30. Results belong to preconditioners $aPLMM_S$ and $aPNM_S$ (left column), $aPLMM_{NS}$ and $aPNM_{NS}$ (middle column), and $gPLMM$ and $gPNM$ (right column) applied to the *steady-state Navier-Stokes equations* on the PD4 domain at $Re = 3.9 \times 10^4$.

preconditioners on the PD4, GLD4, Berea, and Granular domains (see Fig.5). We exclude $gPNM$ from consideration based on its poor performance in Section 8.1. Navier-Stokes-based preconditioners (all except $aPLMM_S$ and $aPNM_S$) are rebuilt at the *beginning* of every Re (or p_{in}) increment, not in between dt steps or Newton iterations. Since dynamic simulations are computationally expensive, underperforming preconditioners are terminated prematurely whenever it is clear they are not competitive. Finally, GMRES iterations preconditioned by $aPNM_{S/NS}$ diverge or stagnate in the Granular domain, even for very small dt . This is why $aPNM_{S/NS}$ results for Granular appear absent from Figs.18-20.

To visualize the complex nature of turbulent flow, Fig.17 depicts snapshots of fluid pressure, velocity magnitude, and vorticity magnitude ($|\nabla \times \mathbf{u}|$) at the highest simulated Re in all domains. Compared to steady-state flow in Figs.7-8, we see many more circulation zones and chaotic patterns. Fig.18 shows that the number of dt time steps, taken within each Re increment, increases with Re , indicating progressively smaller dt is required to capture the growing intricacies of turbulent flow. The number of Newton and GMRES iterations per dt step (averaged over each Re increment) are also depicted as a function of Re . GMRES iterations correspond to the first Newton step of each Re , where GMRES struggles the most compared to later Newton steps. Similar to Section 8.1, Newton iterations are constant (~ 5) versus Re . However, unlike Section 8.1 where GMRES iterations grow with Re , they remain nearly constant here. We attribute this to the small dt used to discretize $\partial_t \mathbf{u}$ in Eq.1, rendering the (1,1)-block of $\hat{A}$ in Eq.2 diagonally dominant. At the limit of large dt , the $\partial_t \mathbf{u}$ term becomes negligible and the steady-state results of Fig.13 are recovered.

Similar to Section 8.1, Fig.18 suggests that the *best-performing preconditioner is $gPLMM$ across all domains*. However, unlike Section 8.1, $aPLMM_{NS}$ is a *very close second*, slower by only <1.5 times in 2D and <2.8 times in the 3D Granular. By contrast, $gPLMM$ is faster than $aPNM_{NS}$ by <2 times in 2D domains (3D diverged), and than $aPLMM_S$, $aPNM_S$, and $PLMM/PNM$ smoothers by >10 times in 2D and <2 times in 3D Granular. Clearly, preconditioners based on Navier Stokes ($gPLMM$, $aPLMM_{NS}$, $aPNM_{NS}$) outperform those based on Stokes ($aPLMM_S$, $aPNM_S$) by about an order of magnitude. This implies that rebuilding the monolithic preconditioners at the start of every Re increment is necessary to efficiently model turbulent flow. Another clear observation is that preconditioners based on PLMM perform twice as fast as those based on PNM. Lastly, smoother-only preconditioners are extremely

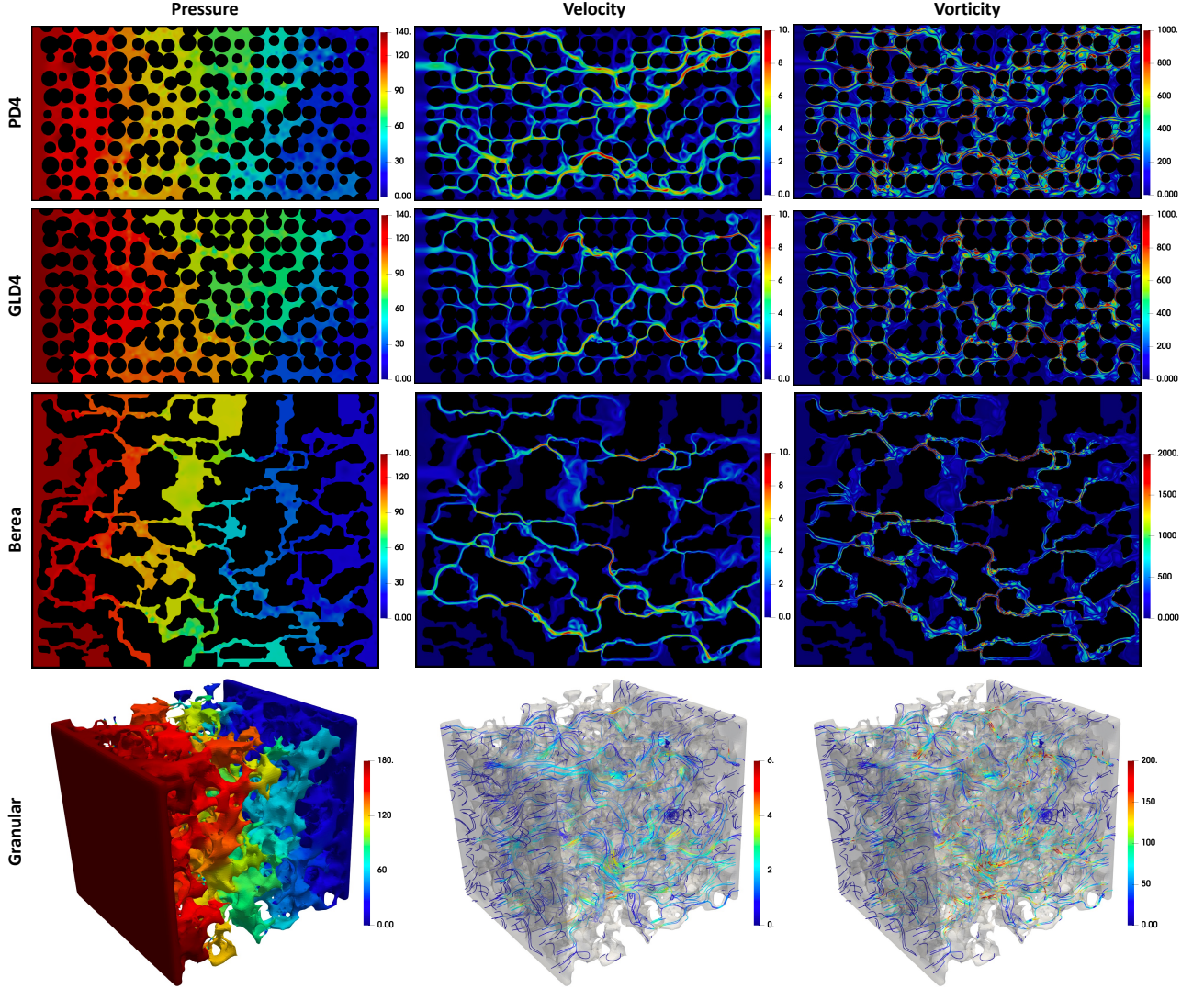


Figure 17: Spatial distributions of fluid pressure, velocity magnitude, and vorticity magnitude for the *unsteady-state Navier-Stokes equations* solved on the domains in Fig.5, namely, PD4 at $Re = 1.3 \times 10^5$, GLD4 at $Re = 5.5 \times 10^4$, Berea at $Re = 4.4 \times 10^4$, and Granular at $Re = 1.8 \times 10^4$.

slow (except in Granular) indicating a pairing with a coarse preconditioner M_G is necessary for rapid convergence.

Fig.19 plots the total wall-clock times (T_{tot}) spent on computations versus Re for all domains. The WCTs of all dt steps taken within each Re increment are summed for better representation of cost. The overall rise in T_{tot} with Re is due to the need for smaller dt in simulating turbulent flow at higher Re . By the same token, T_{sol} also increases with Re but the associated cost-per- dt is roughly constant, as evidenced by the constant number of GMRES iterations versus Re in Fig.18. Unlike Section 8.1, T_{tot} here is dominated by the cost of building the smoother (T_{ML} ; LU-decomposition of local systems in the additive-Schwarz smoothers), especially for gPLMM. The cost of rebuilding M_G is negligible in contrast. To better visualize speedup, Fig.20 normalizes the T_{tot} of all preconditioners against that of gPLMM. Once again, we see that the performance of aPLMM_{NS} is on par with gPLMM in 2D but slower by <3.4 times in 3D Granular. Moreover, gPLMM is faster than aPNM_{NS} by <3 times, aPLMM_S by <6 times, aPNM_S by <9 times, PLMM smoother by <15 times, and the PNM smoother by <30 times. Other trends regarding the relative ranking of preconditioners are consistent with observations already discussed in relation to GMRES iterations in Fig.18.

Fig.21 visualizes the 500 largest eigenvalues of the error-propagation matrix E in Eq.30 for the PD4 domain at

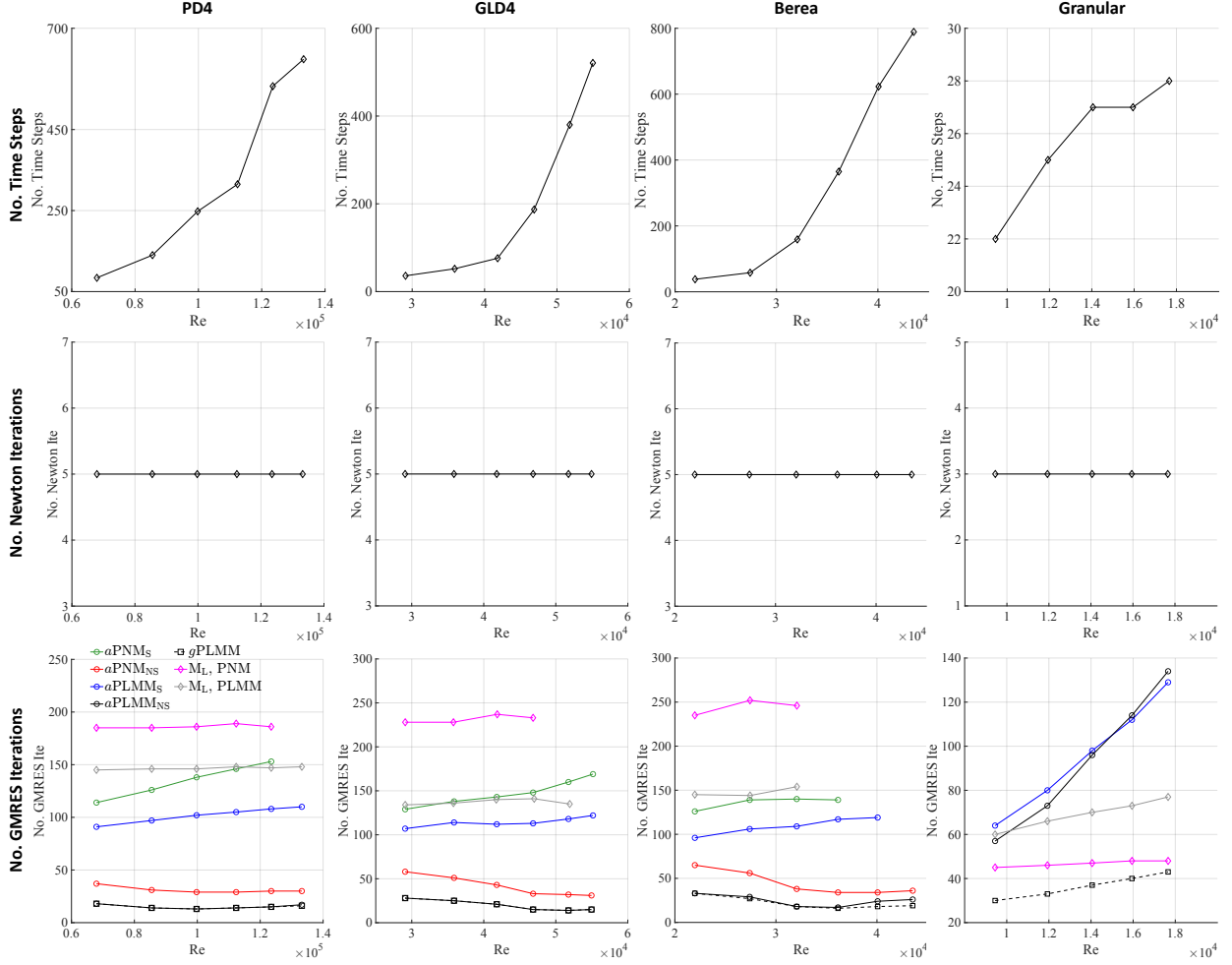


Figure 18: (Top row) Number of time steps dt , taken within each Re increment, versus Re . (Middle row) Number of Newton iterations of the first Newton step versus Re , averaged over each Re increment. (Bottom row) Number of GMRES iterations of the first Newton step versus Re , averaged over each Re increment. Results belong to different preconditioners based on PLMM and PNM applied to the *unsteady-state Navier-Stokes equations* in all domains. Prefix a means “algebraic,” and g means “geometric.” Subscripts “S” and “NS” imply preconditioners are built from Stokes or Navier Stokes, respectively.

$Re = 1.3 \times 10^5$. Included is also the spectrum of E_L , but not E_G because most of its eigenvalues for $gPLMM$, $gPLNM$, $aPLMM_S$ and $aPLNM_S$ lie far outside the unit circle. However, the E_G of $aPLMM_{NS}$ and $aPLNM_{NS}$ has eigenvalues that are either zero or one (see discussion in Section 8.1). The spectral radius of E for $gPLMM$ and $aPLMM_{NS}$ are similar, which explains their comparable GMRES convergence in Fig.18. The same observation holds for $gPLNM$ versus $aPLNM_{NS}$. Notice also that the Navier-Stokes-based preconditioners ($aPLMM_{NS}/aPLNM_{NS}$) have significantly reduced spectral radii than Stokes-based preconditioners ($aPLMM_S/aPLNM_S$), resulting in faster GMRES convergence. Finally, the spectral radius of E_L is almost one, confirming smoothers alone are inadequate for preconditioning GMRES.

9. Discussion

9.1. Coarse-scale solver

We presented a coarse-scale solver for computing approximate solutions to the steady-state Navier-Stokes equations in Algorithm 1, utilizing operators that comprise M_G . We found that the M_G of $gPLMM$ yields far more accurate results than that of $gPLNM$. Approximate solutions so obtained are sufficiently accurate ($< 30\%$ error in permeability) for use in a wide range of manufacturing (e.g., flow battery) and subsurface (e.g., CO_2 storage) applications. But as

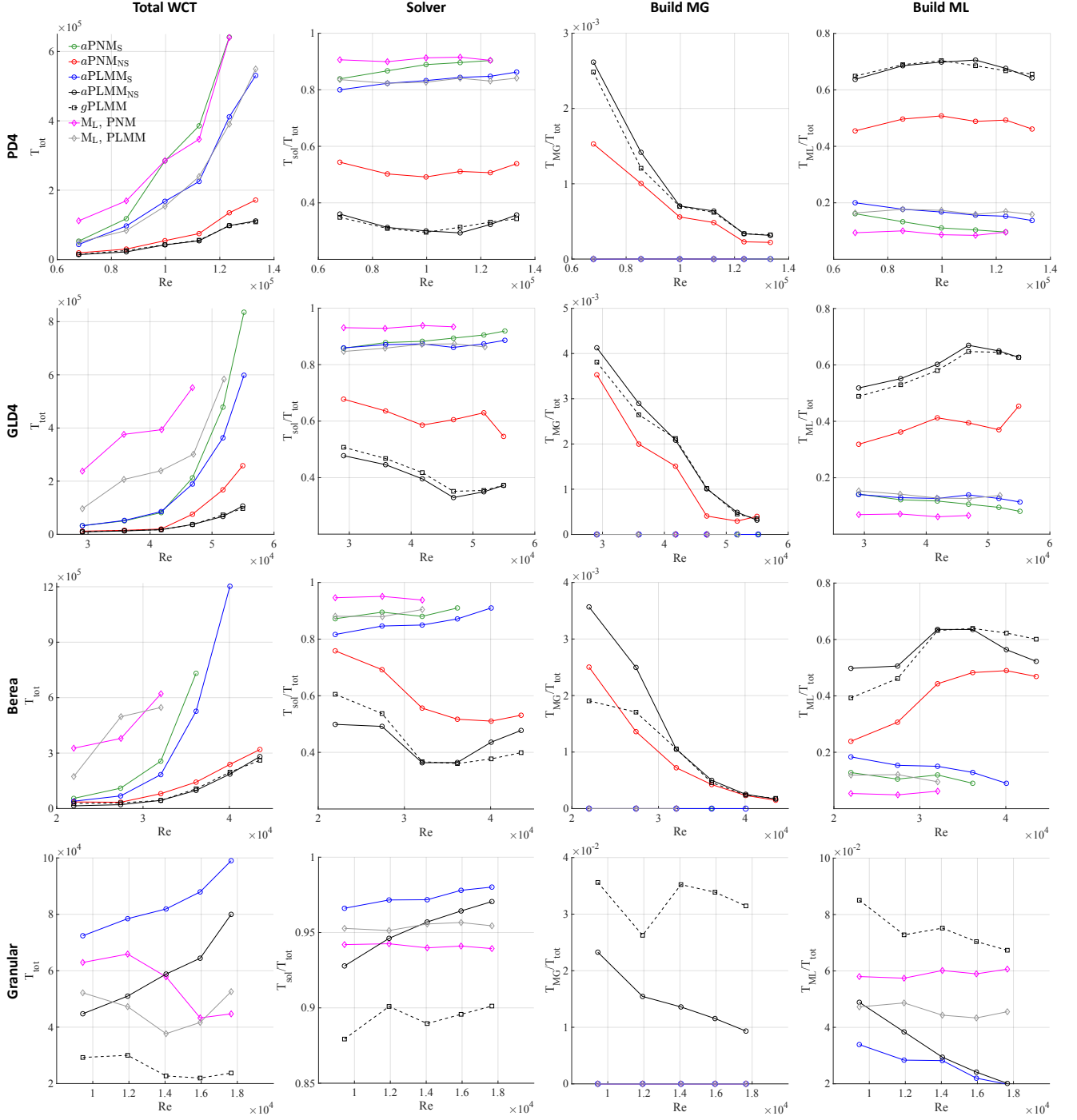


Figure 19: Total wall-clock times (WCT; in seconds) spent solving the *unsteady-state Navier-Stokes equations* (T_{tot}) versus Re , for different preconditioners based on PLMM and PNM (built from Stokes and Navier Stokes) in all domains. T_{tot} consists of the costs of building M_G (T_{MG} ; if monolithic), building M_L (T_{ML}), and self-time of GMRES (T_{sol}). The breakdown of T_{tot} into T_{MG} , T_{ML} , and T_{sol} is illustrated in fractions of T_{tot} .

Re grows, accuracy degrades because the closure $\partial_n \mathbf{u} = \mathbf{0}$ becomes invalid at contact interfaces. Fig.11 demonstrates this, where unlike Stokes flow, streamlines no longer strike normal to the interface at high Re due to the formation of vortices. Recall a key step in imposing the velocity closure was to remove the inertia term from the residual of

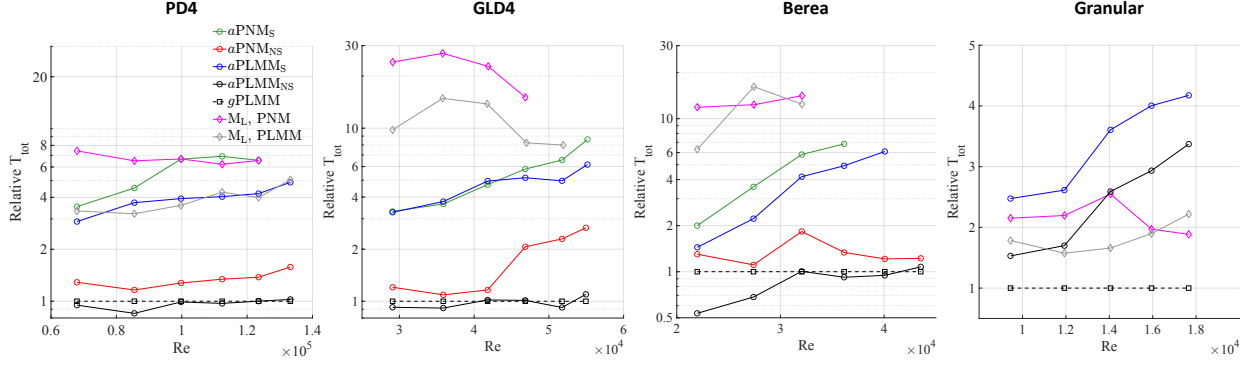


Figure 20: Relative performance of all monolithic and smoother-only preconditioners in Fig.19 measured against gPLMM for the *unsteady-state Navier-Stokes equations*. Relative performance is defined as the ratio of total WCT (T_{tot}) for the preconditioner in question over that of gPLMM.

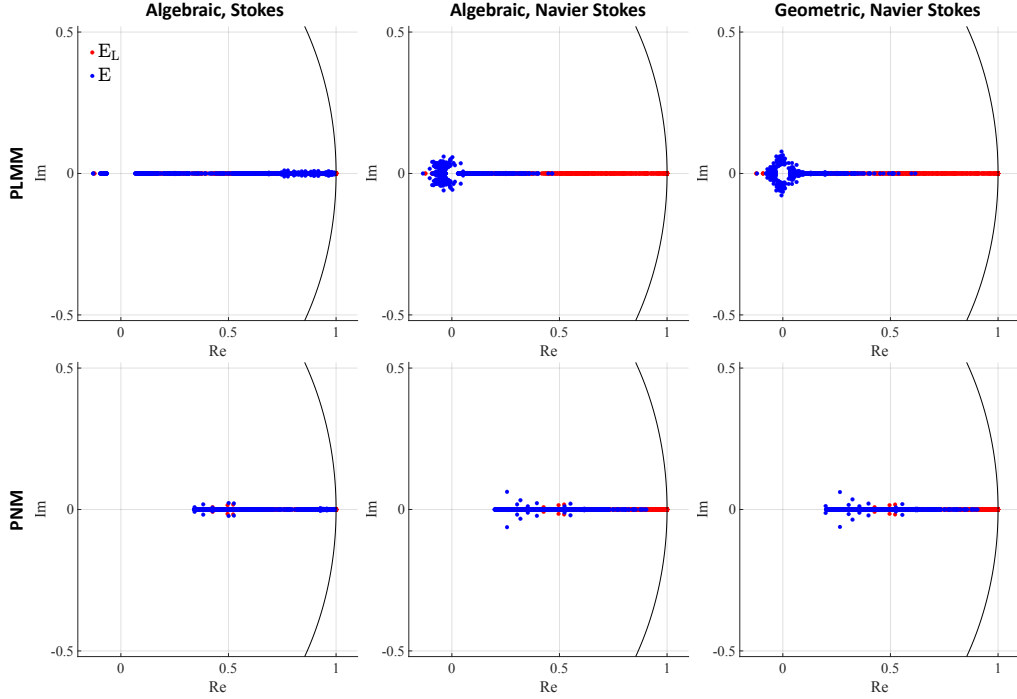


Figure 21: Spectra (500 largest eigenvalues) of the error-propagation matrices E (blue dots) and E_L (red dots) in Eq.30. Results belong to preconditioners $a\text{PLMM}_S$ and $a\text{PNM}_S$ (left column), $a\text{PLMM}_{NS}$ and $a\text{PNM}_{NS}$ (middle column), and gPLMM and gPNM (right column) applied to the *unsteady-state Navier-Stokes equations* on the PD4 domain at $Re = 1.3 \times 10^5$.

faces near each interface (Fig.3), so the column-sum operator $\mathcal{C}(\cdot)$ (in building $\hat{\mathbf{P}}$) imposes $\partial_n \mathbf{u} = \mathbf{0}$ exactly. Since all computations in Algorithm 1 involve only coarse-scale pressure unknowns defined on contact interfaces, with no recourse to the fine grid, the coarse-scale solver is much cheaper than computing solutions directly on the fine grid. Specifically, the T_{sol} and T_{ML} costs in Fig.14 are almost absent, rendering the coarse-scale solver faster by a factor of ~ 10 in 2D and ~ 2.5 in 3D Gyroid. However, Newton iterations of the coarse solver are larger by a factor of 2–5.

Two open questions remain: (1) Is there a more accurate closure at high Re and how does one impose it? (2) How can the coarse solver be generalized to time-dependent flows? The answer to (1) is unknown to the authors, but we suspect there is an opportunity to use mortars (similar to [30]) to increase the flexibility of capturing arbitrary velocities along each interface. This would require building additional shape vectors (Eq.16a) and solving a larger coarse system (Eq.24), as was done in [30] for elliptic equations. Whether the associated costs would render such an

approach prohibitive remains unknown. One answer to (2) involves discretizing $\partial_t \mathbf{u}$ in the Navier-Stokes equations with a fixed time step dt , before building $\mathbf{M}_G$. This embeds dt into the shape and correction vectors, which in turn can be used to march forward in time in increments of dt (similar to [25]). The drawback is committing to one dt for all t .

9.2. Recommended preconditioners

The best-performing preconditioner for both steady-state and unsteady-state Navier-Stokes equations is gPLMM, consistently across all domains studied in Section 8. This, incidentally, coincides with the more accurate coarse-scale solver based on gPLMM. The downside of gPLMM, however, is its need for geometric information, or more specifically, access to the code that can assemble the modified residual and Jacobian ($\tilde{\mathbf{r}}$ and $\tilde{\mathbf{J}}$ from Eq.22). Such access is needed to remove the inertia term from face-residuals near contact interfaces to impose $\partial_n \mathbf{u} = \mathbf{0}$. In the absence of such access, algebraic preconditioners are the only option, with aPLMM_S recommended for the steady-state and aPLMM_{NS} for the unsteady-state Navier-Stokes equations. The former has the benefit of being built only once and reused for all Re . We remind that block preconditioners ($b\text{AMG}$, $b\text{PLMM}$, and $b\text{PNM}$) were extremely prohibitive and eliminated immediately from further consideration. PNM-based preconditioners ($g\text{PNM}$, $a\text{PNM}_{S/NS}$) also exhibited very poor performance, indicating PLMM-based decomposition and closure are more accurate. Finally, smoothers alone are inadequate as preconditioners regardless of the steady-state or time-dependent form of the equations.

9.3. Computational complexity

While a serial machine was used here to perform all numerical tests, building and applying the coarse preconditioner $\mathbf{M}_G$ and the smoother $\mathbf{M}_L$ in all the proposed PLMM/PNM-based preconditioners are parallelizable. The cost of building $\mathbf{M}_G$ is dominated by computing shape vectors, $p_{c_j}^{p_i}$, and correction vectors, c^{p_i} in Eq.16a, whose computations are fully decoupled across all primary grids. Similarly, applying the additive-Schwarz smoother $\mathbf{M}_L$ requires solving a set of decoupled local systems on primary and dual grids, both amenable to parallelism. In the following, we briefly analyze the computational complexity and parallel scalability of constructing and applying $\mathbf{M}_G$ and $\mathbf{M}_L$.

Let Ω consist of N_f fine-scale unknowns, N^p primary grids, and $N^d (= N^c)$ dual grids. Let all linear(ized) systems (global and local) be solved using a linear solver that scales as $O(n^\beta)$, where n is the size of the linear system and $\beta \in (1, 3)$. The wall-clock times (WCTs) associated with building and applying $\mathbf{M}_G$ and $\mathbf{M}_L$ are:

$$\mathcal{T}_{\text{build}}^{\mathbf{M}_G} = O(N_f/N^p)^\beta \times (2N^d + N^p)/N_{\text{prc}} \quad (31a)$$

$$\mathcal{T}_{\text{apply}}^{\mathbf{M}_G} = O(N^d + N^p)^\beta \quad (31b)$$

$$\mathcal{T}_{\text{apply}}^{\mathbf{M}_L} = O\left[(N_f/N^p)^\beta \times N^p/N_{\text{prc}} + (N_f f^d/N^d)^\beta \times N^d/N_{\text{prc}}\right] \quad (31c)$$

where N_{prc} is the number of processors employed and f^d is the fraction of fine-scale unknowns associated with dual grids. Recall dual grids cover only a small portion of Ω (10–30%). Focusing on Eq.31a, the average cost of solving one local system on a primary grid is $O(N_f/N^p)^\beta$, of which $2N^d$ must be solved for shape vectors and N^p for correction vectors to build $\mathbf{M}_G$. These computations can all be performed in parallel, entailing N_{prc} can equal $2N^d + N^p$ at maximum. Focusing next on Eq.31b, one application of $\mathbf{M}_G$ requires solving a small coarse system of size $N^d + N^p$ (Eq.15), whose cost is negligible compared to other overheads. As for Eq.31c, the cost of applying $\mathbf{M}_L$ consists of solving N^p systems on primary grids and N^d systems on dual grids. If no pre-processing is done, the cost of building $\mathbf{M}_L$ is zero and all cost is due to application. But since applying $\mathbf{M}_L$ in iterative solvers requires repeated solves of the same local systems, LU-decomposition during a pre-processing stage can significantly speed up $\mathbf{M}_L$'s application. The construction cost of $\mathbf{M}_L$ then corresponds to these LU-decompositions, and the application cost in Eq.31c is lowered (by reducing β). Both the construction and application of $\mathbf{M}_L$ can be parallelized with N_{prc} up to $\max\{N^p, N^d\}$.

9.4. Challenges beyond linear preconditioning

Solving linear systems like Eq.2 during Newton iterations is the slowest step in solving the Navier-Stokes equations, which we focused on herein. With our best preconditioner (gPLMM), the cost of steady-state simulations is dominated by GMRES's self-time (T_{sol}), whereas that of dynamic simulations is dominated by $\mathbf{M}_L$'s build-time (T_{ML}). Further speedup requires advances in both solvers and smoothers, not coarse preconditioners. Potential smoothers that remain untested include block ILU(k), block Gauss-Seidel, and block additive-Schwarz based on PLMM or other.

Beyond linear preconditioning, the biggest challenge with solving the Navier-Stokes equations lies with Newton convergence. If the initial guess is far from the solution, Newton is very prone to divergence at high Re . What we have done here to reach high Re is essentially a method called *homotopy continuation* [41], where instead of solving the residual $r(\hat{x}; Re)$ in Eq.2 directly at Re (the semicolon separates variable from parameter), an augmented residual $H(\hat{x}, \lambda; Re) := r(\hat{x}; \lambda Re)$ is solved for gradually increasing values of $\lambda \in (0, 1)$. Notice $H(\hat{x}, 0; Re)$ is an easy problem because it corresponds to Stokes flow, whereas $H(\hat{x}, 1; Re) = r(\hat{x}; Re)$ is difficult as it corresponds to Navier-Stokes flow at the desired high Re . The approach we took to solve $H(\hat{x}, \lambda; Re)$ and to increase λ was to increment the inlet pressure p_{in} gradually from a low to a high value. This, in turn, implied an increment in Re , or equivalently λ . After solving $H(\hat{x}, \lambda; Re)$ for a given λ , the solution was used as the initial guess for Newton in the next λ . While effective, our approach is computationally wasteful because $H(\hat{x}, \lambda; Re)$ is solved to completion at each fixed λ . Better schemes, where λ is updated simultaneously as $\hat{x}$ in a single Newton loop, exist and would likely be much more efficient [42, 43].

The above continuation approach was found essential for the steady-state flow equations. However, for unsteady-state flow, it is needed only if the time step dt used to discretize the $\partial_t \mathbf{u}$ term in Eq.1 is large. Note that at the limit $dt \rightarrow \infty$, the discrete unsteady-state system reduces to state steady. But increasing dt too much introduces numerical diffusion into the solution, resulting in the inability to capture intricate details of the turbulent velocity field. In other words, the physics of the problem itself places an upper bound on dt , which is stricter at higher Re . If this bound is to be honored in favor of capturing the solution details accurately, then dt must necessarily be small and the system $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ in Eq.2 becomes diagonally dominant with respect to its (1,1)-block. The latter helps Newton converge with almost *any* initial guess, but practically, it is natural to use the solution from the prior time step, as was done here.

9.5. Future extensions

While our preconditioners target the Navier-Stokes equations for Newtonian fluid flow in a fixed geometry, their applicability extends to more challenging problems. In immiscible two-phase flow, governing equations consist of a momentum balance and a phase-evolution equation, solved in staggered fashion to march forward in time. The former is essentially Navier Stokes plus a source term for interfacial tension, whose solution is the main computational bottleneck [8]. We expect our preconditioners to accelerate solving the momentum equation without any modification to $\mathbf{M}_G$, as the source term will likely introduce high-frequency variations to the flow field. The phase-evolution equation is hyperbolic and discretized explicitly in time. It is, therefore, much cheaper to solve and errors are high-frequency in nature, which can be removed by a smoother alone (no $\mathbf{M}_G$; similar to the phase-field equation in fracture mechanics [36]). Another class of candidate problems exhibit evolving geometry, due either to mineral dissolution/precipitation [44], biofilm growth [45], or phase change [46]. If the representation of Γ_w is sharp (as is here), the decomposition of Ω into primary and dual grids must be updated periodically, which can be done locally and adaptively. But if Γ_w is diffuse, as in the Brinkman-Stokes model [47], extra source terms in Eq.1a capture geometric changes that get embedded into the shape/correction vectors of $\mathbf{M}_G$. Finally, the flow of non-Newtonian fluids (e.g., blood, polymer) may benefit from our preconditioners if applied to linearized forms of the governing equations [48].

10. Conclusions

We present monolithic preconditioners to accelerate the convergence of linear solvers applied to the Navier-Stokes equations in arbitrarily complex porous microstructures. They are based on the pore-level multiscale method (PLMM) and the pore network model (PNM), and they generalize previously proposed preconditioners for the Stokes equations [24]. The preconditioners are two-level, consisting of a coarse preconditioner $\mathbf{M}_G$ and a smoother $\mathbf{M}_L$, designed to attenuate low- and high-frequency errors, respectively. To build both, the computational domain is decomposed into subdomains (primary and dual grids), over which local problems are solved. We formulate algebraic (*a*PLMM and *a*PNM) and geometric (*g*PLMM and *g*PNM) versions of each preconditioner, with the key difference being that the closure BC $\partial_n \mathbf{u} = \mathbf{0}$ is imposed exactly on subdomain interfaces in the geometric form but only approximately in the algebraic form. Hence, geometric preconditioners perform better but require access to code assembling the Jacobian to modify it slightly. Algebraic preconditioners need no such access and operate directly on the linear system.

We test our preconditioners against a wide range of 2D and 3D porous geometries, different Re , and steady-state and time-dependent forms of the Navier-Stokes equations. We also benchmark them against state-of-the-art block preconditioners *b*AMG, *b*PLMM, and *b*PNM. The best-performing preconditioner across the board was *g*PLMM. But

since it requires intrusive access to code, comparable performance is possible with $a\text{PLMM}_S$ for steady-state Navier Stokes and $a\text{PLMM}_{NS}$ for time-dependent Navier Stokes. Subscripts “S” and “NS” indicate the preconditioners are built from the Jacobians of the Stokes and Navier-Stokes equations. Given the Stokes equations are linear, $a\text{PLMM}_S$ need only be built once for usage at other Re , whereas $a\text{PLMM}_{NS}$ (and $g\text{PLMM}$) must be built periodically at a nearby Re if Re changes, to retain optimal performance. Block and PNM-based preconditioners are far from competitive.

Lastly, we formulate a coarse-scale solver based on $g\text{PLMM}$ for the steady-state Navier-Stokes equations, which computes approximate solutions rapidly. Acceleration is achieved by conducting all computations on coarse-scale pressure unknowns defined at subdomain interfaces, without any recourse to the fine grid. The coarse solution can be mapped onto the fine grid using a prolongation operator inherent to $g\text{PLMM}$. Approximate solutions so obtained have acceptable accuracy for a range of applications (permeability error < 30%). Our coarse solver is more accurate than classical PNMs designed to capture inertial flow in pore-scale domains, simplified by a ball-and-stick representation.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-2145222. We acknowledge the Institute for Computational and Data Sciences (ICDS) at Penn State University for access to computational resources.

Appendix A. Validating the fine-grid solver

We validate the fine-grid solver used to assemble the residual and Jacobian of the Navier-Stokes equations, thus produce all results in this work. We consider fluid flow around circular and square obstacles at different Re . Fig.A.1 shows the 2D domains, which consist of rectangles of length $L=3$ cm and height $H=2$ cm. The centroids of the circle and square are positioned at half-height and a distance of 1 cm from the left boundary. The side length of the square and diameter of the circle are both $D=0.2$ cm. The BCs consists of: (1) inflow on the left ($\mathbf{u} = (10, 0)$ and $\partial_n p = 0$); (2) outflow on the right ($\partial_n \mathbf{u} = \mathbf{0}$ and $p = 0$); (3) slip on top and bottom ($\partial_n \mathbf{u} = \mathbf{0}$ and $\partial_n p = 0$); and (4) no-slip on circle and square perimeters ($\mathbf{u} = \mathbf{0}$ and $\partial_n p = 0$). Fig.A.1 shows the pressure, velocity magnitude, and vorticity magnitude over the whole domain at a low and high Re . A von Karman vortex street is seen to form in the latter.

To validate the fine-scale solver, we compute drag coefficient:

$$C_D = \frac{2F_d}{\rho u_{in}^2 D} \quad (\text{A.1})$$

on the obstacles at different $Re = \rho u_{in} D / \mu$ and compare them to literature values in Table A.1. In Eq.A.1, F_d is the drag force exerted on the obstacles, and $u_{in} = 10$ is the inlet velocity imposed in the horizontal direction. Table.A.1 shows the C_D computed here agrees with many other reported values in the literature for Re between 20 and 300.

Table A.1: Comparison of drag coefficient (C_D) computed here against literature for flow around a circle and square at different Re .

	Reynolds number	This work, C_D	Literature, C_D
Cylinder	$Re = 20$	2.18	2.00 [49], 2.09 [50], 2.20 [51]
	$Re = 40$	1.63	1.62 [52], 1.53 [50], 1.60 [53]
	$Re = 100$	1.37	1.35 [54], 1.39 [55], 1.35 [56]
	$Re = 300$	1.32	1.38 [56], 1.28 [51], 1.37 [57]
Square	$Re = 100$	1.57	1.59 [58], 1.51 [59]
	$Re = 250$	1.71	1.66 [56], 1.69-1.72 [60]

References

- [1] Jacob Bear. Dynamics of fluids in porous media. Courier Corporation, 2013.
- [2] Peisheng Liu and Guo-Feng Chen. Porous materials: processing and applications. Elsevier, 2014.

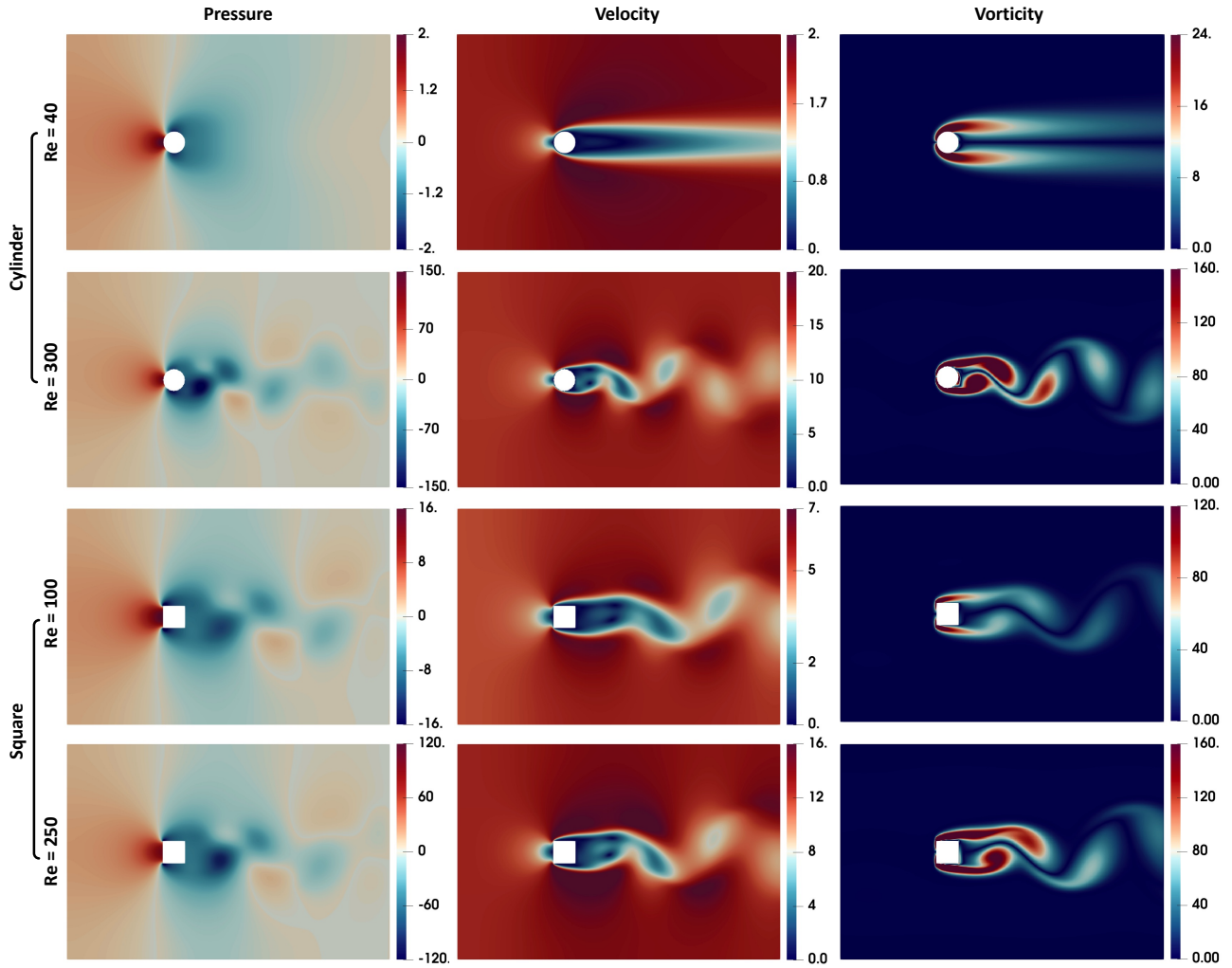


Figure A.1: Pressure, velocity magnitude, and vorticity magnitude for flow around circular and square obstacles at different Re .

- [3] Dorte Wildenschild and Adrian P Sheppard. X-ray imaging and analysis techniques for quantifying pore-scale structure and processes in subsurface porous medium systems. *Advances in Water Resources*, 51:217–246, 2013.
- [4] Blair Perot. Conservation properties of unstructured staggered mesh schemes. *Journal of Computational Physics*, 159(1):58–89, 2000.
- [5] Francis H Harlow and J Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids*, 8(12):2182–2189, 1965.
- [6] Pavel B Bochev, Clark R Dohrmann, and Max D Gunzburger. Stabilization of low-order mixed finite elements for the stokes equations. *SIAM Journal on Numerical Analysis*, 44(1):82–101, 2006.
- [7] Daniel Loghin and AJ Wathen. Schur complement preconditioners for the navier–stokes equations. *International journal for numerical methods in fluids*, 40(3-4):403–412, 2002.
- [8] Yashar Mehmani and Hamdi A Tchelepi. Multiscale formulation of two-phase flow at the pore scale. *Journal of Computational Physics*, 389: 164–188, 2019.
- [9] Stefan Bachu. Co2 storage in geological media: Role, means, status and barriers to deployment. *Progress in energy and combustion science*, 34(2):254–273, 2008.
- [10] Angela Goodman Hanson, Barbara Kutchko, Greg Lackey, Djuna Gulliver, Brian R Strazisar, Kara A Tinker, Foad Haeri, Ruishu Wright, Nicolas Huerta, Seunghwan Baek, et al. Subsurface hydrogen and natural gas storage: State of knowledge and research recommendations report. 2022.
- [11] Enrico Barbier. Geothermal energy technology and current status: an overview. *Renewable and sustainable energy reviews*, 6(1-2):3–65, 2002.
- [12] Martin Andersson, SB Beale, M Espinoza, Z Wu, and W Lehnert. A review of cell-scale multiphase flow modeling, including water management, in polymer electrolyte fuel cells. *Applied Energy*, 180:757–778, 2016.

- [13] Jason K Lee, ChungHyuk Lee, Kieran F Fahy, Pascal J Kim, Kevin Krause, Jacob M LaManna, Elias Baltic, David L Jacobson, Daniel S Hussey, and Aimy Bazylak. Accelerating bubble detachment in porous transport layers with patterned through-pores. *ACS Applied Energy Materials*, 3(10):9676–9684, 2020.
- [14] TW Clyne, IO Golosnoy, JC Tan, and AE Markaki. Porous materials for thermal management under extreme conditions. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 364(1838):125–146, 2006.
- [15] Yousef Saad. Iterative methods for sparse linear systems, volume 82. *siam*, 2003.
- [16] Michele Benzi, Gene H Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta numerica*, 14:1–137, 2005.
- [17] John W Ruge and Klaus Stüben. Algebraic multigrid. In *Multigrid methods*, pages 73–130. SIAM, 1987.
- [18] Yvan Notay. An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis*, 37(6):123–146, 2010.
- [19] David Silvester, Howard Elman, David Kay, and Andrew Wathen. Efficient preconditioning of the linearized navier–stokes equations for incompressible flow. *Journal of Computational and Applied Mathematics*, 128(1-2):261–279, 2001.
- [20] Howard Elman and David Silvester. Fast nonsymmetric iterations and preconditioning for navier–stokes equations. *SIAM Journal on Scientific Computing*, 17(1):33–46, 1996.
- [21] Howard C Elman. Preconditioning for the steady-state navier–stokes equations with low viscosity. *SIAM Journal on Scientific Computing*, 20(4):1299–1316, 1999.
- [22] David Kay, Daniel Loghin, and Andrew Wathen. A preconditioner for the steady-state navier–stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256, 2002.
- [23] Howard Elman, Victoria E Howle, John Shadid, Robert Shuttleworth, and Ray Tuminaro. Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing*, 27(5):1651–1668, 2006.
- [24] Yashar Mehmani and Kangan Li. Multiscale preconditioning of stokes flow in complex porous geometries. *Journal of Computational Physics*, 521:113541, 2025.
- [25] Yashar Mehmani and Hamdi A Tchelepi. Multiscale computation of pore-scale fluid dynamics: Single-phase flow. *Journal of Computational Physics*, 375:1469–1487, 2018.
- [26] Irving Fatt. The network model of porous media. *Trans. AIME*, 207:144–159, 1956.
- [27] Serge Beucher and Christian Lantuéjoul. Use of watersheds in contour detection. In *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation*, Rennes, France, 1979.
- [28] Alexander Heinlein, Christian Hochmuth, and Axel Klawonn. Reduced dimension gds coarse spaces for monolithic schwarz domain decomposition methods for incompressible fluid flow problems. *International Journal for Numerical Methods in Engineering*, 121(6):1101–1119, 2020.
- [29] Alexander Heinlein, Axel Klawonn, Jascha Knepper, and Lea Saßmannshausen. Monolithic and block overlapping schwarz preconditioners for the incompressible navier–stokes equations. *arXiv preprint arXiv:2506.16179*, 2025.
- [30] Sabit Mahmood Khan and Yashar Mehmani. High-order multiscale preconditioner for elasticity of arbitrary structures. *arXiv preprint arXiv:2509.19777*, 2025.
- [31] F Thauvin and KK Mohanty. Network modeling of non-darcy flow through porous media. *Transport in Porous Media*, 31(1):19–37, 1998.
- [32] Matthew T Balhoff and Mary F Wheeler. A predictive pore-scale model for non-darcy flow in porous media. *Spe Journal*, 14(04):579–587, 2009.
- [33] Maziar Veyskarami, Amir Hossein Hassani, and Mohammad Hossein Ghazanfari. A new insight into onset of inertial flow in porous media using network modeling with converging/diverging pores. *Computational Geosciences*, 22(1):329–346, 2018.
- [34] Yashar Mehmani, Nicola Castelletto, and Hamdi A Tchelepi. Multiscale formulation of frictional contact mechanics at the pore scale. *Journal of Computational Physics*, 430:110092, 2021.
- [35] Kangan Li, Sabit Mahmood Khan, and Yashar Mehmani. Machine learning for preconditioning elliptic equations in porous microstructures: A path to error control. *Computer Methods in Applied Mechanics and Engineering*, 427:117056, 2024.
- [36] Kangan Li and Yashar Mehmani. A multiscale preconditioner for crack evolution in porous microstructures: Accelerating phase-field methods. *International Journal for Numerical Methods in Engineering*, 125(11):e7463, 2024.
- [37] Edo S Boek and Maddalena Venturoli. Lattice-boltzmann studies of fluid flow in porous media with realistic rock geometries. *Computers & Mathematics with Applications*, 59(7):2305–2314, 2010.
- [38] Alan Hugh Schoen. *Infinite periodic minimal surfaces without self-intersections*, volume 5541. National Aeronautics and Space Administration, 1970.
- [39] Long Chen. *iFEM: an integrated finite element methods package in MATLAB*. Technical report, 2009. URL <https://github.com/lyc102/ifem>.
- [40] Lloyd N Trefethen and Mark Embree. Spectra and pseudospectra: the behavior of nonnormal matrices and operators. 2020.
- [41] Layne T Watson. Globally convergent homotopy algorithms for nonlinear systems of equations. *Nonlinear Dynamics*, 1(2):143–191, 1990.
- [42] Jiamin Jiang and Hamdi A Tchelepi. Dissipation-based continuation method for multiphase flow in heterogeneous porous media. *Journal of Computational Physics*, 375:307–336, 2018.
- [43] RM M Younis, HA A Tchelepi, and Khalid Aziz. Adaptively localized continuation-newton method-nonlinear solvers that converge all the time. *SPE Journal*, 15(02):526–544, 2010.
- [44] Catherine Noiriél, Carl I Steefel, Li Yang, and Dominique Bernard. Effects of pore-scale precipitation on permeability and flow. *Advances in water resources*, 95:125–137, 2016.
- [45] Jayde A Aufrecht, Jason D Fowlkes, Amber N Bible, Jennifer Morrell-Falvey, Mitchel J Doktycz, and Scott T Retterer. Pore-scale hydrodynamics influence the spatial evolution of bacterial biofilms in a microfluidic porous network. *PloS one*, 14(6):e0218316, 2019.
- [46] Adrian Moure and Xiaojing Fu. A phase-field model for wet snow metamorphism. *Crystal Growth & Design*, 24(19):7808–7821, 2024.
- [47] Cyprien Soulaïne and Hamdi A Tchelepi. Micro-continuum approach for pore-scale simulation of subsurface processes. *Transport in porous media*, 113(3):431–456, 2016.
- [48] Marcel J Crochet, Arthur Russell Davies, and Kenneth Walters. *Numerical simulation of non-Newtonian flow*, volume 1. Elsevier, 2012.
- [49] Bengt Fornberg. A numerical study of steady viscous flow past a circular cylinder. *Journal of Fluid Mechanics*, 98(4):819–855, 1980.

- [50] Gregory J Sheard, Kerry Hourigan, and Mark Christopher Thompson. Computations of the drag coefficients for low-reynolds-number flow past rings. *Journal of Fluid Mechanics*, 526:257–275, 2005.
- [51] C von Wieselsberger. Neuere feststellungen unter die gesetze des flussigkeits und luftwiderstandes. *Phys. z.*, 22:321–328, 1921.
- [52] Donna Calhoun. A cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *Journal of computational physics*, 176(2):231–275, 2002.
- [53] David Russell and Z Jane Wang. A cartesian grid method for modeling multiple moving objects in 2d incompressible viscous flow. *Journal of Computational Physics*, 191(1):177–205, 2003.
- [54] Patrick Bruno Beaudan. *Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number*. Stanford University, 1995.
- [55] ALF Lima E Silva, A Silveira-Neto, and JJR Damasceno. Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method. *Journal of Computational Physics*, 189(2):351–370, 2003.
- [56] Alex Main and Guglielmo Scovazzi. The shifted boundary method for embedded domain computations. part ii: Linear advection–diffusion and incompressible navier–stokes equations. *Journal of Computational Physics*, 372:996–1026, 2018.
- [57] BN Rajani, A Kandasamy, and Sekhar Majumdar. Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modelling*, 33(3):1228–1247, 2009.
- [58] Taha Khademinezhad, Pouyan Talebizadeh, and Hassan Rahimzadeh. Numerical study of unsteady flow around a square cylinder in compare with circular cylinder. In *Conference Paper*, 2015.
- [59] K Lam, YF Lin, L Zou, and Yang Liu. Numerical study of flow patterns and force characteristics for square and rectangular cylinders with wavy surfaces. *Journal of Fluids and structures*, 28:359–377, 2012.
- [60] Arun K Saha, Gautam Biswas, and Krishnamurthy Muralidhar. Three-dimensional study of flow past a square cylinder at low reynolds numbers. *International Journal of Heat and Fluid Flow*, 24(1):54–66, 2003.