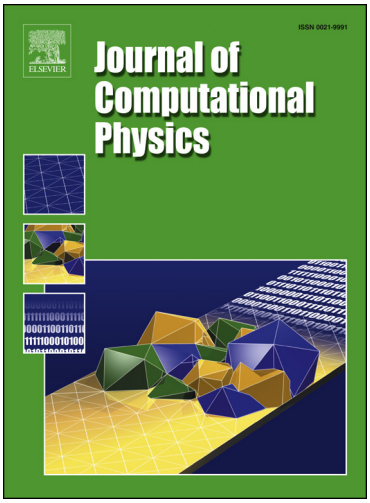
Journal Pre-proof



Multiscale Preconditioning of Stokes Flow in Complex Porous Geometries

Yashar Mehmani and Kangan Li

PII: S0021-9991(24)00789-7
DOI: https://doi.org/10.1016/j.jcp.2024.113541
Reference: YJCPH 113541

To appear in: *Journal of Computational Physics*

Received date: 7 December 2023
Revised date: 23 August 2024
Accepted date: 25 October 2024

Please cite this article as: Y. Mehmani and K. Li, Multiscale Preconditioning of Stokes Flow in Complex Porous Geometries, *Journal of Computational Physics*, 113541, doi: https://doi.org/10.1016/j.jcp.2024.113541.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2024 Published by Elsevier.

Highlights

- Multiscale preconditioners formulated for Stokes flow in porous microstructures.
- Performance is superior to algebraic multigrid variants in complex 2D/3D geometries.
- Monolithic PLMM preconditioner algebraizes the pore-level multiscale method.
- Monolithic PNM preconditioner algebraizes the established pore network model.
- All preconditioners are scalable in parallel and can be implemented non-intrusively.

Journal Pre-proof

Multiscale Preconditioning of Stokes Flow in Complex Porous Geometries

Yashar Mehmani[a], Kangan Li[a]

[a]*Energy and Mineral Engineering Department, The Pennsylvania State University, University Park, Pennsylvania 16802*

Abstract

Fluid flow through porous media is central to many subsurface (e.g., CO_2 storage) and industrial (e.g., fuel cell) applications. The optimization of design and operational protocols, and quantifying the associated uncertainties, requires fluid-dynamics simulations inside the microscale void space of porous samples. This often results in large and ill-conditioned linear(ized) systems that require iterative solvers, for which preconditioning is key to ensure rapid convergence. We present robust and efficient preconditioners for the accelerated solution of saddle-point systems arising from the discretization of the Stokes equation on geometrically complex porous microstructures. They are based on the recently proposed pore-level multiscale method (PLMM) and the more established reduced-order method called the pore network model (PNM). The four preconditioners presented are the monolithic PLMM, monolithic PNM, block PLMM, and block PNM. Compared to existing block preconditioners, accelerated by the algebraic multigrid method, we show our preconditioners are far more robust and efficient. The monolithic PLMM is an algebraic reformulation of the original PLMM, which renders it portable and amenable to non-intrusive implementation in existing software. Similarly, the monolithic PNM is an algebraization of PNM, allowing it to be used as an accelerator of direct numerical simulations (DNS). This bestows PNM with the, heretofore absent, ability to estimate and control prediction errors. The monolithic PLMM/PNM can also be used as approximate solvers that yield globally flux-conservative solutions, usable in many practical settings. We systematically test all preconditioners on 2D/3D geometries and show the monolithic PLMM outperforms all others. All preconditioners can be built and applied on parallel machines.

Keywords: Stokes equation, Porous media, Multiscale method, Preconditioning, Pore network, Krylov solver

1. Introduction

We concern ourselves with the flow of a single-phase, incompressible, Newtonian fluid governed by the Stokes equation inside a porous microstructure. This is relevant to several subsurface and manufacturing applications including the geologic sequestration of CO_2 [1], seasonal storage of H_2 underground [2], extraction of geothermal heat [3], and designing new porous transport layers for fuel cells and electrolyzers [4, 5]. To optimize design and operational protocols in these applications, and to quantify their associated uncertainties, rapid simulations of fluid flow inside the microscopic void space of porous samples is required. The latter is often captured in great detail by a pore-scale *image* using, e.g., an X-ray μCT scanner [6]. The computational domains so obtained tend to be rather large, to represent spatial and statistical variabilities, and they exhibit extreme geometric complexity. The images are then used as direct inputs to pore-scale computations using various direct numerical simulation (DNS) methods (e.g., FVM, MFEM).

In DNS, the Stokes equation is discretized and solved over a fine grid that often coincides with an integer fraction of the pixels in the pore-scale image. Because conservation of flux is a desired property for subsequent solute transport (and two-phase flow) simulations [7], the staggered finite volume (FVM) [8], marker and cell (MAC) finite difference [9], and mixed finite element (MFEM) [10, 11] methods are popular DNS approaches. Their disretization results in a large, sparse, linear system in terms of pressure, p, and velocity, $\boldsymbol{u}$, unknowns that exhibits a saddle-point structure; wherein the (2,2)–block is zero (except when stabilization is used in MFEM). Solving such systems requires iterative

*Corresponding author: Yashar Mehmani. Email: yzm5192@psu.edu
Email addresses: yzm5192@psu.edu (Yashar Mehmani), kbl5610@psu.edu (Kangan Li)

Journal Pre-proof

(e.g., Krylov) solvers, for which preconditioning is a must to ensure rapid convergence. Our goal is to develop robust and efficient preconditioners for the Stokes equation defined on geometrically complex porous domains.

Existing preconditioners for the (Navier-)Stokes equation are based on splitting the saddle-point system, $\hat{\mathrm{A}}\hat{x}=\hat{b}$ where $\hat{x}=[\boldsymbol{u}, p]^{\top}$, into two sub-systems: one in terms of $\boldsymbol{u}$ with a coefficient matrix F, and another in terms of p with a coefficient matrix equal to the Schur complement $\mathrm{S}=-\mathrm{GF}^{-1}\mathrm{G}^{\top}$ [12]. Here, G is the discrete divergence operator. Preconditioning involves solving each sub-system *approximately*. Since F is the velocity Laplacian, its approximate solution is obtained efficiently via algebraic multigrid (AMG) methods [13]. However, S poses challenges because of the F^{-1} term sandwiched in between G and $\mathrm{G}^{\top}$. Therefore, existing preconditioners differ primarily in the way S or S^{-1} is approximated [11, 14–18], for example by replacing F^{-1} with its diagonal or commuting F^{-1} and $\mathrm{G}^{\top}$ [12]. In the well-known BFBt preconditioner [14], the sub-system associated with S is replaced by two others that involve the pressure Laplacian $\mathrm{GG}^{\top}$, hence allowing their efficient solution via AMG. Here, we refer to all such preconditioners based on approximating S as *block AMG*, since they are often formulated as block-triangular matrices. Almost all are tested exclusively on very simple geometries in the literature (e.g., lid-driven cavity flow, flow past an obstacle/step), which are not representative of the complex porous domains, with poorly conditioned $\hat{\mathrm{A}}$, that interest us here. We note ill-conditioning amplifies due to poorly connected void spaces and/or very fine grids to capture geometric details.

Our main contribution is the formulation of two *monolithic* preconditioners that outperform the block AMG variants above: *monolithic PLMM* and *monolithic PNM*. The term "monolithic" means that the $\boldsymbol{u}$ and p unknowns are *not* split but solved together. This preserves certain coupling terms that leads to the faster convergence of iterative solvers. The monolithic PLMM preconditioner is based on an algebraic reformulation of the pore-level multiscale method (PLMM), originally proposed by the authors in [19]. PLMM yields highly accurate approximate solutions to the Stokes equation by executing three steps: (1) Decompose the void space, Ω, into *primary grids*, Ω_i^p; (2) Solve local basis/correction problems on each Ω_i^p; (3) Couple the local problems with a global, coarse-scale, interface problem. The approximate, or *first-pass*, solution is obtained by interpolating the coarse-scale solution on the fine grid via the basis/correction functions. Its errors tend to concentrate near the interfaces between adjacent primary grids, and can be reduced by solving a second set of local problems over so-called *dual grids*, Ω_i^d, which cover a thin region around each interface. Despite its accuracy and parallel scalability, the geometric variant of PLMM in [19] suffers from two drawbacks: (1) To implement it in existing codes requires significant development time and effort; (2) In high-porosity domains, very large dual grids can form (due to the merger of overlapping dual grids), which makes the cost of local problems solved on them prohibitive. The monolithic PLMM preconditioner herein removes both drawbacks.

The monolithic PNM preconditioner is based on an algebraic reformulation of a popular (and old) reduced-order method in the porous-media literature known as the pore network model (PNM) [20]. In PNM, the complex void space geometry is simplified by a computational graph, whose nodes approximate the *pore* shapes (via, e.g., sphere, cubes) and edges approximate the *throat* shapes (via, e.g., cylinders, prisms). The terms "pore" and "throat" refer to geometric enlargements and constrictions of the void, respectively. PNM is widely used due to its low computational cost, but its major drawback lies in its inability to estimate and control prediction errors [21]. The latter renders PNM unreliable when applied to new microstructures. This trade-off between cost and accuracy has served as a major deciding factor when using PNM versus DNS in different applications [22, 23]. Moreover, PNM requires the explicit extraction of a network from a pore-scale image, which can be computationally costly [6, 24]. The monolithic PNM preconditioner not only removes these drawbacks, but offers a different perspective altogether: instead of having to choose between PNM and DNS, *use PNM to accelerate DNS*. Thereby, error estimation and control come for free.

Aside from faster convergence in Krylov solvers, the monolithic PLMM/PNM have two other advantages over the block AMG preconditioners: (1) They can be used as approximate solvers if applied inside a Richardson-type loop. Each iteration is globally flux-conservative, and can be made locally (or pointwise) so by solving a special (inexpensive) dual-grid problem outlined in [7] (not pursued). The first iterate, in particular, is the first-pass solution given by the geometric PLMM/PNM; (2) At no extra cost, the monolithic PLMM/PNM can be recast as block preconditioners that perform much more robustly and efficiently within Krylov solvers than block AMG preconditioners. Concretely, instead of using AMG to solve the sub-systems associated with F and S, we formulate prolongation matrices derived directly from their monolithic counterparts. We call the resulting preconditioners *block PLMM* and *block PNM*.

We systematically test the monolithic/block PLMM/PNM preconditioners proposed herein against block AMG for different 2D/3D porous microstructures. The Schur complement matrix S in *all* block preconditioners is approximated with the scaled-BFBt method [15], but other approaches are also possible and discussed. Our main findings are: (1) Monolithic preconditioners perform better than block preconditioners in Krylov solvers; (2) Both as a monolithic and

Journal Pre-proof

block preconditioner, PLMM is superior to PNM; (3) As approximate solvers, the first-pass solution of monolithic PLMM is an order of magnitude more accurate than monolithic PNM, consistent with previous comparisons involving the geometric formulations of PLMM and PNM in [19]; (4) Block PLMM/PNM are superior preconditioners to block AMG, as the latter diverges in over half of the domains tested; (5) The algebraic reformulations of PLMM and PNM herein enable their non-intrusive use in existing codes; And (6) the monolithic/block PLMM/PNM preconditioners can be built and applied on parallel machines. A rigorous parallel scalability analysis is not performed but discussed.

The algorithmic details of the monolithic PLMM and PNM, and by extension block PLMM and PNM, are almost identical, implying the same code can be used to build both. The difference lies in the way PLMM and PNM decompose a given void geometry. In PLMM, primary grids coincide with physical pores and dual grids with physical throats. The interfaces between the primary grids occur at geometric constrictions. By contrast, in PNM, primary grids coincide with throats and dual grids with pores, and interfaces occur at geometric enlargements. The reason why PLMM outperforms PNM both as a monolithic/block preconditioner and an approximate solver here, and geometric model in [19], is because of this decomposition and associated local (closure) BCs used to build the basis/correction functions. PNM assumes that the pressure field inside each pore is approximately constant, whereas PLMM makes this assumption only along each interface. The optimality of the latter is analyzed in [19] and discussed later.

We conclude by noting that the first-pass solution from the monolithic PNM is more accurate than approximate solutions produced by current PNM schemes in the literature [22]. This is because, unlike classical PNM, the first-pass solution here: (1) does *not* simplify the geometry of the void space; and (2) allows reconstructing a *pointwise* velocity field over the entire fine grid, instead of just yielding integrated flowrates through the throats. Finally, our monolithic PLMM preconditioner extends the variants in [25, 26] for linear-elastic fracture mechanics to fluid dynamics.

The paper's outline is as follows: Section 2 describes the governing equations and linear system to be solved. Section 3 presents the monolithic PLMM/PNM preconditioners. We begin with PLMM and discuss its overall structure, its domain decomposition, followed by the formulations of its global preconditioner and local smoother. Since the mathematics of the monolithic PNM are very similar, we only highlight substantive differences versus the monolithic PLMM. Section 4 presents the block PLMM/PNM preconditioners including an existing block AMG used as a benchmark. Section 5 describes the validation set considered for testing the proposed preconditioners in Section 6. We discuss the implications of the results in Section 7 and conclude with a summary of our findings in Section 8.

2. Problem statement

We consider the Stokes equation governed by:

$$\mu\,\Delta\boldsymbol{u} - \nabla p = 0 \tag{1a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{1b}$$

Eq.1a imposes momentum conservation, and Eq.1b mass conservation on a single-phase, incompressible, Newtonian fluid. We are interested in solving Eq.1 on a complex, porous geometry, Ω, like the white region shown in Fig.1a. No-slip boundary conditions (BCs) are imposed on the fluid-solid interface, Γ_w, and inlet/outlet BCs are imposed on the left/right sides of Ω (i.e., flow is from left to right). The no-slip BCs involve setting $\boldsymbol{u}=0$ and $\partial p/\partial n=0$ on Γ_w, where $\boldsymbol{n}$ is the unit normal on the fluid-solid interface. The inlet/outlet BCs involve imposing constant pressure and zero normal gradient of velocity, $\partial\boldsymbol{u}/\partial n=0$, where $\boldsymbol{n}$ is again the unit normal on the inlet or outlet boundary.

Several approaches for discretizing Eq.1 on a fine grid exist. Among those that are flux-conservative up to machine precision, the finite volume method (FVM), marker and cell (MAC) finite difference method, and mixed finite element method (MFEM) are popular. Here, we adopt the MAC scheme [9] applied to a Cartesian fine grid, like the one shown in Fig.2c. Pressure unknowns are defined at the cell centers, and velocity unknowns at the cell faces. We note the proposed preconditioners are not limited to Cartesian grids or the MAC scheme. Any discretization of Eq.1 where the p and $\boldsymbol{u}$ unknowns are arranged in a spatially staggered configuration [8, 10] will benefit from this work.

The MAC scheme yields the following linear, saddle-point system:

$$\begin{bmatrix}\mathrm{F} & \mathrm{G}^\top\\ \mathrm{G} & \mathrm{O}\end{bmatrix}\begin{bmatrix}\hat{x}_u\\ \hat{x}_p\end{bmatrix}=\begin{bmatrix}\hat{b}_u\\ \hat{b}_p\end{bmatrix}\qquad\Rightarrow\qquad\hat{\mathrm{A}}\hat{x}=\hat{b} \tag{2}$$

Journal Pre-proof

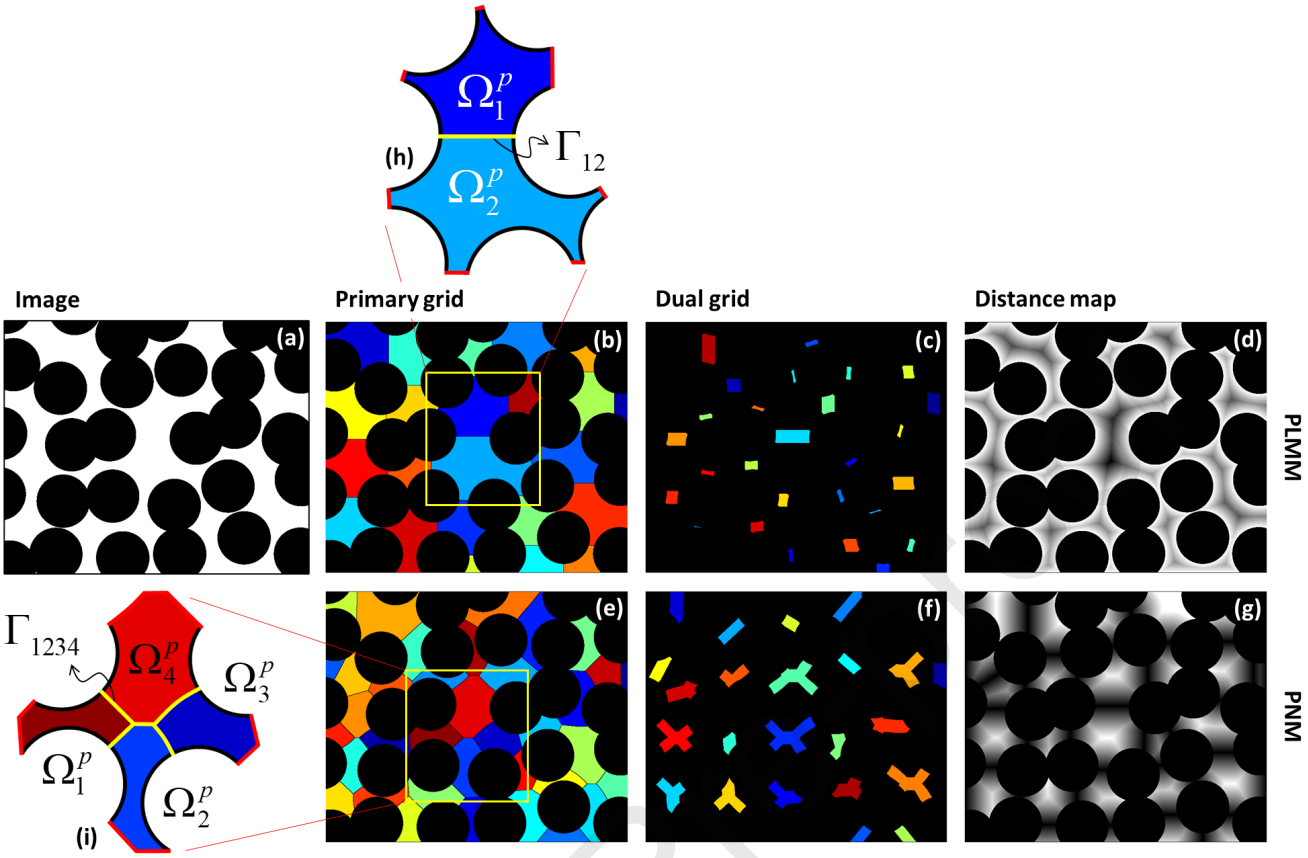


Figure 1: (a) Schematic of a binary image with white representing void and black representing solid. (b, e) Primary grids (or subdomains) obtained from the PLMM (b) and PNM (e) decompositions. (c, f) Corresponding dual grids that cover the interfaces between the primary grids. (d, g) Distance maps used in marker-based watershed segmentation to obtain the primary grids in (b) and (e). (h, i) Each basis function is associated with an interface (yellow). In PLMM, interfaces are shared between two primary grids only, but in PNM, they are shared among multiple primary grids.

where the blocks F, $G^\top$, G are the discretized forms of the Laplacian ($\mu\Delta$), gradient ($-\nabla$), and divergence ($\nabla\cdot$) operators. The unknowns $\hat{x}_u$ and $\hat{x}_p$ correspond to face velocities and cell pressures, respectively, as shown in Fig.2c. The right-hand side (RHS) vectors $\hat{b}_u$ and $\hat{b}_p$ encode the inlet/outlet BCs mentioned above. When Ω is large, so is the discretized matrix $\hat{A}$, and iterative or Krylov solvers become necessary. However, such solvers converge very slowly without preconditioning. Our goal is to develop highly efficient preconditioners for solving Eq.2. In the following, we present two such preconditioners: *monolithic* and *block* variants. We discuss these in that order.

3. Monolithic preconditioning

3.1. Overall structure

We present two monolithic preconditioners for Eq.2. One is based on an algebraic reformulation of the pore-level multiscale method (PLMM) [19], and the other based on the pore network model (PNM) used in the reduced-order simulation of single-phase, creeping flow in porous media. For short, we shall refer to them as the *monolithic PLMM* and *monolithic PNM* preconditioners. Their overall structures are identical and consists of two parts: a global preconditioner, M_G, and a local smoother, M_L. The role of M_G is to find an approximate solution in a low-dimensional coarse space, and thus attenuate low-frequency error modes. In contrast, the role of M_L is to seek approximations on the fine grid and attenuate high-frequency errors. We combine M_G and M_L into a monolithic preconditioner M via:

$$M^{-1} = M_G^{-1} + M_L^{-1}(I - \hat{A}M_G^{-1}) \tag{3a}$$

Journal Pre-proof

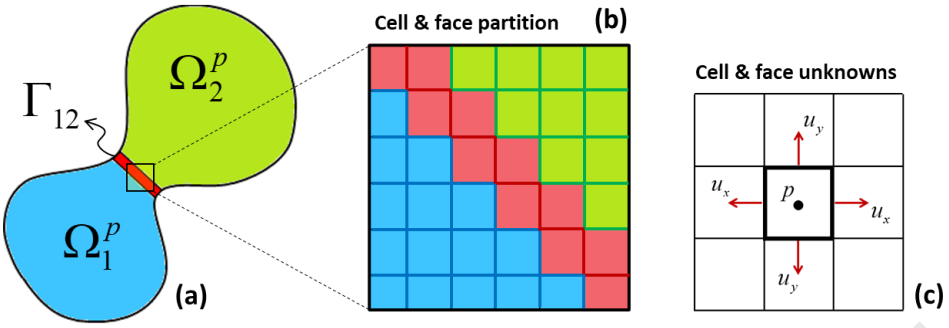


Figure 2: (a) Two primary grids, Ω_1^p and Ω_2^p, separated by a contact interface, Γ_{12}. (b) Partitioning of the cells and faces of the fine grid among the two primary grids and the contact interface. Light/dark blue cells/faces belong to Ω_1^p, light/dark green cells/faces belong to Ω_2^p, and light/dark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

$$\mathrm{M}_{\mathrm{L}}^{-1} = \sum_{i=1}^{n_{st}} \mathrm{M}_l^{-1}(\mathrm{I} - \hat{\mathrm{A}}\mathrm{M}_l^{-1})^{i-1} \tag{3b}$$

where M_{L} is expressed as an n_{st}-stage application of a *base smoother*, M_l. The base smoother can either be a black-box preconditioner, like block Gauss-Seidel, or a customized variant, like the additive-Schwarz smoothers presented later. Applying the combined M to the linear system in Eq.2 attenuates low- and high-frequency errors simultaneously.

In the following, we first formulate M_{G} and M_{L} for the monolithic PLMM. This involves discussing the specific domain decomposition used, and various permutation, reduction, and coarsening operations performed. After these details are established, the monolithic PNM preconditioner is presented, which follows an identical formulation as the monolithic PLMM, except for a few key differences regarding the decomposition and permutation operations.

3.2. Preconditioner based on the pore-level multiscale method (PLMM)

3.2.1. Domain decomposition

The first step in formulating M_{G} and M_{L} for the monolithic PLMM preconditioner is to partition Ω into a set of non-overlapping subdomains, or *primary grids*, denoted by Ω_i^p. For the Ω in Fig.1a, the primary grids correspond to the randomly colored regions in Fig.1b. We refer to the interface shared between two adjacent primary grids Ω_i^p and Ω_j^p as a *contact interface* and denote it by Γ_{ij}. This is shown by the thick yellow line in Fig.1h. The decomposition is used later to build numerical basis functions, each associated with a single contact interface and whose support spans the two neighboring primary grids that share the interface (see Fig.1h and Fig.3).

The decomposition of Ω into Ω_i^p is performed using the marker-based watershed segmentation algorithm [27, 28], as discussed in [19] for PLMM. The idea is to first compute a distance map like the one in Fig.1d, where each white pixel is assigned a real number inversely proportional to that pixel's distance from the nearest solid surface (black pixel). Hence, pixels furthest from the solid appear dark in Fig.1d (low value), and pixels closest to the solid appear bright (high value). In watershed segmentation, this distance map is treated as a topographical surface that is gradually flooded with water. Initially, isolated puddles form that eventually grow into the primary grids. The growth of a puddle is terminated as soon as it comes into contact with another puddle. The interfaces that adjacent puddles share are the contact interfaces defined above. The "marker-based" aspect of this segmentation means that the initial "seed" of each puddle is provided as input, which here are the local minima of the distance map in Fig.1d (see Appendix A). No new puddles beyond these seeds are allowed to form. For more details on marker-based watershed transform see [28].

In addition to primary grids, we also define so-called *dual grids* that are comprised of thin regions covering the interfaces between the primary grids, as shown in Fig.1c. We denote dual grids by Ω_i^d and note their union is much smaller in size than Ω. Dual grids are used later to formulate M_{L}, and they are key to eliminating high-frequency errors that tend to accumulate near contact interfaces after each application of M_{G}. To create Ω_i^d, we perform successive morphological dilations (an operation in image analysis) of the pixels that comprise each contact interface. Unlike the geometric variant of PLMM in [19], here, we allow dual grids to overlap. In [19], overlapping dual grids were merged, which can lead to large sample-spanning regions in high-porosity domains. Local problems defined on such

Journal Pre-proof

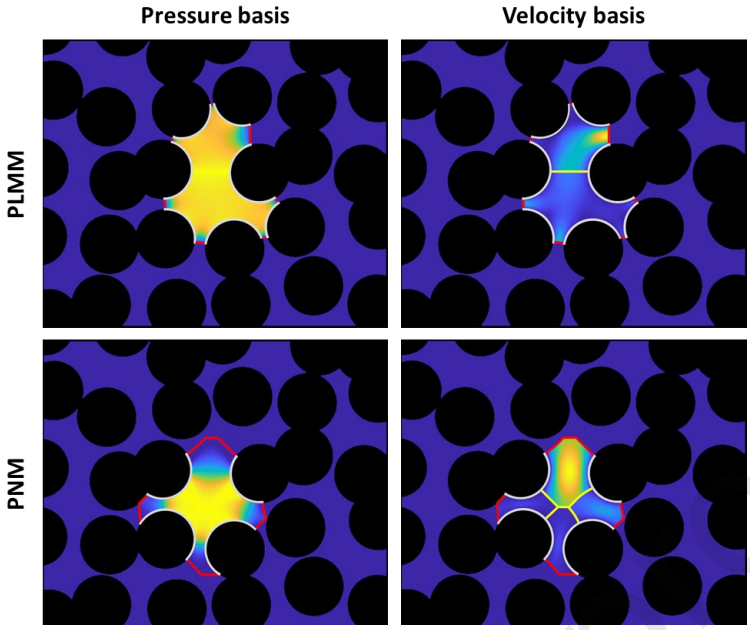


Figure 3: Schematic of a pressure basis and its corresponding velocity basis in the PLMM and PNM preconditioners. These bases correspond to the primary grids shown in Figs.1h-i. The support of each basis is delineated by the thick red/gray lines. Red lines are shared interfaces with other primary grids, where $p=0$ and $\partial \boldsymbol{u}/\partial n=0$ BCs are imposed. Yellow lines are where $p=1$ and $\partial \boldsymbol{u}/\partial n=0$ are imposed. Gray lines are the void-solid interface where no-slip BCs are imposed. The pressure and velocity bases together form one column in the B matrix given by Eq.15a.

regions are as costly to solve as the original system in Eq.2. This drawback is eliminated by the algebraic formulation herein. Finally, we shall refer to primary and dual grids as *coarse grids* if the distinction is not important.

3.2.2. *Global preconditioner*

We now formulate $\mathrm{M_G}$, which is equivalent to outlining a series of steps for solving Eq.2 approximately. We begin with a permutation operation based on the decomposition described in Section 3.2.1. Consider the simple 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, the fine-grid *cells* comprising Ω can be 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 and/or assignments of cells 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, we can construct a permutation matrix W such that when it is applied to Eq.2, yields:

$$\underbrace{\mathrm{W}^\top \hat{\mathrm{A}} \mathrm{W}}_{\mathrm{A}} \underbrace{\mathrm{W}^\top \hat{x}}_{x} = \underbrace{\mathrm{W}^\top \hat{b}}_{b} \qquad \Rightarrow \qquad \mathrm{A}x = b \tag{4a}$$

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

$$\mathrm{A} = \begin{bmatrix} \mathrm{A}_p^p & \mathrm{A}_c^p & \mathrm{A}_f^p \\ \mathrm{A}_p^c & \mathrm{A}_c^c & \mathrm{A}_f^c \\ \mathrm{A}_p^f & \mathrm{A}_c^f & \mathrm{A}_f^f \end{bmatrix} \qquad b = \begin{bmatrix} b^p \\ b^c \\ b^f \end{bmatrix} \qquad x = \begin{bmatrix} x^p \\ x^c \\ x^f \end{bmatrix} \tag{4b}$$

Note the matrix W is unitary and satisfies $\mathrm{WW}^\top = \mathrm{I}$.

The super/subscript p in each block denotes that the corresponding *cell and face* unknowns/residuals (i.e., columns/rows) belong to the primary grids. The super/subscript c denotes the corresponding *cell only* unknowns/residuals be-

Journal Pre-proof

long to the contact interfaces. And the super/subscript f denotes the corresponding *face only* unknowns/residuals belong to the contact interfaces. Recall that cell unknowns correspond to pressure, and their residuals to the continuity Eq.1b. Similarly, face unknowns correspond to velocity, and their residuals to the momentum balance Eq.1a.

The next step in formulating $\mathrm{M_G}$ is to introduce a number of *closure* or *localization assumptions* to simplify, and effectively decouple, the permuted system in Eq.4. We call this operation a *reduction*, and it imposes the following BCs on each contact interface: $p = \text{const}$ and $\partial \boldsymbol{u}/\partial n = 0$. The physical justification for the optimality of these BCs in PLMM was argued in [19]. Briefly, contact interfaces coincide with geometric bottlenecks of Ω where streamlines exhibit a converging-diverging pattern. The closure BCs follow from a local analysis of Eq.1 at such bottlenecks [19]. The reduction operation proceeds in three stages. The first is to approximate A as follows:

$$\mathrm{A} \approx \begin{bmatrix} \tilde{\mathrm{A}}_p^p & \mathrm{A}_c^p & \mathrm{O} \\ \mathrm{A}_p^c & \tilde{\mathrm{A}}_c^c & \mathrm{O} \\ \mathrm{A}_p^f & \mathrm{A}_c^f & \mathrm{A}_f^f \end{bmatrix} \qquad \begin{aligned} \tilde{\mathrm{A}}_p^p &= \mathrm{A}_p^p + \mathrm{diag}\left(\mathrm{csum}(\mathrm{A}_f^p)\right) \\ \tilde{\mathrm{A}}_c^c &= \mathrm{A}_c^c + \mathrm{diag}\left(\mathrm{csum}(\mathrm{A}_f^c)\right) \end{aligned} \tag{5}$$

where the operation $\mathrm{csum}(\cdot)$ sums all the columns of its input matrix and yields a single column vector. The operation $\mathrm{diag}(\cdot)$ takes an input vector and creates a diagonal matrix whose diagonal entries coincide with those of the vector. Later, we also use $\mathrm{diag}(\cdot)$ in a different way: If the input is a matrix (instead of a vector here), then the output of $\mathrm{diag}(\cdot)$ is the diagonal of the matrix expressed as a column vector. Eq.5 imposes $\partial \boldsymbol{u}/\partial n = 0$ at contact interfaces, and thereby decouples the x^f unknowns in Eq.4b. In other words, given Eq.5, we can now focus on solving $\tilde{\mathrm{A}}\tilde{x} = \tilde{b}$, where:

$$\tilde{\mathrm{A}} = \begin{bmatrix} \tilde{\mathrm{A}}_p^p & \mathrm{A}_c^p \\ \mathrm{A}_p^c & \tilde{\mathrm{A}}_c^c \end{bmatrix} \qquad \tilde{b} = \begin{bmatrix} b^p \\ b^c \end{bmatrix} \qquad \tilde{x} = \begin{bmatrix} x^p \\ x^c \end{bmatrix} \tag{6}$$

after which x^f can be computed rapidly as follows (notice A_f^f is a very small matrix):

$$x^f = \left(\mathrm{A}_f^f\right)^{-1}\left(b^f - \mathrm{A}_p^f x^p - \mathrm{A}_c^f x^c\right) \tag{7}$$

Let us denote the total number of primary grids by N^p, and the number of contact interfaces by N^c. By construction, N^c equals the number of dual grids N^d. The expanded form of the reduced system in Eq.6 has the block structure:

$$\tilde{\mathrm{A}}_p^p = \begin{bmatrix} \tilde{\mathrm{A}}_{p_1}^{p_1} & \cdots & \tilde{\mathrm{A}}_{p_n}^{p_1} \\ \vdots & \ddots & \vdots \\ \tilde{\mathrm{A}}_{p_1}^{p_n} & \cdots & \tilde{\mathrm{A}}_{p_n}^{p_n} \end{bmatrix} \quad \tilde{\mathrm{A}}_c^c = \begin{bmatrix} \tilde{\mathrm{A}}_{c_1}^{c_1} & \cdots & \tilde{\mathrm{A}}_{c_m}^{c_1} \\ \vdots & \ddots & \vdots \\ \tilde{\mathrm{A}}_{c_1}^{c_m} & \cdots & \tilde{\mathrm{A}}_{c_m}^{c_m} \end{bmatrix} \quad \mathrm{A}_c^p = \begin{bmatrix} \mathrm{A}_{c_1}^{p_1} & \cdots & \mathrm{A}_{c_m}^{p_1} \\ \vdots & \ddots & \vdots \\ \mathrm{A}_{c_1}^{p_n} & \cdots & \mathrm{A}_{c_m}^{p_n} \end{bmatrix} \quad \mathrm{A}_p^c = \begin{bmatrix} \mathrm{A}_{p_1}^{c_1} & \cdots & \mathrm{A}_{p_n}^{c_1} \\ \vdots & \ddots & \vdots \\ \mathrm{A}_{p_1}^{c_m} & \cdots & \mathrm{A}_{p_n}^{c_m} \end{bmatrix} \tag{8a}$$

$$b^p = \begin{bmatrix} b^{p_1} \\ \vdots \\ b^{p_n} \end{bmatrix} \quad b^c = \begin{bmatrix} b^{c_1} \\ \vdots \\ b^{c_m} \end{bmatrix} \quad x^p = \begin{bmatrix} x^{p_1} \\ \vdots \\ x^{p_n} \end{bmatrix} \quad x^c = \begin{bmatrix} x^{c_1} \\ \vdots \\ x^{c_m} \end{bmatrix} \tag{8b}$$

where $n = N^p$ and $m = N^c$. Each block corresponds to unknowns associated with either a primary grid, a contact interface, or a coupling between the two. The second stage of the reduction involves approximating $\tilde{\mathrm{A}}_p^p$ as follows:

$$\tilde{\mathrm{A}}_p^p \approx \bar{\mathrm{A}}_p^p = \begin{bmatrix} \bar{\mathrm{A}}_{p_1}^{p_1} & & \mathrm{O} \\ & \ddots & \\ \mathrm{O} & & \bar{\mathrm{A}}_{p_n}^{p_n} \end{bmatrix} \qquad \bar{\mathrm{A}}_{p_i}^{p_i} = \tilde{\mathrm{A}}_{p_i}^{p_i} + \mathrm{diag}\left(\sum_{\forall j \neq i} \mathrm{csum}(\tilde{\mathrm{A}}_{p_j}^{p_i})\right) \tag{9}$$

which decouples the unknowns of each primary grid from those of all other primary grids. Similar to Eq.5, off-diagonal blocks of each block-row are column-summed and added to the diagonal entries. This completes the enforcement of the $\partial \boldsymbol{u}/\partial n = 0$ BC at all contact interfaces. Given Eq.9, we can now solve the even simpler system $\bar{\mathrm{A}}\tilde{x} = \tilde{b}$, where:

$$\bar{\mathrm{A}} = \begin{bmatrix} \bar{\mathrm{A}}_p^p & \mathrm{A}_c^p \\ \mathrm{A}_p^c & \tilde{\mathrm{A}}_c^c \end{bmatrix} \tag{10}$$

Journal Pre-proof

The third and final stage of the reduction aims to enforce $p = \text{const}$ at all contact interfaces. This is done as follows:

$$\bar{\mathrm{A}}\tilde{x} = \tilde{b}\,, \quad \tilde{x} \simeq \mathrm{Q}x_M \quad \Rightarrow \quad \underbrace{\mathrm{Q}^\top \bar{\mathrm{A}}\mathrm{Q}}_{\mathrm{A_M}} x_M = \underbrace{\mathrm{Q}^\top \tilde{b}}_{b_M} \quad \Rightarrow \quad \mathrm{A_M} x_M = b_M \tag{11a}$$

where

$$\mathrm{Q} = \begin{bmatrix} \mathrm{I}_{N_f^p \times N_f^p} & \mathrm{O} \\ \mathrm{O} & \mathrm{Q^o} \end{bmatrix} \qquad \mathrm{Q^o} = \begin{bmatrix} \mathbf{1}^{c_1} & & \mathrm{O} \\ & \ddots & \\ \mathrm{O} & & \mathbf{1}^{c_m} \end{bmatrix}_{N_f^c \times N^c} \qquad \mathbf{1}^{c_i} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N_f^{c_i} \times 1} \tag{11b}$$

In Eq.11b, 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 c_i, respectively. Recall 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*, Q, is block-diagonal and consists of 0 and 1 entries. Left-multiplying a matrix/vector by $\mathrm{Q}^\top$ performs a row-sum of all fine-scale entries associated with each interface c_i. Because the rows belonging to c_i in $\bar{\mathrm{A}}$ correspond to the discretized continuity Eq.1b, performing a row-sum is equivalent to imposing an overall (or integrated) mass balance over the whole interface. This is a weak, as opposed to a strong or pointwise, imposition of continuity. Conversely, right-multiplying $\bar{\mathrm{A}}$ by Q performs a column-sum of all fine-scale entries associated with c_i. Because the columns belonging to c_i in $\bar{\mathrm{A}}$ correspond to the pressure unknowns, the column-sum is equivalent to imposing the closure BC $p = \text{const}$ over c_i.

With the above three-stage reduction of $\mathrm{A}x = b$ in Eq.4a complete, we can focus on solving the reduced system $\mathrm{A_M} x_M = b_M$ in Eq.11a instead. The latter has the following block structure:

$$\mathrm{A_M} = \begin{bmatrix} \bar{\mathrm{A}}_p^p & \bar{\mathrm{A}}_c^p \\ \bar{\mathrm{A}}_p^c & \bar{\mathrm{A}}_c^c \end{bmatrix} \qquad b_M = \begin{bmatrix} b^p \\ \bar{b}^c \end{bmatrix} \qquad x_M = \begin{bmatrix} x^p \\ x^o \end{bmatrix} \tag{12}$$

where x^o contains the *coarse-scale* pressure unknowns associated with the contact interfaces (one per interface). Its length is $N^c \times 1$. Our next step is to decouple x^o from the fine-scale pressure/velocity unknowns in x^p that belong to the primary grids. We proceed by introducing a *prolongation matrix*, P, and a *restriction matrix*, R, whereby the following coarse-scale problem can be formulated and solved for x^o, and for the auxiliary coarse-scale unknown y^o:

$$\mathrm{A_M} x_M = b_M, \quad x_M = \mathrm{P}\begin{bmatrix} x^o \\ y^o \end{bmatrix} \quad \Rightarrow \quad \underbrace{\mathrm{R}\mathrm{A_M}\mathrm{P}}_{\mathrm{A^o}} \begin{bmatrix} x^o \\ y^o \end{bmatrix} = \underbrace{\mathrm{R}b_M}_{b^o} \tag{13}$$

The length of y^o is $N^p \times 1$, one entry per primary grid. Before presenting P and R, we need the definition below:

Definition 1. Let $\mathrm{E}_p^{p_i}$ and $\mathrm{R}_p^{p_i}$ denote the *extension* and *contraction* matrices of the primary grid p_i, and $e_c^{c_i}$ and $r_c^{c_i}$ the extension and contraction vectors of the contact interface c_i, respectively. They are defined as follows:

$$\mathrm{E}_p^{p_i} = \left[\Delta_{p_1}^{p_i}, \Delta_{p_2}^{p_i}, \cdots, \Delta_{p_n}^{p_i}\right]^\top_{N_f^p \times N_f^{p_i}} \qquad \mathrm{R}_p^{p_i} = (\mathrm{E}_p^{p_i})^\top \tag{14a}$$

$$e_c^{c_i} = \left[\delta_{1i}, \delta_{2i}, \cdots, \delta_{mi}\right]^\top_{N^c \times 1} \qquad r_c^{c_i} = (e_c^{c_i})^\top \tag{14b}$$

where

$$\Delta_{p_j}^{p_i} = \begin{cases} \mathrm{I}_{N_f^{p_i} \times N_f^{p_i}} & \text{if } i = j \\ \mathrm{O}_{N_f^{p_i} \times N_f^{p_j}} & \text{if } i \neq j \end{cases} \qquad \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{14c}$$

Similar to $N_f^{c_i}$ defined earlier, $N_f^{p_i}$ is the number of fine-scale unknowns belonging to the primary grid p_i. Multiplying a $N_f^{p_i} \times 1$ vector by $\mathrm{E}_p^{p_i}$ yields a $N_f^p \times 1$ vector defined over all primary grids (excluding the contact interfaces) with the same entries inside Ω_i^p but zero outside. Similarly, multiplying a scalar defined on the interface c_i by $e_c^{c_i}$ extends it to a $N^c \times 1$ vector defined over all contact interfaces. Multiplication of a vector by $\mathrm{R}_p^{p_i}$ or $r_c^{c_i}$ maps it in the opposite direction.

Journal Pre-proof

The prolongation matrix, P, in Eq.13 can now be defined as follows:

$$\mathrm{P} = \begin{bmatrix} \mathrm{B} & \mathrm{C} \\ \mathrm{I} & \mathrm{O} \end{bmatrix} \qquad \mathrm{B} = \begin{bmatrix} p^{p_1}_{c_1} & p^{p_1}_{c_2} & \cdots & p^{p_1}_{c_m} \\ p^{p_2}_{c_1} & p^{p_2}_{c_2} & \cdots & p^{p_2}_{c_m} \\ \vdots & \vdots & \ddots & \vdots \\ p^{p_n}_{c_1} & p^{p_n}_{c_2} & \cdots & p^{p_n}_{c_m} \end{bmatrix}_{N^p_f \times N^c} \qquad \mathrm{C} = \begin{bmatrix} c^{p_1} & & & \mathrm{O} \\ & c^{p_2} & & \\ & & \ddots & \\ \mathrm{O} & & & c^{p_n} \end{bmatrix}_{N^p_f \times N^p} \tag{15a}$$

where $p^{p_i}_{c_j}$ and c^{p_i} are referred to as *shape* and *correction* vectors, respectively, defined as follows:

$$p^{p_i}_{c_j} = \begin{cases} -\left(\bar{\mathrm{A}}^{p_i}_{p_i}\right)^{-1} R^{p_i}_p \bar{\mathrm{A}}^p_c e^{c_j}_c \,, & c_j \in C^{p_i} \\ \mathrm{O}\,, & c_j \notin C^{p_i} \end{cases} \qquad c^{p_i} = \left(\bar{\mathrm{A}}^{p_i}_{p_i}\right)^{-1} b^{p_i} \tag{15b}$$

In Eq.15b, C^{p_i} is the index set of all contact interfaces intersecting the boundary of the primary grid p_i. Eq.15b implies that only two entries in each column of B are non-zero. Given $p^{p_i}_{c_j}$ is defined over the primary grid p_i and associated with the interface c_j, the two non-zero entries are the only two shape vectors sharing c_j. We call each column of B a *basis vector*, an example of which is shown in Fig.3. Each basis vector consists of two shape vectors, and is comprised of a velocity field and a pressure field defined over two adjacent primary grids. By contrast, c^{p_i} is defined on only p_i.

To formulate the coarse system in Eq.13, we also need a restriction matrix, R. We propose two options:

$$\textit{FEM restriction:} \qquad \mathrm{R} = \mathrm{P}^\top \tag{16a}$$

$$\textit{FVM restriction:} \qquad \mathrm{R} = \begin{bmatrix} \mathrm{O} & \mathrm{I} \\ \Pi(\mathrm{C}^\top) & \mathrm{O} \end{bmatrix} \tag{16b}$$

where I is an $N^c \times N^c$ identity matrix, and $\Pi(\mathrm{C}^\top)$ is a matrix with 0 and 1 entries that reflects the sparsity pattern of $\mathrm{C}^\top$ in Eq.15a (i.e., its entries are 1 where $\mathrm{C}^\top$ is non-zero, and 0 elsewhere). Left-multiplying a vector by the FEM restriction performs a Galerkin projection onto the subspace spanned by the columns of P. In contrast, left-multiplying by the FVM restriction performs a row-sum over each primary grid, p_i, while leaving the rows corresponding to each contact interface, c_i, unchanged. The latter is because a similar row-sum over each interface was already performed during the reduction step that led to the system in Eq.11a, upon which R acts in Eq.13. Together, the two row-sums enforce mass conservation on each interface and each primary grid in a weak (or integrated) sense. For brevity, we denote the two restrictions matrices by $\mathrm{R}_{\mathrm{FEM}}$ and $\mathrm{R}_{\mathrm{FVM}}$ hereafter. We note $\mathrm{R}_{\mathrm{FEM}}$ is popular in preconditioning solid mechanics problems [25, 29], where $\hat{\mathrm{A}}$ in Eq.2 is symmetric and positive definite. But in Stokes flow, $\hat{\mathrm{A}}$ has a saddle-point structure, for which $\mathrm{R}_{\mathrm{FVM}}$ is more appropriate and exhibits superior performance as shown later in Section 6.

Given P and R from Eqs.15-16, we can now formulate and solve Eq.13 for the coarse-scale unknowns x^o and y^o. To obtain the approximate solution $\hat{x}$, which was our ultimate goal from the start, we must simply undo all of the transformations introduced above. This is done by executing the following steps in order: (1) Compute $x_M = \mathrm{P}\,[x^o, y^o]^\top$ from Eq.13; (2) Compute $\tilde{x} = \mathrm{Q} x_M$ from Eq.11a; (3) Use $\tilde{x} = [x^p, x^c]^\top$ from Eq.6 to obtain x^f from Eq.7; (4) Assemble $x = [x^p, x^c, x^f]^\top$ from Eq.4b and undo the permutation via $\hat{x} = \mathrm{W} x$ in Eq.4a. This completes the formulation of M_{G}.

In a Krylov solver, preconditioning means solving systems like $\hat{\mathrm{A}}\hat{w} = \hat{v}$ for $\hat{w}$ approximately for a given RHS vector $\hat{v}$. The above formulation of M_{G} defines a set of steps (or recipe) for computing this approximate solution rapidly and with a parallel machine. The latter is because the most computationally expensive step is in computing the shape and correction vectors in Eq.15b, which is decoupled and parallelizable across all primary grids. While M_{G} is in principle a matrix, its closed-form expression is difficult to write down given some of the special operators introduced above, e.g., csum(·). We consider the "recipe" way of presenting M_{G} more clear and less prone to confusion. We remark the formulation of M_{G} for PNM is almost identical to PLMM, save for a few modifications discussed in Section 3.3.

3.2.3. Local smoother

Here, we present a compatible smoother, M_{L}, for the global preconditioner, M_{G}, in the previous section. We follow the idea in [26] for linear-elastic deformation. Namely, we build M_{L} from the combination of two additive-Schwarz (or block-Jacobi) [30] smoothers: one that acts on primary grids denoted by M_{p}, and another that acts on dual grids denoted by M_{d}. The former attenuates high-frequency errors on the primary grids, and the latter on the dual grids. We

Journal Pre-proof

note that most of the errors that remain after applying $\mathrm{M_G}$ tend to concentrate at contact interfaces, which are covered by the dual grids (Fig.1c). Hence, $\mathrm{M_d}$ specifically targets these errors. To formulate $\mathrm{M_p}$ and $\mathrm{M_d}$, we need a definition:

Definition 2. Let $\mathrm{E}_f^{d_i}$ and $\mathrm{R}_f^{d_i}$ denote the *extension* and *contraction* matrices of the dual grid d_i, defined as follows:

$$\mathrm{E}_f^{d_i} = \left[u_1^{d_i}, u_2^{d_i}, \cdots, u_\nu^{d_i}\right]_{N_f \times N_f^{d_i}} \qquad \mathrm{R}_f^{d_i} = \left[\eta_1^{d_i} u_1^{d_i},\, \eta_2^{d_i} u_2^{d_i},\, \cdots,\, \eta_\nu^{d_i} u_\nu^{d_i}\right]^\top \tag{17a}$$

where

$$u_j^{d_i} = [m_j^{d_i}(1)\,,\, m_j^{d_i}(2)\,,\, \cdots, m_j^{d_i}(N^f)]^\top \tag{17b}$$

We have denoted the total number of fine-scale unknowns by N_f (i.e., length of $\hat{x}$ in Eq.2), and the number of fine-scale unknowns in the dual grid d_i by $N_f^{d_i}$. In Eq.17a, we have used $\nu = N_f^{d_i}$ for brevity. In Eq.17b, $m_j^{d_i}(k)$ is a function that returns 1 if the global index k corresponds to the same unknown as the local index j contained within the dual grid d_i. Otherwise, it returns 0. The length of $u_j^{d_i}$ is $N_f \times 1$, and it contains only one entry that is equal to 1, with the rest equal to 0. Multiplying a $N_f^{d_i} \times 1$ vector defined on d_i by $\mathrm{E}_f^{d_i}$ extends it to a $N_f \times 1$ vector defined over the entire domain. Multiplication by $\mathrm{R}_f^{d_i}$ maps in the opposite direction, restricting a globally defined vector to one defined on d_i. The scalars $\eta_j^{d_i}$ serve as weights that ensure the following partition of unity: $\sum_{d_i} \mathrm{E}_f^{d_i} \mathrm{R}_f^{d_i} = \mathrm{I}$.[1] These weights essentially average the contributions from multiple overlapping dual grids. Wherever no overlap occurs, the weight is 1.

We can similarly define *extension* and *contraction* matrices for the primary grid p_i as follows:

$$\mathrm{E}_f^{p_i} = \left[u_1^{p_i}, u_2^{p_i}, \cdots, u_\mu^{p_i}\right]_{N_f \times N_f^{p_i}} \qquad \mathrm{R}_f^{p_i} = \left(\mathrm{E}_f^{p_i}\right)^\top \tag{18a}$$

where

$$u_j^{p_i} = [m_j^{p_i}(1)\,,\, m_j^{p_i}(2)\,,\, \cdots, m_j^{p_i}(N^f)]^\top \tag{18b}$$

and $\mu = N_f^{p_i}$. Notice $\mathrm{E}_f^{p_i}$ and $\mathrm{R}_f^{p_i}$ are different from the extension/contraction matrices in Definition 1, which act on the permuted and reduced system $\mathrm{A_M} x_M = b_M$ in Eq.13. In contrast, $\mathrm{E}_f^{p_i}$ and $\mathrm{R}_f^{p_i}$ act on the original system $\hat{\mathrm{A}}\hat{x} = \hat{b}$ in Eq.2. Given that primary grids do not overlap, no weights are needed in defining the contraction matrix in Eq.18a.

We can now formulate $\mathrm{M_p}$ and $\mathrm{M_d}$, through their inverses, as follows:

$$\mathrm{M_p^{-1}} = \sum_{i=1}^{N^p} \mathrm{E}_f^{p_i} \underbrace{(\mathrm{R}_f^{p_i} \hat{\mathrm{A}}\, \mathrm{E}_f^{p_i})^{-1}}_{\hat{\mathrm{A}}_{p_i}} \mathrm{R}_f^{p_i} \qquad \mathrm{M_d^{-1}} = \sum_{i=1}^{N^c} \mathrm{E}_f^{d_i} \underbrace{(\mathrm{R}_f^{d_i} \hat{\mathrm{A}}\, \mathrm{E}_f^{d_i})^{-1}}_{\hat{\mathrm{A}}_{d_i}} \mathrm{R}_f^{d_i} \tag{19}$$

which we combine into a single smoother in the following multiplicative fashion:

$$\mathrm{M_{dp}^{-1}} = \mathrm{M_d^{-1}} + \mathrm{M_p^{-1}} \left(\mathrm{I} - \hat{\mathrm{A}}\mathrm{M_d^{-1}}\right) \tag{20}$$

Applying $\mathrm{M_p^{-1}}$ to an $N_f \times 1$ vector involves solving N^p decoupled local problems on the primary grids, whose coefficient matrices are $\hat{\mathrm{A}}_{p_i}$. Similarly, applying $\mathrm{M_d^{-1}}$ involves solving N^c decoupled problems on the dual grids, whose coefficient matrices are $\hat{\mathrm{A}}_{d_i}$. Both are amenable to parallelism. We note that in Eq.20, we first apply $\mathrm{M_d^{-1}}$, then $\mathrm{M_p^{-1}}$. This order was found to perform better for the monolithic (but not block, as discussed later) preconditioners proposed. To arrive at the final expression for the smoother $\mathrm{M_L}$, we substitute $\mathrm{M}_l = \mathrm{M_{dp}}$ into Eq.3b as the base smoother.

3.3. Preconditioner based on the pore network model (PNM)

The formulation of the monolithic PNM preconditioner is almost identical to that of the monolithic PLMM, except for the very important distinction of what we mean by "primary grids" and "dual grids." In other words, the domain decomposition is very different, which has implications on some of the details outlined in Sections 3.2.2-3.2.3 for $\mathrm{M_G}$ and $\mathrm{M_L}$. These are discussed below. The PNM preconditioner is the first repurposing of PNM to accelerate DNS.

[1] In [26], this criterion was stated incorrectly as $\mathrm{R}_f^{\zeta_i} \mathrm{E}_f^{\zeta_i} = \mathrm{I}$, where the superscript ζ_i was used instead of d_i herein.

Journal Pre-proof

3.3.1. Domain decomposition

In PNM, the domain Ω is decomposed into primary grids Ω_i^p as shown in Fig.1e. Notice the Ω_i^p correspond to the local constrictions of Ω, or *throats* as they are called in the PNM literature. This is very different from PLMM, where primary grids correspond to the local enlargements of Ω, or *pores*. Moreover, contact interfaces here may be shared by more than two primary grids, unlike PLMM. Fig.1i shows one such interface (yellow line) denoted by Γ_{1234}, which is shared by the primary grids Ω_1^p, Ω_2^p, Ω_3^p, and Ω_4^p. Notice Γ_{1234} is located inside a pore, whereas in PLMM, a contact interface is always located inside a throat. The above constitute the main differences between PNM and PLMM, and below we detail how the Ω_i^p are obtained in PNM from decomposing Ω. Note that the dual grids, Ω_i^d, here serve the same purpose as in PLMM: cover the contact interfaces as shown in Fig.1f. Constructing the dual grids follows the exact same procedure as outlined in Section 3.2.1, i.e., by dilating the pixels belonging to each interface (e.g., Γ_{1234}).

To decompose Ω into non-overlapping Ω_i^p, we first compute a distance map like the one shown in Fig.1g. However this time, the distance map is computed with respect to the contact interfaces identified from PLMM. Concretely, two steps are executed: (1) Perform the domain decomposition described in Section 3.2.1 for PLMM and identify the pixels associated with the contact interfaces (yellow line in Fig.1h); (2) Assign a value to each pixel (white in Fig.1a) that is proportional to its distance from the nearest contact-interface pixel identified in Step 1. This yields the distance map in Fig.1g. We then input this map into the same marker-based watershed segmentation algorithm described in Section 3.2.1. The "seeds" (or markers) passed as inputs to this segmentation are the interface pixels from Step 1 (see Appendix A). The result is the partitioning of Ω into Ω_i^p as depicted by the randomly colored regions in Fig.1e.

3.3.2. Global preconditioner

Given the domain decomposition of Section 3.3.1, we can now proceed to formulate $\mathrm{M_G}$ in *exactly* the same way as we did for PLMM in Section 3.2.2. Namely, we label the cell/face unknowns to built the permutation matrix W in Eq.4, followed by the reduction steps performed in Eqs.5-11, and lastly formulating and solving the coarse problem in Eq.13. Even the expressions for the prolongation matrix, P, and restriction matrix, R, remain the same. The *only* difference to be noted here is with regards to the number of non-zero shape vectors, $p_{c_j}^{p_i}$, in each column of the matrix B in Eq.15a. Recall we referred to these columns as *basis* vectors. In PLMM, only two entries per column are non-zero because each interface c_j is shared by only two primary grids. This is seen from the contact interface in Fig.1h (yellow), whose corresponding basis vector is plotted in Fig.3 (top row). In PNM, more than two entries per column of B can be non-zero, because each interface c_j may be shared by more than two primary grids. For example, the contact interface in Fig.1i (yellow) is shared by 4 primary grids, and corresponds to the basis vector in Fig.3 (bottom row); consisting of a pressure and a velocity field. The expressions in Eq.15a for calculating B and $p_{c_j}^{p_i}$ remain unaltered.

3.3.3. Local smoother

The construction of the smoother $\mathrm{M_L}$ for the monolithic PNM preconditioner follows the *exact* same steps as those in Section 3.2.3 for PLMM. The only distinction is that the primary and dual grids correspond to the subdomains obtained from the domain decomposition of Section 3.3.1. All equations in Section 3.2.3 remain unaltered.

4. Block preconditioning

Block preconditioners are one of the most popular for saddle-point systems like Eq.2 arising from discretizing the Stokes Eq.1. The general structure, in block upper-triangular form $\mathrm{M_B}$, and applied as a right-preconditioner is:

$$\mathrm{M_B} = \begin{bmatrix} \mathrm{F} & \mathrm{G}^\top \\ \mathrm{O} & \mathrm{X} \end{bmatrix} \quad \Rightarrow \quad \mathrm{M_B^{-1}} = \begin{bmatrix} \mathrm{F}^{-1} & -\mathrm{F}^{-1}\mathrm{G}^\top\mathrm{X}^{-1} \\ \mathrm{O} & \mathrm{X}^{-1} \end{bmatrix} \quad \Rightarrow \quad \hat{\mathrm{A}}\mathrm{M_B^{-1}}\hat{x} = \hat{b} \tag{21}$$

We note that similar block lower-triangular or block diagonal preconditioners can be formulated. The preconditioned matrix $\hat{\mathrm{A}}\mathrm{M_B^{-1}}$ will have an ideal condition number of 1 if the matrix X is equal to the Schur complement $\mathrm{S} = -\mathrm{G}\mathrm{F}^{-1}\mathrm{G}^\top$. However, forming S explicitly is *not* feasible because it requires inverting F. Hence, the crux of all block preconditioners is to approximate S or S^{-1} and substitute it for X in Eq.21. Multiple approximations have been proposed [12], among which $\mathrm{X}_1 = (h^d/\mu)\,\mathrm{I}$ and $\mathrm{X}_2 = -\mathrm{G}\,\mathrm{D_F^{-1}}\,\mathrm{G}^\top$ are the simplest; where h is the grid size, μ is viscosity, d the problem dimension, I the identity matrix, and $\mathrm{D_F} = \mathrm{diag}(\mathrm{diag}(\mathrm{F}))$ a diagonal matrix with the diagonal entries of F. Preliminary

Journal Pre-proof

tests found that neither performs well on the complex geometries considered in Section 5. We thus opted for the more sophisticated "BFBt" approximation [14] below, which performed better and whose construction is fully algebraic:

$$\mathrm{X}^{-1} = -(\mathrm{G}\mathrm{G}^{\top})^{-1}\mathrm{G}\mathrm{F}\mathrm{G}^{\top}(\mathrm{G}\mathrm{G}^{\top})^{-1} \tag{22}$$

To improve on Eq.22 further, we use the *scaled*-BFBt variant proposed by [15] in all our simulations:

$$\mathrm{X}^{-1} = -(\mathrm{G}\mathrm{D}_{\mathrm{F}}^{-1}\mathrm{G}^{\top})^{-1}\mathrm{G}\mathrm{D}_{\mathrm{F}}^{-1}\mathrm{F}\mathrm{D}_{\mathrm{F}}^{-1}\mathrm{G}^{\top}(\mathrm{G}\mathrm{D}_{\mathrm{F}}^{-1}\mathrm{G}^{\top})^{-1} \tag{23}$$

Like Eq.22, Eq.23 involves only algebraic operations that require no knowledge of the problem parameters.

We can now summarize the main steps involved in applying the block preconditioner M_{B} to a system like $\hat{w} = \mathrm{M}_{\mathrm{B}}^{-1}\hat{v}$ inside an iterative solver. Notice the solution vector $\hat{w} = [\hat{w}_u, \hat{w}_p]^{\top}$ and RHS vector $\hat{v} = [\hat{v}_u, \hat{v}_p]^{\top}$ consist of pressure and velocity components as specified by the subscripts p and u, respectively. These steps are:

Step 1. Solve $\hat{w}_p = \mathrm{X}^{-1}\hat{v}_p$ using Eq.23
Step 2. Compute $\hat{v}_u^* = \hat{v}_u - \mathrm{G}^{\top}\hat{w}_p$
Step 3. Solve $\hat{w}_u = \mathrm{F}^{-1}\hat{v}_u^*$

Step 1 itself consists of three steps:

Step 1a. Solve $\hat{w}_p^{(1)} = \mathrm{Y}^{-1}\hat{v}_p$ where $\mathrm{Y} = \mathrm{G}\mathrm{D}_{\mathrm{F}}^{-1}\mathrm{G}^{\top}$
Step 1b. Compute $\hat{w}_p^{(2)} = \mathrm{Z}\hat{w}_p^{(1)}$ where $\mathrm{Z} = \mathrm{G}\mathrm{D}_{\mathrm{F}}^{-1}\mathrm{F}\mathrm{D}_{\mathrm{F}}^{-1}\mathrm{G}^{\top}$
Step 1c. Solve $\hat{w}_p = -\mathrm{Y}^{-1}\hat{w}_p^{(2)}$

The most costly steps are 1a, 1c, and 3, since they require solving linear systems with respect to the coefficient matrices Y and F. These correspond to the discretized weighted-pressure Laplacian and velocity Laplacian, respectively. To ensure efficiency, it is crucial to develop preconditioners for Y and F and solve their corresponding systems *approximately*. The most attractive existing option, which we adopt here as a benchmark, is AMG. Following [18], we apply a single multigrid V-cycle accompanied by one pre- and one post-smoothing operation with Gauss Seidel per level. We found this to perform well with little to no improvement from increasing the number of cycles.

Our goal now is to formulate block PLMM and block PNM preconditioners for F and Y that surpass the AMG benchmark discussed above in performance. Crucially, we get these *for free* (i.e., without any additional computations) from the permutation, reduction, and prolongation matrices calculated in Sections 3.2.2 and 3.3.2 for the monolithic PLMM and PNM preconditioners. Specifically, we proceed by defining the following global prolongation matrix:

$$\hat{\mathrm{P}} = \mathrm{W}\begin{bmatrix}\mathrm{Q}\mathrm{P}\\ \mathrm{O}\end{bmatrix} = \begin{bmatrix}\hat{\mathrm{P}}_{\mathrm{u}}\\ \hat{\mathrm{P}}_{\mathrm{p}}\end{bmatrix} \tag{24}$$

using W from Eq.4a, Q from Eq.11b, and P from Eq.15. The block $\hat{\mathrm{P}}_{\mathrm{u}}$ can be used as a prolongation matrix for F, and the block $\hat{\mathrm{P}}_{\mathrm{p}}$ as a prolongation matrix for Y. The columns of $\hat{\mathrm{P}}_{\mathrm{u}}$ and $\hat{\mathrm{P}}_{\mathrm{p}}$ are the velocity and pressure components, respectively, of the same basis vectors as in the monolithic preconditioners shown in Fig.3. We use Eq.24 to formulate the following global preconditioners for F and Y, which allow attenuating their low-frequency errors:

$$\mathrm{M}_{\mathrm{G,F}}^{-1} = \hat{\mathrm{P}}_{\mathrm{u}}\left(\hat{\mathrm{P}}_{\mathrm{u}}^{\top}\,\mathrm{F}\,\hat{\mathrm{P}}_{\mathrm{u}}\right)^{-1}\hat{\mathrm{P}}_{\mathrm{u}}^{\top} \qquad \mathrm{M}_{\mathrm{G,Y}}^{-1} = \hat{\mathrm{P}}_{\mathrm{p}}\left(\hat{\mathrm{P}}_{\mathrm{p}}^{\top}\,\mathrm{Y}\,\hat{\mathrm{P}}_{\mathrm{p}}\right)^{-1}\hat{\mathrm{P}}_{\mathrm{p}}^{\top} \tag{25}$$

Smoothers that are compatible with Eq.25 can be formulated in a similar fashion to Section 3.2.3. Namely:

$$\mathrm{M}_{\mathrm{L,F}}^{-1} = \mathrm{M}_{\mathrm{p,F}}^{-1} + \mathrm{M}_{\mathrm{d,F}}^{-1}\left(\mathrm{I} - \hat{\mathrm{A}}\mathrm{M}_{\mathrm{p,F}}^{-1}\right) \qquad \mathrm{M}_{\mathrm{L,Y}}^{-1} = \mathrm{M}_{\mathrm{p,Y}}^{-1} + \mathrm{M}_{\mathrm{d,Y}}^{-1}\left(\mathrm{I} - \hat{\mathrm{A}}\mathrm{M}_{\mathrm{p,Y}}^{-1}\right) \tag{26a}$$

where

$$\mathrm{M}_{\mathrm{p,F}}^{-1} = \sum_{i=1}^{N^p}\mathrm{E}_{f,u}^{p_i}\,\underbrace{(\mathrm{R}_{f,u}^{p_i}\,\mathrm{F}\,\mathrm{E}_{f,u}^{p_i})^{-1}}_{\mathrm{F}_{p_i}}\,\mathrm{R}_{f,u}^{p_i} \qquad \mathrm{M}_{\mathrm{d,F}}^{-1} = \sum_{i=1}^{N^c}\mathrm{E}_{f,u}^{d_i}\,\underbrace{(\mathrm{R}_{f,u}^{d_i}\,\mathrm{F}\,\mathrm{E}_{f,u}^{d_i})^{-1}}_{\mathrm{F}_{d_i}}\,\mathrm{R}_{f,u}^{d_i} \tag{26b}$$

Journal Pre-proof

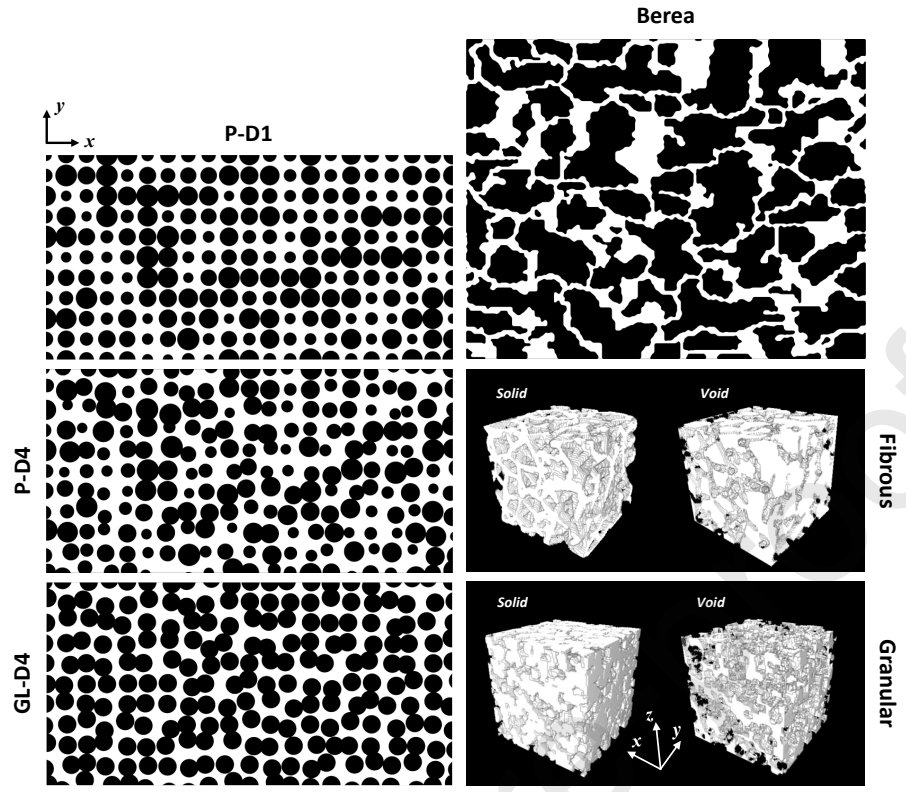


Figure 4: Schematic of the 2D and 3D porous geometries used to test the proposed preconditioners. In 2D, the void space is depicted in white and the solid phase in black. In 3D, the void space and solid phase are shown separately. We solve the Stokes Eq.1 over the void space in each case.

$$\mathrm{M}_{\mathrm{p,Y}}^{-1} = \sum_{i=1}^{N^p} \mathrm{E}_{f,p}^{p_i} \underbrace{(\mathrm{R}_{f,p}^{p_i} \mathrm{Y}\, \mathrm{E}_{f,p}^{p_i})^{-1}}_{\mathrm{Y}_{p_i}} \mathrm{R}_{f,p}^{p_i} \qquad \mathrm{M}_{\mathrm{d,Y}}^{-1} = \sum_{i=1}^{N^c} \mathrm{E}_{f,p}^{d_i} \underbrace{(\mathrm{R}_{f,p}^{d_i} \mathrm{Y}\, \mathrm{E}_{f,p}^{d_i})^{-1}}_{\mathrm{Y}_{d_i}} \mathrm{R}_{f,p}^{d_i} \tag{26c}$$

Notice $\mathrm{M}_{\mathrm{L,F}}^{-1}$ is the smoother for F, and $\mathrm{M}_{\mathrm{L,Y}}^{-1}$ is the smoother for Y. Similar to Eq.20, each is made up of two additive-Schwarz preconditioners: one acting on the primary grids ($\mathrm{M}_{\mathrm{p,F}}^{-1}$ and $\mathrm{M}_{\mathrm{p,Y}}^{-1}$) and another acting on the dual grids ($\mathrm{M}_{\mathrm{d,F}}^{-1}$ and $\mathrm{M}_{\mathrm{d,Y}}^{-1}$). Eqs.26b-26c parallel Eq.19, except the extension/contraction matrices $\mathrm{E}_{f,u}^{p_i}$, $\mathrm{R}_{f,u}^{p_i}$, $\mathrm{E}_{f,u}^{d_i}$, and $\mathrm{R}_{f,u}^{d_i}$ are understood to act only on the velocity unknowns of Eq.2. Similarly, the extension/contraction matrices $\mathrm{E}_{f,p}^{p_i}$, $\mathrm{R}_{f,p}^{p_i}$, $\mathrm{E}_{f,p}^{d_i}$, and $\mathrm{R}_{f,p}^{d_i}$ act upon the pressure unknowns. We remark that the order of applying the primary- versus dual-grid additive-Schwarz preconditioners in Eq.26a is opposite to that of Eq.20 for the monolithic preconditioners (i.e., here primary is applied first, then dual). This was found to work best for the problems in Section 5. Finally, the global and local preconditioners in Eqs.25-26 are combined in multiplicative fashion via Eq.3a. The resulting preconditioners are called *block PLMM* and *block PNM* depending on the method used to obtain the above matrices (e.g., W, Q, P).

5. Validation set

To test the performance of the proposed monolithic and block PLMM and PNM preconditioners, we consider the porous geometries depicted in Fig.4. They consist of: (1) a polydisperse but spatially ordered disk pack (P-D1); (2) a polydisperse and spatially disordered disk pack (P-D4); (3) a monodisperse and spatially disordered disk pack (GL-D4); (4) a 2D representation of a Berea sandstone taken from [31] (Berea); (5) a 3D fibrous foam generated stochastically using the porous microstructure generator (PMG) software [32] (Fibrous);[2] and (6) a 3D dense packing

[2]Software settings were: bubble fraction = 0.65, max. particle radius = 0.1, fiber radius = 0.02, reduction factor = 0.25, smoothing factor = 3.

Journal Pre-proof

of non-spherical grains generated using PMG (Granular).[3] The P-D1, P-D4, GL-D4, and Berea domains were originally analyzed by [19] using the geometric formulation of PLMM. We have chosen them to allow comparison against the algebraic formulation herein. Note the disk positions in P-D4 and GL-D4 are identical and only the disk sizes differ. Also, the disk sizes in P-D1 and P-D4 are identical, and only the disk positions differ. For the 2D domains in Fig.4, the void space is depicted in white and the solid phase in black. For the 3D domains, the void space and solid phase are shown separately. We solve the Stokes Eq.1 over the void space in each case. In the 2D domains, $p = 1$ and $\partial_n \boldsymbol{u} = 0$ are imposed on the left boundary and $p = 0$ and $\partial_n \boldsymbol{u} = 0$ on the right boundary. All lateral boundaries are sealed (no-slip). Similarly in the 3D domains, a unit pressure drop is imposed across two opposing boundaries while all lateral boundaries are sealed. Table 1 summarizes the domain sizes, number of fine-scale unknowns N_f in Eq.2, the number of cell pressures n_c, and the number of face velocities n_f, for all the domains. Note $N_f = n_c + n_f$.

Each domain is decomposed into primary grids and dual grids using the PLMM and PNM algorithms described in Sections 3.2.1 and 3.3.1. These are depicted by the randomly colored regions in Figs.5-6. Table 1 includes the number of primary grids, N^p, and dual grids, N^d, obtained from each decomposition. Recall N^d equals the number of contact interfaces N^c by construction. In Table 1, notice the N^p obtained from PLMM are roughly equal to the N^c from PNM, and the N^c from PLMM are roughly equal to the N^p from PNM. These are also consequences of the decomposition algorithms in Sections 3.2.1 and 3.3.1, which guarantee that *pores* coincide with primary grids in PLMM but dual grids (or contact interfaces) in PNM, and *throats* coincide with dual grids in PLMM but primary grids in PNM.

Table 1: Summary of domain size, number of fine-scale unknowns (N_f), number of cell pressures (n_c), number of face velocities (n_f) for the domains in Fig.4. The number of primary (N^p) and dual grids (N^c) obtained from the PLMM and PNM decompositions in Figs.5-6 are included.

	Domain size	N_f	n_c	n_f	N^p, PLMM	N^c, PLMM	N^p, PNM	N^c, PNM
P-D1	1×2	6,682,606	2,211,787	4,470,819	202	351	370	201
P-D4	1×2	6,810,716	2,255,498	4,555,218	205	323	343	202
GL-D4	1×2	5,493,617	1,815,809	3,677,808	211	283	308	203
Berea	1.42×1.77	6,642,951	2,198,652	4,444,299	241	281	344	230
Fibrous	$1 \times 1 \times 1$	2,370,727	554,712	1,816,015	153	1150	891	152
Granular	$1 \times 1 \times 1$	1,712,721	385,763	1,326,958	156	885	871	150

6. Results

Below, we probe the accuracy and performance of the monolithic and block preconditioners for PLMM and PNM. In Section 6.1, we use the global preconditioners, $\mathrm{M_G}$, associated with the monolithic PLMM and PNM preconditioners proposed in Sections 3.2.2 and 3.3.2 as *approximate solvers* and compare their accuracy. In Section 6.2, we pair $\mathrm{M_G}$ with the smoothers, $\mathrm{M_L}$, proposed in Sections 3.2.3 and 3.3.3 and compare their performance in accelerating the convergence of Krylov solvers. Section 6.3 compares the monolithic PLMM/PNM preconditioners to the block PLMM/PNM preconditioners proposed in Section 4. Section 6.4 benchmarks the latter against block AMG.

6.1. Monolithic PLMM versus monolithic PNM as approximate solvers

Figs.7 and 8 show the pressure and velocity-magnitude fields, respectively, obtained from a single application of the global preconditioner, $\mathrm{M_G}$, associated with the monolithic PLMM and PNM preconditioners. These approximate (or first-pass) solutions are compared against the exact solution of Eq.2 obtained from a direct solver, labeled here as DNS. The pressure fields of PLMM and PNM are in excellent agreement with those of DNS. However, the velocity magnitudes of PLMM are visibly much more accurate than those of PNM. Specifically, PNM's velocity appears more discontinuous at the pores, where multiple throats intersect. That said, PNM captures the global velocity field well.

Table 2 summarizes the L_2 errors of the pressure and velocity-magnitude fields in Figs.7-8 for all domains. They are calculated using the following formulae, to ensure consistency with and allow comparison against [19]:[4]

$$E_2^{\xi} \equiv \left(\frac{1}{|\Omega|} \int_{\Omega} \left(E_p^{\xi} \right)^2 d\Omega \right)^{1/2} \qquad E_p^{\xi} \equiv \frac{|\xi_S - \xi_M|}{\sup_{\Omega} |\xi_S|} \times 100 \tag{27}$$

[3] Software settings were: porosity = 0.25, particle size = 0.05, particle rotation = [1,1,1], option = random isotropic, correlation weight = 1.

[4] The factor $1/|\Omega|$ in Eq.27 had been mistakenly omitted from Eq.27 in [19], but accounted for in the numerical results reported therein.

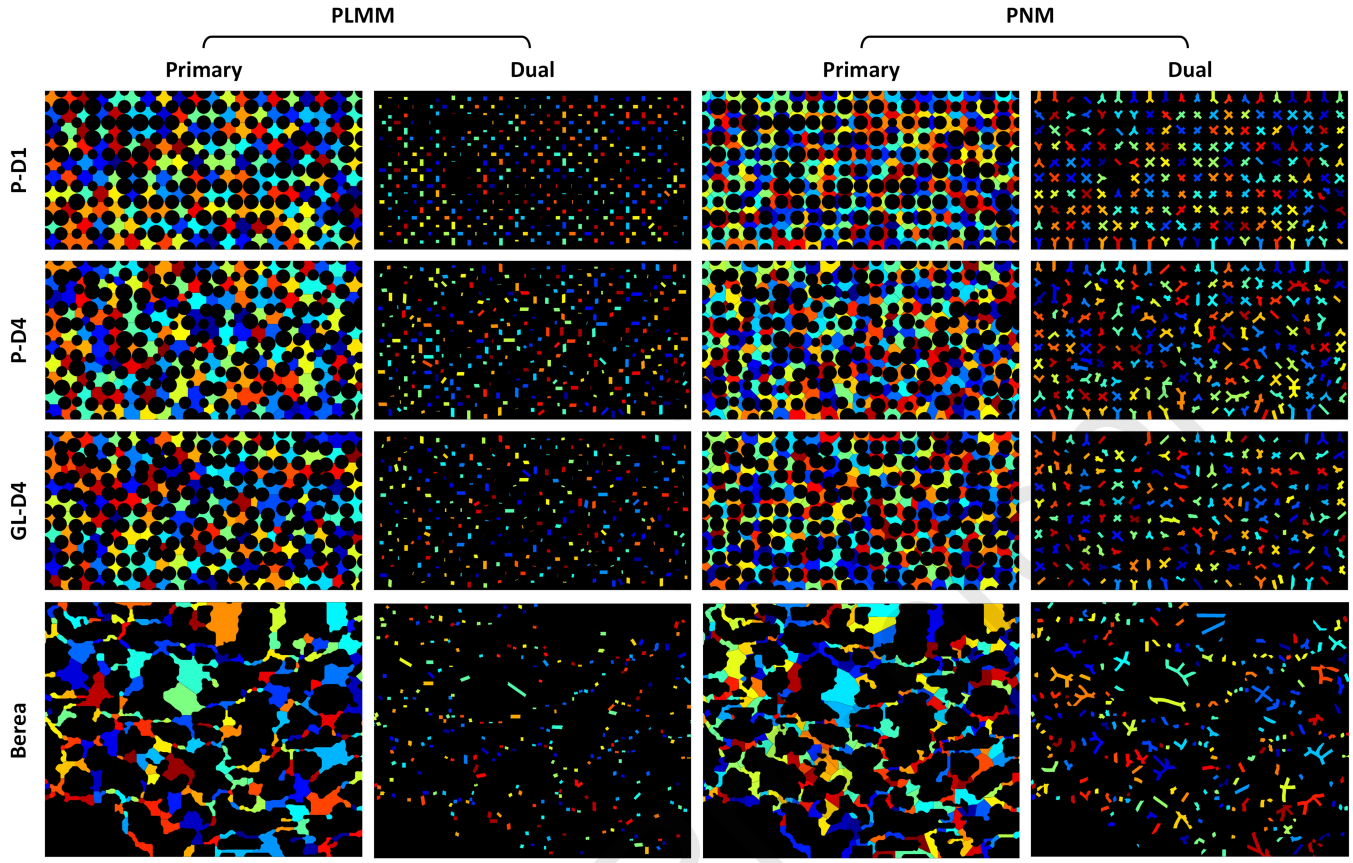


Figure 5: Decomposition of the 2D void-space geometries in Fig.4 into primary grids and dual grids using the PLMM and PNM algorithms described in Sections 3.2.1 and 3.3.1. Each primary or dual grid is highlighted as a randomly colored region. Primary grids form a non-overlapping partition of the void space. Dual grids cover the contact interfaces shared between the primary grids, but do not cover the entire void space.

ξ is a placeholder for either pressure, p, or velocity magnitude, $|\boldsymbol{u}|$. The subscripts M and S refer to the approximate solution obtained from the monolithic PLMM/PNM and the exact solution of Eq.2, respectively. E_2^{ξ} and E_p^{ξ} represent the L_2 and pointwise errors of ξ over Ω, respectively. The values in Table 2 denote E_2^{ξ}. Fig.9 shows the corresponding pointwise errors for the P-D4 and Granular domains. All other domains exhibit similar spatial error distributions and are thus omitted. Notice from Table 2 and Fig.9 that the errors from PLMM are nearly an order of magnitude smaller than those from PNM. This is the result of the more accurate closure BCs imposed during the reduction step outlined in Section 3.2.2. The physical justification of this was briefly stated there and detailed in [19]. Fig.9 shows that errors tend to concentrate near the contact interfaces, which coincide with throats in PLMM but with pores in PNM. The latter is clearly the worse option. Hence, the M_G of the monolithic PLMM is a superior approximate solver than that of PNM, which is consistent with the geometric analysis of [19]. Both yield approximate solutions that are globally flux conservative (i.e., $\nabla \cdot \boldsymbol{u} = 0$ is honored pointwise inside primary grids and in integrated sense across interfaces), thus usable in many practical settings due to their low absolute error: $E_2^{\xi} < 1\%$ for PLMM and $E_2^{\xi} < 5\%$ for PNM.

In Figs.7-9 and Table 2, the FVM restriction matrix, R_{FVM}, in Eq.16b was used to build M_G. Fig.10 shows the pressure and velocity magnitudes for the P-D4 domain when the FEM restriction matrix, R_{FEM}, in Eq.16a is used instead. The approximations are very poor compared to those of Figs.7-8. The inaccuracy is mainly due to the fact that R_{FVM} ensures (integrated) flux conservation across contact interfaces, whereas R_{FEM} does not. Similar plots were obtained for all other domains (not shown). Thus, we abandon R_{FEM} hereafter and use R_{FVM} in all later analyses.

Journal Pre-proof

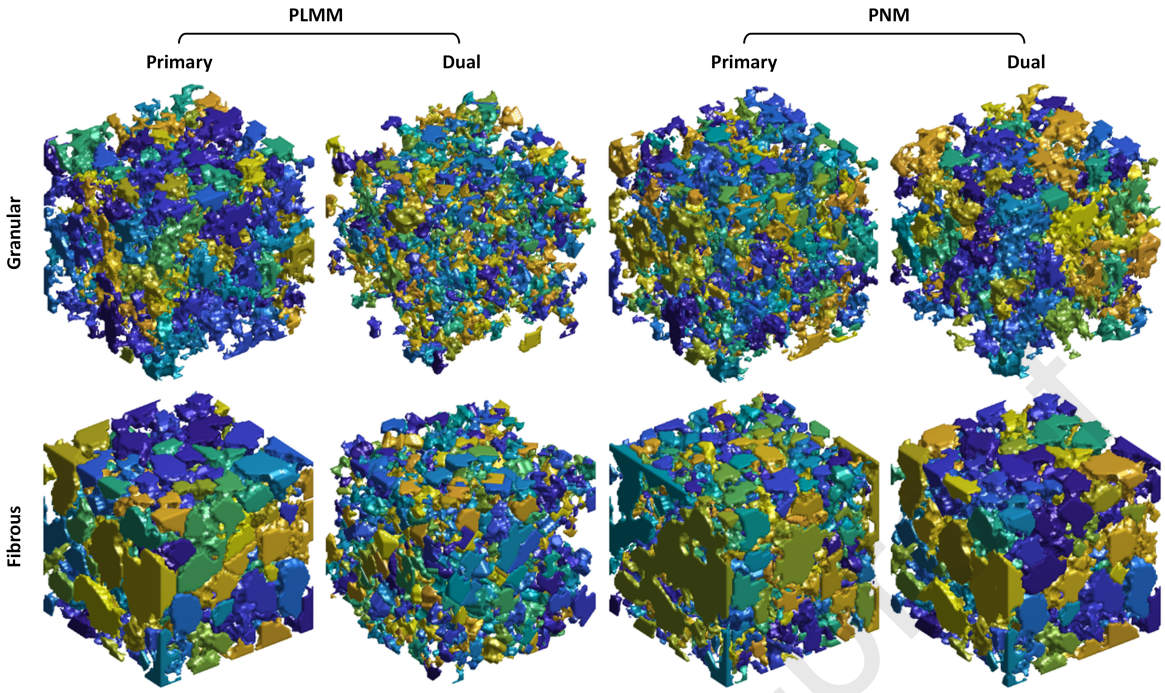


Figure 6: The caption here is the same as Fig.5 except that it applies to the 3D void-space geometries shown in Fig.4.

6.2. *Monolithic PLMM versus monolithic PNM in accelerating Krylov solvers*

We next compare the performance of the monolithic PLMM preconditioner to that of the monolithic PNM preconditioner in accelerating the convergence of a GMRES solver. As discussed, these preconditioners are obtained by pairing the $\mathrm{M_G}$ from Section 3.2.2 (3.3.2) with the local smoother $\mathrm{M_L}$ from Section 3.2.3 (3.3.3) for PLMM (PNM). The pairing is done via Eq.3a. We note that applying the smoother in more than one stage, via Eq.3b with $n_{st} > 1$ and the base smoother M_l set to $\mathrm{M_{dp}}$ in Eq.20, did not improve the convergence rate enough to compensate for the added cost. Hence, we set $n_{st} = 1$ herein. The monolithic PLMM and PNM preconditioners are used to solve the discretized Stokes system in Eq.2 for all of the domains in Fig.4 via a right-preconditioned GMRES solver. The GMRES restart value is set to 20 and a direct solver is used to solve all local problems on Ω_i^p and Ω_i^d. Table 3 summarizes the wall-clock times (WCTs) required to construct the preconditioners, which are broken down into the costs of building the global preconditioner, $\mathrm{M_G}$, and the local smoother, $\mathrm{M_L}$. The smoother's cost is due to LU factorizing the local matrices $\hat{\mathrm{A}}_{p_i}$ and $\hat{\mathrm{A}}_{d_i}$ in Eq.19 to speedup its repeated application. Table 4 lists the number of iterations and WCTs required by GMRES to converge to a normalized residual of $\|\hat{\mathrm{A}}\hat{x} - \hat{b}\|/\|\hat{b}\| < 10^{-8}$. If this criterion could not be satisfied in under 300 iterations, the solver is said to have "diverged." All computations are performed in series.

Table 2: L_2 errors (%) of the pressure and velocity-magnitude fields shown in Figs.7-8 obtained from a single application of the global preconditioner, $\mathrm{M_G}$, associated with the monolithic PLMM and PNM preconditioners. The FVM restriction matrix in Eq.16b is used to build $\mathrm{M_G}$ for both.

	PLMM		PNM	
Domain	Pressure	Velocity	Pressure	Velocity
P-D1	0.085	0.11	3.80	2.73
P-D4	0.31	0.58	5.01	3.58
GL-D4	0.14	0.39	1.79	1.76
Berea	0.24	0.56	0.55	4.12
Granular	0.72	1.70	4.35	5.66
Fibrous	0.89	1.48	3.31	5.31

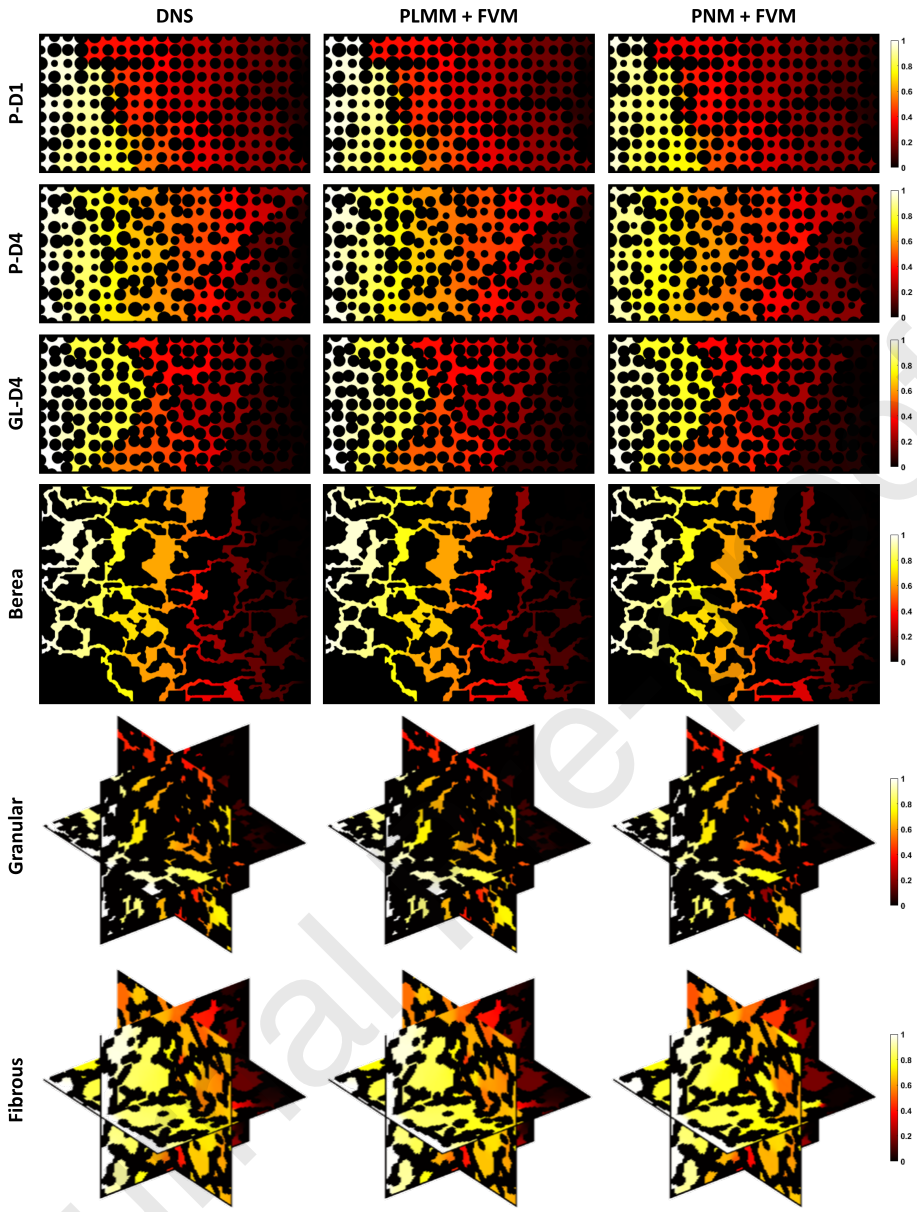


Figure 7: Comparison of pressure fields obtained from a single application of the global preconditioner, M_G, associated with the monolithic PLMM and PNM preconditioners against DNS. The FVM restriction in Eq.16b is used to build M_G. The corresponding velocity fields are shown in Fig.8. The pressure fields in each row have been normalized by the maximum DNS value so that they fall between 0 and 1.

Three observations can be made: (1) The build-times of PLMM and PNM are comparable. The only exception is the Fibrous domain, where PLMM is more costly with its build-time dominated by the cost of M_G. This is due to the presence of a few large pores (primary grids in PLMM; see Fig.6) combined with the use of a direct solver to build shape functions (Eq.15b). Refining the decomposition [33] and/or using an iterative solver to build the shape functions, would remove the discrepancy; (2) The GMRES run-times of PLMM are 1.5–2.3 times smaller than PNM in all cases, except Fibrous for reasons similar to those just stated; (3) PLMM requires 1.3–2.2 times fewer iterations to converge than PNM in all domains. This is made clearer by Fig.11, where the normalized residual is plotted against the number of GMRES iterations. In all domains, the monolithic PLMM preconditioner converges faster than the monolithic PNM. These observations are not surprising in light of the more accurate approximate (first-pass) solutions

Journal Pre-proof

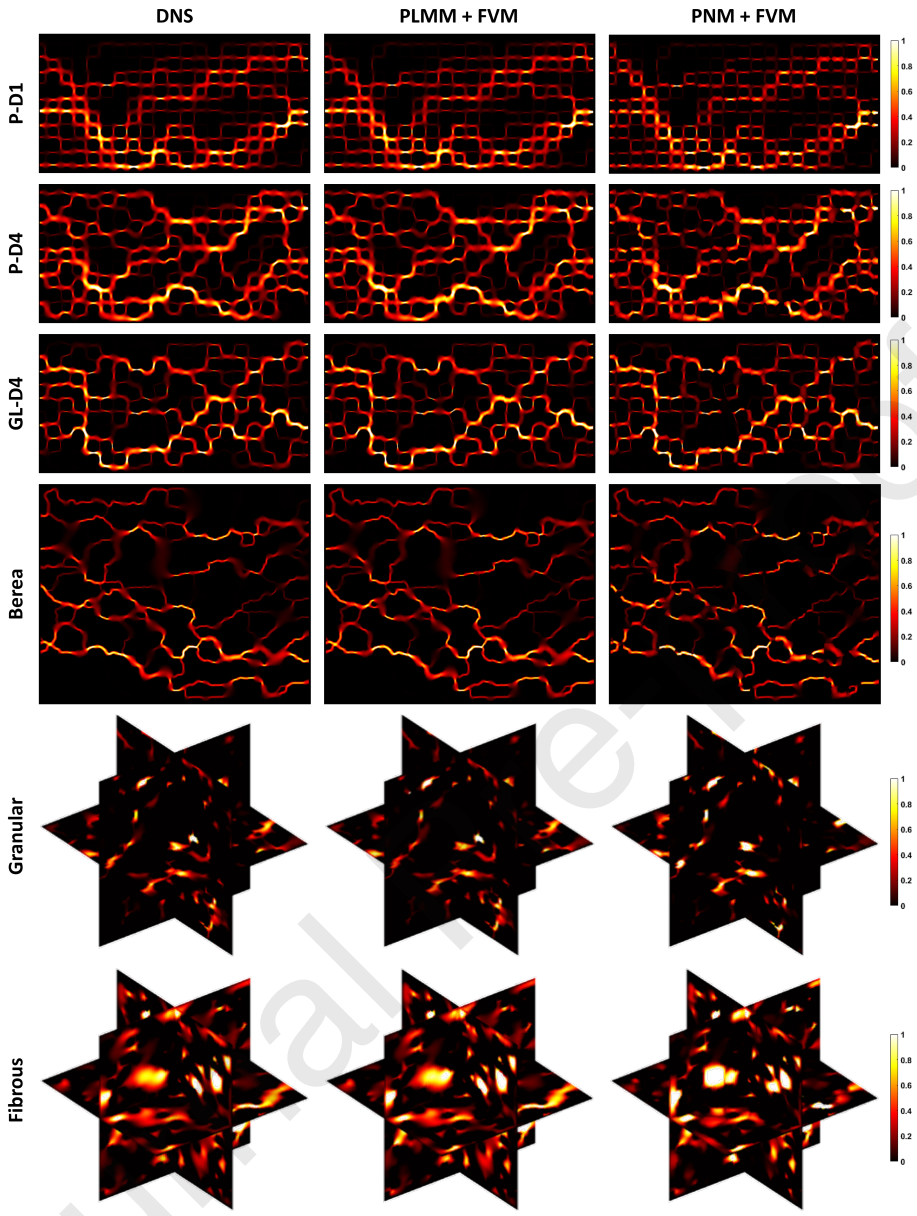


Figure 8: This plot contains the velocity-magnitude fields associated with the pressure fields in Fig.7. The caption is otherwise the same. The velocity magnitudes in each row have been normalized by the maximum DNS value so that they fall between 0 and 1.

obtained in Section 6.1 by the M_G of PLMM versus the M_G of PNM. Hence, the monolithic PLMM preconditioner is the preferred choice in iterative solvers for the Stokes Eq.1 on large and geometrically complex, porous domains.

6.3. Monolithic versus block preconditioners in accelerating Krylov solvers

Table 3 also includes the WCTs associated with constructing the block PLMM and block PNM preconditioners of Section 4 for all the domains in Fig.4. The WCTs are broken down into the costs of building the global preconditioners $M_{G,F}$ and $M_{G,Y}$ in Eq.25 (denoted by M_G) and the local smoothers $M_{L,F}$ and $M_{L,Y}$ in Eq.26 (denoted by M_L). Similar to the previous section, the M_L costs are largely due to LU factorizing the local matrices in Eq.26 (i.e., F_{p_i}, F_{d_i}, Y_{p_i}, Y_{d_i}). Table 4 lists the number of GMRES iterations until convergence (i.e., $||\hat{A}\hat{x} - \hat{b}||/||\hat{b}|| < 10^{-8}$) and the corresponding

Journal Pre-proof

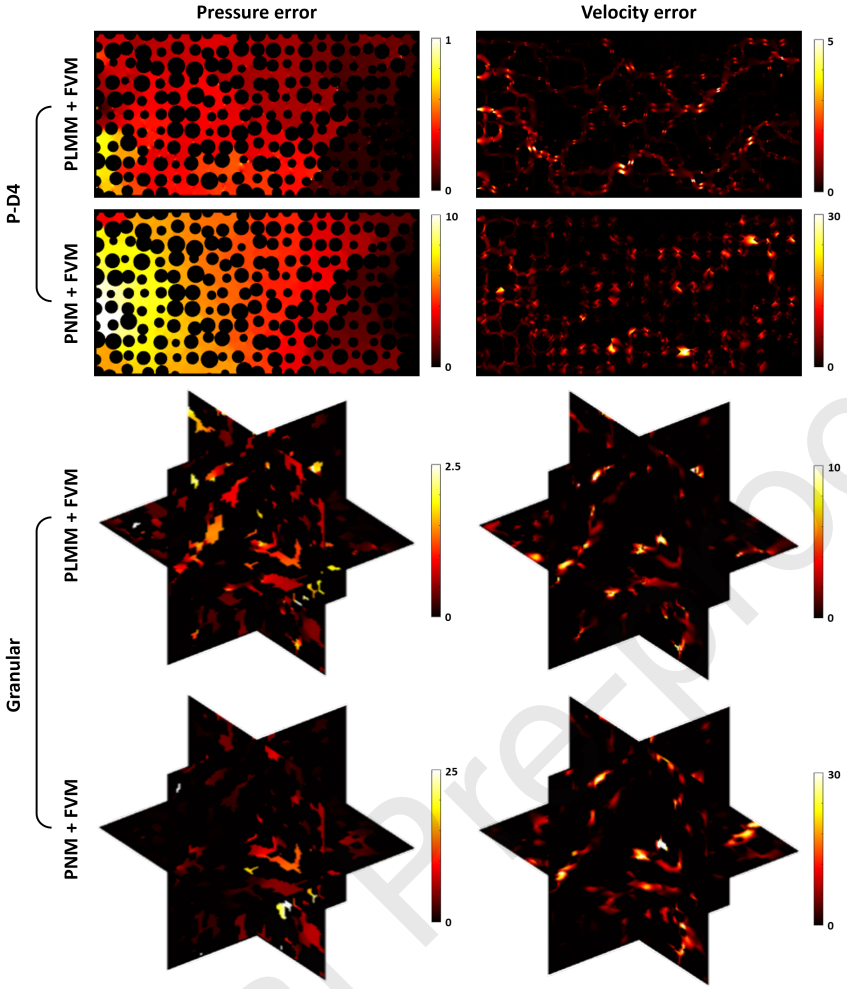


Figure 9: Pointwise errors (%) of pressure (E_p^p) and velocity magnitude ($E_p^{\boldsymbol{u}}$) from a single application of the global preconditioner, M_G, associated with the monolithic PLMM and PNM preconditioners with the FVM restriction (Eq.16b). Results shown only for the P-D4 and Granular domains.

run-times accelerated by the block PLMM and block PNM . Fig.11 shows the convergence patterns of the block PLMM/PNM in terms of the normalized residual versus GMRES iterations. Tables 3-4 and Fig.11 also include the results for the block AMG preconditioner, which we described in Section 4 and adopted as our benchmark.

The block AMG involves solving linear sub-systems associated with the matrices Y and F (i.e., in steps 1a, 1c, and 3 of Section 4) approximately. To do this, a single multigrid V-cycle is accompanied by one pre- and one post-smoothing operation per level using Gauss Seidel. To apply and build the data structures for the V-cycle, we adapted the `amg.m` code from the iFEM GitHub repository [34], wherein a modified Ruge-Stuben [13] coarsening and a two-point interpolation was used; other settings led to similar or inferior results. Given the algorithmic differences and multilevel structure of the block AMG, we only report its total build-time in Table 3.

Focusing here on a comparison between the monolithic PLMM/PNM and block PLMM/PNM, we make the following observations: (1) The total build-times (M_G + M_L) associated with the block and monolithic preconditioners are comparable. The costs of M_G are nearly identical because the prolongation matrices of the block PLMM/PNM are extracted directly from their monolithic counterparts via Eq.24. In other words, to build the block PLMM/PNM, we must construct the monolithic PLMM/PNM first. The costs associated with M_L are lower for the block than the monolithic preconditioners by a factor of 2–3, because LU factorization scales nonlinearly with matrix size. In block PLMM/PNM, local matrices are smaller because velocity and pressure unknowns are decoupled; (2) In all domains,

Journal Pre-proof

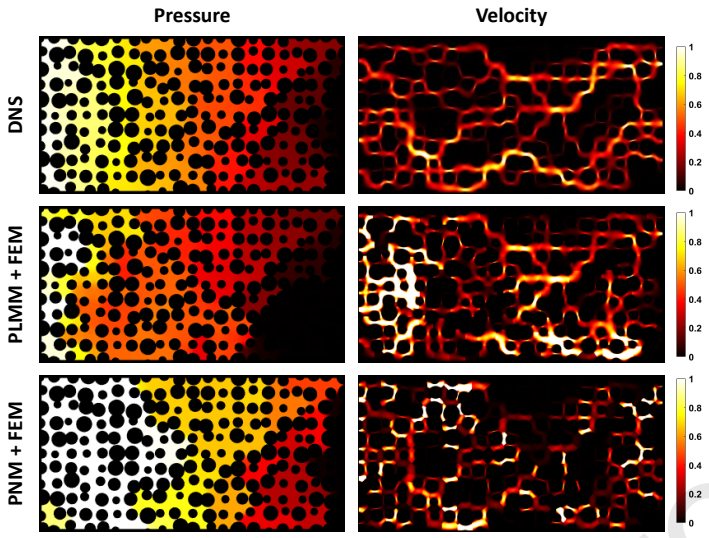


Figure 10: Comparison of the pressure and velocity-magnitude fields obtained from a single application of the global preconditioner, M_G, associated with the monolithic PLMM and PNM preconditioners against DNS. Here, the FEM restriction in Eq.16a is used to build M_G, which results in poor approximations compared to the same preconditioners built via the FVM restriction in Eq.16b (see Figs.7-8). Plots correspond to the P-D4 domain. The pressure and velocity-magnitude fields have been normalized by the maximum DNS values so that they fall between 0 and 1.

Table 3: Wall-clock times (WCTs; "min") associated with constructing the monolithic (PLMM, PNM) and block (PLMM, PNM, AMG) preconditioners in each domain. The WCTs of monolithic/block PLMM/PNM are broken down into the costs of building the global preconditioner, M_G, and the local smoother, M_L. Building the smoother consists of performing LU factorization of the local systems ($\hat{A}_{p_i}$ and $\hat{A}_{d_i}$ in Eq.19, and F_{p_i}, F_{d_i}, Y_{p_i}, Y_{d_i} in Eq.26). Because of its multilevel structure, only the total cost is reported for building the block AMG preconditioner.

	Monolithic		Block		
Domain	PLMM	PNM	PLMM	PNM	AMG
P-D1	M_G: 2.28 M_L: 2.05	M_G: 2.06 M_L: 1.71	M_G: 1.93 M_L: 0.60	M_G: 2.01 M_L: 0.65	tot: 0.27
P-D4	M_G: 2.44 M_L: 2.07	M_G: 2.29 M_L: 1.99	M_G: 2.24 M_L: 0.65	M_G: 2.09 M_L: 0.66	tot: 0.25
GL-D4	M_G: 1.74 M_L: 1.29	M_G: 1.72 M_L: 1.41	M_G: 1.71 M_L: 0.50	M_G: 1.50 M_L: 0.52	tot: 0.19
Berea	M_G: 2.26 M_L: 1.83	M_G: 1.98 M_L: 1.75	M_G: 2.32 M_L: 0.68	M_G: 2.14 M_L: 0.70	tot: 0.29
Granular	M_G: 1.01 M_L: 0.17	M_G: 0.38 M_L: 0.19	M_G: 1.07 M_L: 0.09	M_G: 0.42 M_L: 0.11	tot: 0.03
Fibrous	M_G: 6.31 M_L: 0.82	M_G: 0.79 M_L: 0.82	M_G: 6.07 M_L: 0.27	M_G: 0.74 M_L: 0.26	tot: 0.05

the monolithic preconditioners converge in far fewer GMRES iterations, by a factor of 3.5–5.5, than the block preconditioners; (3) The corresponding run-times of monolithic PLMM/PNM are also smaller than block PLMM/PNM. Concretely, monolithic PLMM is faster than block PLMM by a factor of 2–3.3 in WCTs, with the exception of the Fibrous domain where the block PLMM preconditioner outperforms by a factor of 2. Similarly, the monolithic PNM is faster than the block PNM by a factor of 1.2–5.6 in WCTs for all the domains. Hence, the monolithic PLMM/PNM preconditioners are clearly the preferred choice over the block PLMM/PNM preconditioners.

6.4. Block PLMM and block PNM versus block algebraic multigrid

Focusing next on an intercomparison between the block PLMM, block PNM, and block AMG preconditioners, the following observations stand out from Tables 3-4 and Fig.11: (1) In nearly all cases, block PLMM converges in the fewest number of GMRES iterations, followed by block PNM, then block AMG; (2) This order is preserved in the

Journal Pre-proof

Table 4: The number of iterations ("iter") and wall-clock times ("min") required by the right-preconditioned GMRES to converge (i.e., satisfy $\|\hat{A}\hat{x} - \hat{b}\|/\|\hat{b}\| < 10^{-8}$) in each domain. Results for the monolithic (PLMM, PNM) and block (PLMM, PNM, AMG) preconditioners are listed. "NA" means the solver did not converge within 300 iterations. All simulations are run on a serial machine. R_{FVM} in Eq.16b is used in PLMM and PNM.

	Monolithic		Block		
Domain	PLMM	PNM	PLMM	PNM	AMG
P-D1	19 iter 2.4 min	35 iter 3.8 min	93 iter 8.0 min	117 iter 11.3 min	206 iter 32.1 min
P-D4	31 iter 3.8 min	43 iter 5.0 min	111 iter 10.9 min	180 iter 21.6 min	NA NA
GL-D4	22 iter 1.8 min	33 iter 2.7 min	92 iter 6.0 min	118 iter 8.5 min	NA NA
Berea	25 iter 3.1 min	44 iter 5.4 min	138 iter 14.4 min	206 iter 30.5 min	181 iter 27.8 min
Granular	12 iter 0.6 min	27 iter 1.4 min	35 iter 1.2 min	113 iter 6.2 min	NA NA
Fibrous	11 iter 2.5 min	13 iter 1.5 min	36 iter 1.3 min	55 iter 1.8 min	NA NA

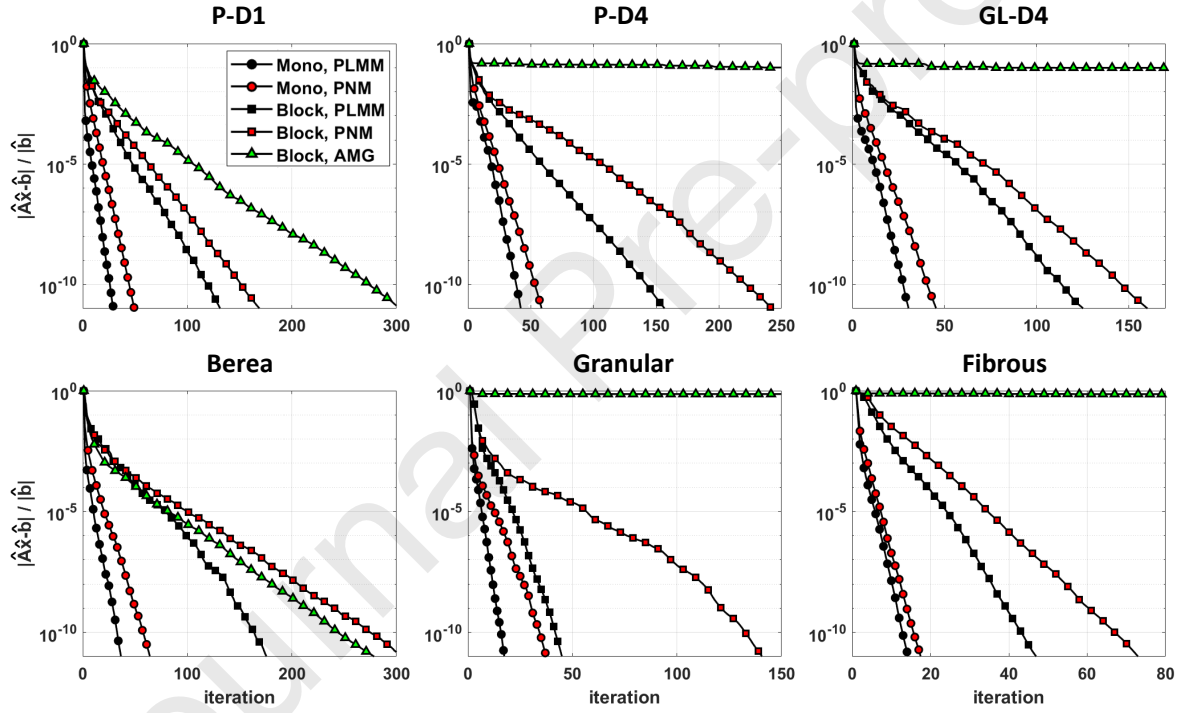


Figure 11: Normalized residual versus the number of GMRES iterations for the monolithic PLMM, monolithic PNM, block PLMM, block PNM, and block AMG preconditioners applied to all the domains in Fig.4. Monolithic PLMM/PNM use the FVM restriction matrix (Eq.16b).

corresponding GMRES run-times, with the block PLMM preconditioner being 1.4–5 times faster than block PNM; Moreover, (3) in four out of six of the domains (i.e., P-D4, GL-D4, Granular, and Fibrous), block AMG *diverges*, whereas block PLMM and block PNM converge in *all* cases (and so do the monolithic PLMM/PNM preconditioners). We think this is because the void space of these four domains is more poorly connected (with many choke points) than P-D1 and Berea; Finally, (4) while block AMG is an order of magnitude cheaper to build than block PLMM/PNM, its GMRES run-time is much larger. This is especially the case for the four divergent simulations that were terminated after 300 iterations, or roughly 30 mins (see Fig.11). Hence, the combined cost of building and applying the block AMG preconditioner is much higher than that of the block PLMM or block PNM. The main implication of these ob-

Journal Pre-proof

servations is that repurposing the monolithic PLMM/PNM to build the block PLMM/PNM preconditioners in Section 4, at no additional cost, results in more robust and efficient performance than simply using off-the-shelf block AMG preconditioners to approximately solve the sub-systems involving matrices Y and F in Section 4.

7. Discussion

7.1. PLMM versus PNM as a model and a preconditioner

In [19], a geometric formulation of PLMM for the Stokes Eq.1 was presented and compared to a geometric formulation of PNM [35]. The term "geometric" here means that both PLMM and PNM were formulated as computational *models*, wherein a domain is explicitly decomposed, the governing equations are discretized, and geometric and material parameters of the domain/grid are used explicitly as input. This work presents a first translation of these two models into fully algebraic, *monolithic* preconditioners, in which *no* knowledge of the problem parameters is required. Their starting point is the linear system in Eq.2, which can be assembled via any existing code. This renders the preconditioner formulations of PLMM/PNM portable and amenable to non-intrusive implementation. When used as approximate solvers (i.e., without smoothers as described in Section 6.1), monolithic PLMM/PNM are equivalent to the PLMM/PNM models.[5] In [19], the authors found that the PLMM model yields more accurate approximate solutions than the PNM model (i.e., permeability errors were five times lower). We see the same trend here: the monolithic PLMM solver is more accurate than the monolithic PNM solver by nearly an order of magnitude in L_2 and pointwise errors (Table 2 and Fig.9). The higher accuracy of the PLMM solver has led, in turn, to the faster convergence of GMRES when preconditioned by the monolithic PLMM preconditioner over the monolithic PNM preconditioner. Since the block PLMM/PNM preconditioners are derived from their monolithic counterparts, a similarly better performance of GMRES was observed for the block PLMM over the block PNM preconditioner. The superiority of PLMM over PNM as a model, approximate algebraic solver, and monolithic/block preconditioners is no coincidence. The reason lies in PLMM's decomposition of the void space at geometric constrictions (not enlargements as in PNM), or throats (not pores), and imposing the closure BCs $p = \text{const}$ and $\partial_n \boldsymbol{u} = 0$. The optimality of this decomposition paird with these BCs was detailed in [19]. Put differently, PLMM is simply better at *decoupling* Eq.1 spatially than PNM.

7.2. Algebraic versus geometric PLMM

Aside from increased portability and non-intrusive implementation in existing codes, the algebraic form of PLMM as a monolithic preconditioner has one more advantage over its geometric counterpart in [19]. In high-porosity domains (e.g., Fibrous in Fig.4), the geometric PLMM model would result in sample-spanning dual grids due to the requirement that overlapping dual grids be merged [19]. Such large dual grids result in local problems that are very expensive to solve, defeating the purpose of accelerating calculations via PLMM in the first place. The algebraic formulation of PLMM removes this drawback because the dual grids are allowed to overlap, and their sub-problems are commensurately smaller. Recall that dual grids here are used to apply the additive-Schwarz smoother M_{dp} in Eq.20. In theory, it is possible to adapt the geometric PLMM of [19] so that overlapping dual grids are not merged, and used much like subdomains in a geometric additive Schwarz method [30]. But for the (image-based) Cartesian grids herein, we encountered significant challenges in [19] to implement the various mappings involved.

7.3. PNM as an accelerator of DNS

In modeling fluid flow through porous media, the pore space is often captured by an X-ray μCT image that is rather large. It is, thus, customary to make a choice about whether to use direct numerical simulation (DNS) or pore network modeling (PNM) to solve the Stokes Eq.1. In DNS, the governing equations are discretized with a preferred method (e.g., FVM, FEM) to yield a linear system like Eq.2, which is then solved exactly with an iterative or direct solver. In existing PNM, the geometry of the void space is simplified by a computational graph, whose nodes approximate the pore shapes (e.g., sphere, cubes) and edges approximate the throat shapes (e.g., cylinders, prisms). The arguments used to justify either choice are often presented as a dichotomy [22, 23, 36]: PNM is fast but less accurate and DNS is slow but more accurate. The real drawback of PNM is that it lacks the ability to estimate and control approximation

[5] The first-pass solution of monolithic PLMM is equivalent to the M_0 approximation (no corrective iterations) of the geometric PLMM in [19].

Journal Pre-proof

errors, unlike DNS, rendering its predictions unreliable [21]. In this paper, we provided a different perspective that removes this dichotomy, and associated drawbacks of PNM, altogether: *use PNM as a preconditioner to accelerate DNS*. The monolithic PNM preconditioner herein can be used to yield approximate solutions (see Section 6.1), like existing PNM models, or called from within DNS to accelerate its convergence. The latter bestows PNM with error control and estimation capabilities that have thus far been lacking. The dichotomy is removed because a DNS method equipped with this preconditioner can *stop* at any point before convergence to yield an approximate solution (with error bars) that is guaranteed to be globally flux-conservative. Concretely, this can be done by applying M in Eq.3a within a Richardson loop [29, 37]. The solution can even be made locally (or pointwise) conservative by solving a special (and inexpensive) dual-grid problem described in [7]. These statements apply equally to the monolithic PLMM. We note the monolithic PNM's first-pass solution is different from and more accurate than existing PNM. Unlike classical PNM [20], the void geometry here is *not* approximated by simplified shapes, and a *pointwise* velocity field can be reconstructed over the entire fine grid. Existing PNM only yields *integrated* flowrates through throats. Moreover, unlike classical PNM, the need to *extract* an explicit network, a potentially costly task [24], is obviated here. While watershed segmentation, an $O(N)$ operation [33], is still a key step in monolithic PNM, all downstream operations (e.g., graph pruning, node merger, shape approximation) are made redundant (see Appendix A of [35]).

7.4. Monolithic versus block preconditioning

In Section 6, we saw that the monolithic PLMM/PNM preconditioners significantly outperform *all* block preconditioners in nearly all domains (except Fibrous, where block PLMM was faster than monolithic PLMM), both in terms of number of GMRES iterations and wall-clock times (WCTs). Moreover, monolithic PLMM was 1.5-2.3 times faster in WCTs than monolithic PNM, making it the most successful preconditioner in solving the Stokes Eq.1. We also saw that the block PLMM/PNM preconditioners, derived from their monolithic counterparts, handily outperformed existing block AMG. The latter even diverged in four out of six domains. We note the build-times of the preconditioners were not a factor in our takeaways because the costs of constructing all monolithic/block PLMM/PNM preconditioners are comparable, and the total cost of block AMG is dominated by its run-(not build-)-time. The block AMG here was based on the scaled-BFBt approximation of the Schur complement matrix, $\mathrm{S} = -\mathrm{G}\mathrm{F}^{-1}\mathrm{G}^{\top}$, via Eq.23 [15]. As discussed in Section 4, other approximations of S were also tested but all performed worse than Eq.23. Even though the BFBt approximation is considered more suitable for the Oseen equation in the literature, where the inertial term is included in Eq.1, and that cheaper alternatives (e.g., the diagonal mass matrix X_1 in Section 4) are said to be preferred for the Stokes Eq.1, we think the geometric complexity of the domains here offer the BFBt approximation a clear advantage. Even so, *any* approximation of S, by definition, neglects some coupling terms between p and $\boldsymbol{u}$ in Eq.2, which are largely preserved by the monolithic PLMM/PNM. This is likely why the latter performs more efficiently.

7.5. Parallel scalability

While all simulations in Section 6 were conducted on a serial machine, the building and application of the monolithic PLMM/PNM and block PLMM/PNM preconditioners within iterative solvers is fully parallelizable. Hence, the WCTs reported in Tables 3-4 would be even less if the simulations were run on a parallel machine. In building any of the proposed preconditioners, the most computationally expensive step is constructing the prolongation matrix, P, in Eq.15. However, this involves computing a set of shape vectors, $p_{c_j}^{p_i}$, and correction vectors, c^{p_i}, that are spatially *decoupled* over the primary grids. Hence, if $\#C^p$ denotes the average number of contact interfaces per primary grid, and N^p is the total number of primary grids, then a maximum of $\#C^pN^p$ parallel processors can be employed to reduce the cost of building P to that of solving a single linear sub-system on *one* primary grid. Notice, the prolongation matrix P (Eq.15), the restriction matrix R (Eq.16), the reduction matrix Q (Eq.11b), and the permutation matrix W (Eq.4a) need to be computed only *once*. They are reused across all iterations of a Krylov solver. In applying any of the proposed preconditioners, the most computationally costly step is applying the additive-Schwarz smoothers corresponding to M_{dp} in Eq.20 for the monolithic PLMM/PNM, and $\mathrm{M}_{\mathrm{L,F}}$ and $\mathrm{M}_{\mathrm{L,Y}}$ in Eq.26a for the block PLMM/PNM preconditioners. However, notice again that applying these smoothers entails solving a set of spatially decoupled linear sub-systems on the primary and dual grids, which is an embarrassingly parallel task. While a rigorous parallel scalability study falls outside the scope of this work, future work should quantify and demonstrate its magnitude.

Journal Pre-proof

7.6. Future extensions

The proposed monolithic/block PLMM/PNM preconditioners are designed for Stokes flow, resulting in a symmetric saddle-point system with $\mathrm{F} = \mathrm{F}^\top$ in Eq.2. For the Navier-Stokes or Oseen equations where inertial forces are present, $\mathrm{F} \neq \mathrm{F}^\top$ and the system is non-symmetric. We expect our preconditioners to still perform well under these circumstances, as long as the non-symmetric F is used to build the prolongation matrix P in Eq.15. For low to moderate Reynolds numbers, even preconditioners built for Stokes flow ($\mathrm{F} = \mathrm{F}^\top$) may be adequate in accelerating solver convergence, but performance is expected to degrade as Reynolds number grows. Another extension pertains to flow through domains with unresolved porosity. Such problems are described by the Darcy-Brinkman-Stokes (DBS) equation [38], where an extra Darcy term (linear in velocity) is added to Eq.1a. This term modifies the diagonal entries of F, thus preserving symmetry. Again, our preconditioners are expected to apply to such problems, but the watershed-based decompositions described in Sections 3.2.1 and 3.3.1 may need modification to prevent performance degradation [39]. In flow problems where the pore-space geometry evolves with time, for example due to dissolution or precipitation of solids, the preconditioners may need to be periodically updated or rebuilt entirely. Since the latter can be costly, adaptive strategies similar to [26] are likely required to update only select columns of P during each time step. Finally, immiscible two-phase flow, or solute transport that significantly alters the fluid's rheology, would likely benefit from a similar treatment. These problems consist of a flow (mass plus momentum balance) and a transport (phase/solute-indicator balance) equation that are often solved sequentially. The flow equation, which is generally more expensive, could benefit from our preconditioners with periodic/adaptive update of P's columns. Update criteria may be devised based on the geometric PLMM for two-phase flow in [7]. Future research should test the above hypotheses.

8. Conclusions

In this work, we have developed monolithic and block preconditioners to robustly and efficiently solve saddle-point systems that arise from discretizing the Stokes Eq.1 via iterative solvers. They include the monolithic PLMM, monolithic PNM, block PLMM, and block PNM preconditioners. We compared their performances against each other and benchmarked them against a standard block AMG preconditioner on multiple 2D/3D, geometrically complex, porous domains. The following summarizes our key contributions and findings:

- The monolithic PLMM preconditioner is a fully algebraic reformulation of the geometric PLMM model in [19]. The preconditioner form renders PLMM portable and allows its non-intrusive implementation in existing codes. It also removes certain drawbacks of the geometric variant regarding sample-spanning dual grids (Section 7.2).
- The monolithic PNM preconditioner is a first, fully algebraic reformulation of the widely used PNM model in the literature [20]. This enables using PNM as a means to accelerate DNS, instead of the commonly held view that one must trade off accuracy for speed when using PNM over DNS to solve Eq.1. The preconditioner form also allows estimating and controlling the errors of PNM, which has thus far proven to be elusive.
- The monolithic PLMM/PNM preconditioners can also be used as *approximate solvers*. This involves applying the global preconditioner $\mathrm{M_G}$ associated with the monolithic PLMM/PNM to the RHS vector of Eq.2, i.e., $\hat{x}_{aprx} = \mathrm{M_G^{-1}}\hat{b}$. This is equivalent to the approximate solutions produced by existing geometric PNM and PLMM models (without corrective iterations). Moreover, the accuracy of $\hat{x}_{aprx}$ can be progressively improved by combining $\mathrm{M_G}$ with the smoother $\mathrm{M_L} = \mathrm{M_{dp}}$ in Eq.20 and applying the combined M in Eq.3 within a Richardson-type iteration. Each iteration would be globally flux-conservative, thus usable as an intermediate solution.
- The first-pass solution $\hat{x}_{aprx} = \mathrm{M_G^{-1}}\hat{b}$ of the monolithic PNM is more accurate than classical PNM models in the literature. This is because, here, the pore-space geometry is *not* simplified and a pointwise velocity field can be constructed on the fine grid. There is also no need to *extract* an explicit network, which may be costly [24].
- The monolithic PLMM/PNM can be repurposed to build the block PLMM/PNM preconditioners at no added cost. The latter is more robust/efficient than existing block AMG, which can diverge in complex domains.
- In terms of WCTs, the performance of all preconditioners herein are ranked from best to worst as follows: monolithic PLMM, monolithic PNM, block PLMM, block PNM, and block AMG. The monolithic/block PLMM/PNM preconditioners are amenable to parallel computing in building and applying them within iterative solvers. The monolithic preconditioners must utilize the FVM (not FEM) restriction matrix in Eq16b to be fast/accurate.

Journal Pre-proof

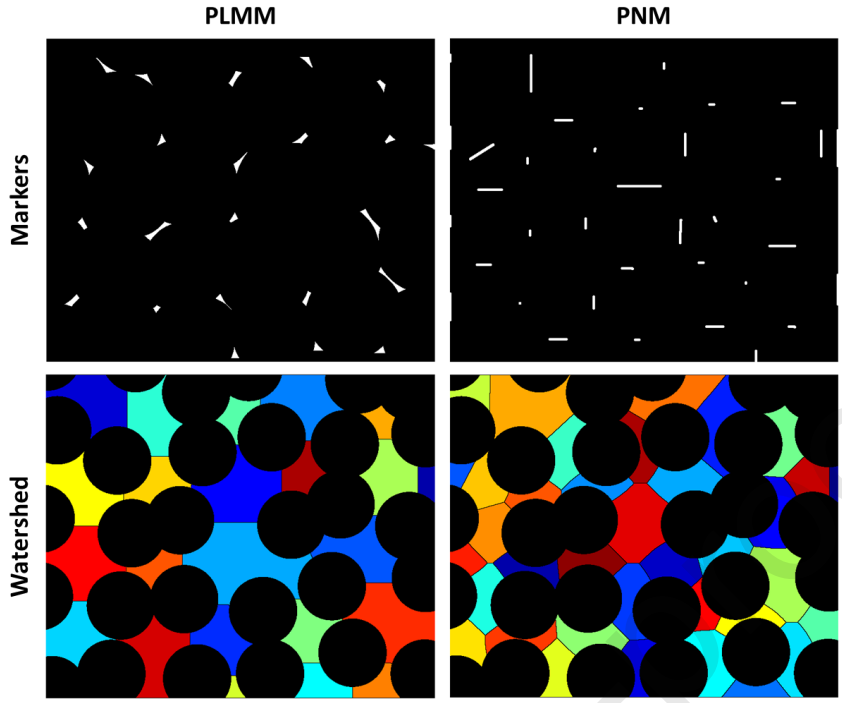


Figure A.1: Markers (or "seeds") used to perform the watershed transform in the PLMM and PNM domain decompositions.

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. Markers used in the watershed decompositions of PLMM and PNM

To perform the marker-based watershed transforms described in Sections 3.2.1 and 3.3.1, "seeds" (or markers) must be provided as input. Fig.A.1 shows these markers for the domain decompositions used in the PLMM and PNM preconditioners for the pore space shown in Fig.1a. The PLMM markers correspond to the local minima of the distance map in Fig.1d, and the PNM markers to the minima of the distance map in Fig.1g. Notice from Fig.A.1 that the PNM markers also correspond to the contact interfaces shared between the primary grids of the PLMM decomposition.[6]

References

[1] 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.

[2] 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.

[3] Enrico Barbier. Geothermal energy technology and current status: an overview. Renewable and sustainable energy reviews, 6(1-2):3–65, 2002.

[4] 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.

[6] The PNM markers in Fig.A.1 are slightly more numerous than the PLMM interfaces because a finer PLMM decomposition was used as input for the latter. The refinement is controlled by the size of a "structuring element" used to identify local minimal in the PLMM distance map [33].

Journal Pre-proof

[5] 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.

[6] Dorthe 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.

[7] Yashar Mehmani and Hamdi A Tchelepi. Multiscale formulation of two-phase flow at the pore scale. Journal of Computational Physics, 389: 164–188, 2019.

[8] Blair Perot. Conservation properties of unstructured staggered mesh schemes. Journal of Computational Physics, 159(1):58–89, 2000.

[9] 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.

[10] 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.

[11] 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.

[12] Michele Benzi, Gene H Golub, and Jörg Liesen. Numerical solution of saddle point problems. Acta numerica, 14:1–137, 2005.

[13] John W Ruge and Klaus Stüben. Algebraic multigrid. In Multigrid methods, pages 73–130. SIAM, 1987.

[14] Howard C Elman. Preconditioning for the steady-state navier–stokes equations with low viscosity. SIAM Journal on Scientific Computing, 20(4):1299–1316, 1999.

[15] 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.

[16] 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.

[17] Howard Elman and David Silvester. Fast nonsymmetric iterations and preconditioning for navier–stokes equations. SIAM Journal on Scientific Computing, 17(1):33–46, 1996.

[18] 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.

[19] Yashar Mehmani and Hamdi A Tchelepi. Multiscale computation of pore-scale fluid dynamics: Single-phase flow. Journal of Computational Physics, 375:1469–1487, 2018.

[20] Irving Fatt. The network model of porous media. Trans. AIME, 207:144–159, 1956.

[21] Yashar Mehmani, Timothy Anderson, Yuhang Wang, Saman A Aryana, Ilenia Battiato, Hamdi A Tchelepi, and Anthony R Kovscek. Striving to translate shale physics across ten orders of magnitude: What have we learned? Earth-Science Reviews, 223:103848, 2021.

[22] Yashar Mehmani and Matthew T Balhoff. Mesoscale and hybrid models of fluid flow and solute transport. Reviews in Mineralogy and Geochemistry, 80(1):433–459, 2015.

[23] Paul Meakin and Alexandre M Tartakovsky. Modeling and simulation of pore-scale multiphase fluid flow and reactive transport in fractured and porous media. Reviews of Geophysics, 47(3), 2009.

[24] Arash Rabbani, Peyman Mostaghimi, and Ryan T Armstrong. Pore network extraction using geometrical domain decomposition. Advances in water resources, 123:70–83, 2019.

[25] Yashar Mehmani and Kangan Li. A multiscale preconditioner for microscale deformation of fractured porous media. Journal of Computational Physics, 482:112061, 2023.

[26] 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.

[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] Jeff T Gostick. Versatile and efficient pore network extraction method using marker-based watershed segmentation. Physical Review E, 96 (2):023307, 2017.

[29] Nicola Castelletto, Hadi Hajibeygi, and Hamdi A Tchelepi. Multiscale finite-element method for linear elastic geomechanics. Journal of Computational Physics, 331:337–356, 2017.

[30] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. An introduction to domain decomposition methods: algorithms, theory, and parallel implementation. SIAM, 2015.

[31] 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.

[32] Daniel Niblett, Mohamed Mamlouk, Omar Emmanuel Godinez Brizuela, and Senyou An. Porous microstructure generator (software), 2022.

[33] Sabit Mahmood Khan, Kangan Li, and Yashar Mehmani. Order reduction of fracture mechanics in porous microstructures: A multiscale computing framework. Computer Methods in Applied Mechanics and Engineering, 420:116706, 2024.

[34] Long Chen. *i*FEM: an integrated finite element methods package in MATLAB. Technical report, 2009. URL `https://github.com/lyc102/ifem`.

[35] Yashar Mehmani and Hamdi A Tchelepi. Minimum requirements for predictive pore-network modeling of solute transport in micromodels. Advances in water resources, 108:83–98, 2017.

[36] Xiaofan Yang, Yashar Mehmani, William A Perkins, Andrea Pasquali, Martin Schönherr, Kyungjoo Kim, Mauro Perego, Michael L Parks, Nathaniel Trask, Matthew T Balhoff, et al. Intercomparison of 3d pore-scale flow and solute transport simulation methods. Advances in water resources, 95:176–189, 2016.

[37] 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.

[38] Hendrik C Brinkman. A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles. Flow, Turbulence and Combustion, 1(1):27–34, 1949.

Journal Pre-proof

[39] Bo Guo, Yashar Mehmani, and Hamdi A Tchelepi. Multiscale formulation of pore-scale compressible darcy-stokes flow. Journal of Computational Physics, 397:108849, 2019.

Journal Pre-proof

Journal Pre-proof

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Journal Pre-proof