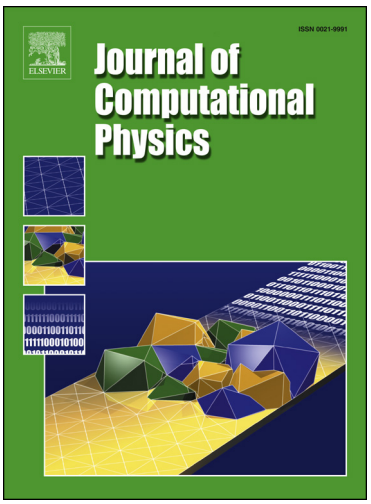
Journal Pre-proof



A Multiscale Preconditioner for Microscale Deformation of Fractured Porous Media

Yashar Mehmani and Kangan Li

PII: S0021-9991(23)00156-0
DOI: https://doi.org/10.1016/j.jcp.2023.112061
Reference: YJCPH 112061

To appear in: *Journal of Computational Physics*

Received date: 27 October 2022
Revised date: 13 February 2023
Accepted date: 6 March 2023

Please cite this article as: Y. Mehmani and K. Li, A Multiscale Preconditioner for Microscale Deformation of Fractured Porous Media, *Journal of Computational Physics*, 112061, doi: https://doi.org/10.1016/j.jcp.2023.112061.

Highlights

- Multiscale preconditioner for deformation of fractured porous media developed.
- The preconditioner accelerates direct numerical simulations at the pore scale.
- Application of the preconditioner in existing codes in fully non-intrusive.
- Strategy for updating the preconditioner in evolving-crack problems is proposed.
- Errors and convergence rates in various microstructures and crack patterns tested.

A Multiscale Preconditioner for Microscale Deformation of Fractured Porous Media

Yashar Mehmani[1†], Kangan Li[1]

[1] *The Pennsylvania State University, Department of Energy and Mineral Engineering, USA*
[†] *Corresponding author: yzm5192@psu.edu, 110 Hosler Building, University Park, PA, 16802*

Abstract. The direct numerical simulation (DNS) of elastic deformation on digitized images of porous materials at the micron (or pore) scale requires the solution of large systems of linear(ized) equations. Krylov solvers are instrumental but suffer from slow convergence without a preconditioner. We present a multiscale preconditioner that significantly accelerates DNS, is scalable on parallel machines, and can be non-intrusively applied in existing codes. The preconditioner is an algebraic reinterpretation of a recent pore-level multiscale method (PLMM) proposed by the authors. It combines a global preconditioner with a local smoother to simultaneously attenuate low- and high-frequency errors, respectively. Like PLMM, a single application of the global preconditioner yields an approximate solution that is sufficiently accurate in a wide range of applications (e.g., geologic CO_2/H_2 storage). The combination with the smoother enables improving the approximation even further. While all cracks here are assumed to be static, we propose an adaptive strategy to update the preconditioner efficiently for evolving-crack problems without affecting the convergence rate of the Krylov solver. We validate the preconditioner against DNS and test its convergence on various 2D/3D microstructures and crack patterns. The agreement and performance are favorable.

Keywords. Porous media, Pore-scale modeling, Preconditioning, Multiscale method, Fracture mechanics

1. Introduction

A mechanistic understanding of mechanical deformation in fractured porous materials is key to several subsurface, manufacturing, and medical applications. They include preserving the integrity of cap rocks in geologic CO_2 sequestration and H_2 storage,[1] inducing hydraulic fractures in geothermal energy and natural gas extraction,[2] designing high-strength lightweight materials for military armors and airplanes,[3,4] understanding the failure of battery electrodes,[5,6] and developing medical treatments for stress fractures in trabecular bones.[7,8] Pore-scale models are valuable tools that provide such understanding by providing the link between the microstructure of a porous sample and its macroscopic response.[9] The main challenge of existing models, however, lies in that they are either overly approximate or computationally expensive.

Approximate models such as the discrete element method (DEM)[10–12] make a series of assumptions about the geometry (e.g., grains are spherical) and material properties (e.g., grains are rigid) of a porous sample that render them suitable for granular media (e.g., soil), but not the amorphous and highly complex microstructures encountered in the above mentioned applications (e.g., bones). Moreover, methods such as DEM lack the ability to estimate and control their approximation errors.[13,14] Direct numerical simulation (DNS) refers to a collection of higher-fidelity alternatives, in which computations are performed directly on the microstructure of a sample captured by a pore-scale image[15] (e.g., X-ray μCT). Examples include the extended finite element method (XFEM),[16] shifted fracture method,[17] phase-field models,[18,19] immersed boundary finite volume method,[20] and peridynamics.[21] In DNS, the governing equations are discretized and solved on a fine grid that often coincides with the image pixels. This results in large linear (or linearized) systems of equations, whose solution is computationally demanding for practical

domain sizes. Iterative, in particular Krylov, solvers are a prime choice for solving such systems, but they require efficient preconditioning to converge at an acceptable rate.[22]

We present a multiscale preconditioner for solving the linear-elastic deformation of porous domains with arbitrary microstructure and crack pattern. It combines a global preconditioner, M_G, with a local smoother, M_L, to simultaneously attenuate low- and high-frequency error modes, respectively. Our main contribution is to develop M_G, as we choose a standard ILU(k) smoother for M_L. The proposed M_G is an algebraic reinterpretation of a recent pore-level multiscale method (PLMM) developed by the authors;[23,24] originally for fluid mechanics.[25–27] PLMM approximates DNS efficiently and accurately by executing the following steps in order: decompose the solid into subdomains via watershed segmentation,[28] construct fine-scale local basis and correction functions on the subdomains, and couple basis/correction functions by solving a global coarse-scale interface problem. Here, we translate these steps into a global preconditioner, the benefits of which are twofold: (1) unlike PLMM, the use of M_G in existing codes is non-intrusive; and (2) the flexible choice of smoothers removes a key limitation of PLMM in controlling its approximation errors. Namely, PLMM requires solving local problems on a second set of subdomains, called *contact grids*, to control errors. But in some (low porosity) microstructures, contact grids span large parts of the domain, rendering local problems to be solved on them costly. Even though we only consider static cracks herein, our second contribution is an efficient way of updating M_G for evolving-crack problems, in which cracks can nucleate and grow by progressive loading. We define an adaptivity criterion that updates M_G infrequently while preserving rapid convergence of Krylov solvers. All computations associated with building and updating M_G are scalable on parallel machines.

Multiscale preconditioners for elastic deformation of porous media are not new, but existing ones target the continuum (or Darcy) scale. Most are based on the multiscale finite element method (MsFE),[29–34] but others for scalar-valued elliptic equations (describing fluid flow) also exist and are based on, e.g., the multiscale finite volume method (MsFV)[35–40] and the multiscale mortar finite element method (MoMsFE),[41–44] among others.[45] In all, a prolongation matrix is built by assembling numerically computed fine-scale basis vectors on coarse grids into its columns. A restriction matrix is built too, often by transposing the prolongation matrix.[22,36] Together, these matrices are used to substantially reduce the size of original fine-grid linear(ized) system. Solving the reduced system yields a coarse-scale solution that is then interpolated onto the fine grid using the basis vectors. The result is an initial approximation, or a first-pass solution, whose errors are reduced further using a local smoother.[22,46] These steps bear a close relation to algebraic multigrid methods (AMG),[47] with the key difference being the prolongation matrices are *informed* by the physics and parameter space of the problem (see e.g.,[48]). Our third and final contribution, therefore, is to establish an explicit link between M_G, and hence PLMM, and AMG or MsFE preconditioners by defining appropriate prolongation and restriction matrices in Appendix B.

The paper outline is as follows: Section 2 describes the problem considered and the DNS method used to solve it. In Section 3, we detail the multiscale preconditioner through an algebraic translation of PLMM. Specifically, Section 3.3 briefly reviews PLMM, Section 3.4 shows how M_G is built from PLMM, and Section 3.6 details how M_G is updated for evolving-crack problems. Section 4 shows the 2D/3D microstructures and crack patterns used to validate the preconditioner against DNS. In Section 5, we present results for the first-pass solution obtained from a single application of M_G, and the convergence rate of preconditioned GMRES. In Section 6, we discuss implications, current gaps, and perform a complexity analysis. Section 7 concludes the paper.

2. Problem description

Consider a porous domain Ω comprised of a solid phase Ω_s and a void space Ω_v, as shown in Fig.1a (i.e., $\Omega = \Omega_s \cup \Omega_v$). Let Ω^o_s be the interior of Ω_s and $\partial\Omega_s$ its boundary. The void-solid interface is $\Gamma^w = \partial\Omega_s \cap \partial\Omega_v$. The open set Ω^o_s may include fractures Γ^F, as depicted by the black lines in Fig.1a. In this work, we represent Γ^F using a continuous damage variable defined on Ω^o_s. The solid boundary $\partial\Omega_s$ consists of Γ^w and an external boundary (or bounding box) of the domain, Γ^{ex} (i.e., $\partial\Omega_s = \Gamma^w \cup \Gamma^{ex}$). Our goal is to solve the linear-elastic momentum equation:

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{f} = 0 \tag{1a}$$

$$\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon} \tag{1b}$$

on Ω^o_s, subject to the boundary conditions (BCs):

$$\boldsymbol{u}|_{\Gamma^D} = \boldsymbol{u}_D \tag{2a}$$

$$\boldsymbol{\sigma} \cdot \boldsymbol{n}|_{\Gamma^N} = \boldsymbol{t}_N \tag{2b}$$

$$\boldsymbol{u} \cdot \boldsymbol{n}|_{\Gamma^R} = 0 \qquad \boldsymbol{m} \cdot \boldsymbol{\sigma} \cdot \boldsymbol{n}|_{\Gamma^R} = 0 \tag{2c}$$

on $\partial\Omega_s$. In Eq.2, Γ^D, Γ^N, and Γ^R are subsets of $\partial\Omega_s$ where Dirichlet, Neumann, and Roller BCs are imposed, respectively. In Eqs.1-2, $\boldsymbol{\sigma}$ is the stress tensor, $\boldsymbol{\varepsilon}$ the strain tensor, $\mathbb{C}$ the fourth-order stiffness tensor, $\boldsymbol{u}$ the displacement vector, and $\boldsymbol{f}$ the body force. Strain depends on displacement via $\boldsymbol{\varepsilon} = \nabla^s \boldsymbol{u} = ½(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)$, where ∇^s is the symmetric gradient operator and superscript T is the transpose operator. The vector $\boldsymbol{n}$ is the unit normal, and $\boldsymbol{m}$ the unit tangent on any given boundary. Notice, in 2D, $\boldsymbol{m}$ is a single vector, but in 3D, it consists of two orthonormal vectors $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$. The data $\boldsymbol{u}_D$ and $\boldsymbol{t}_N$ are known values of displacement and traction on Γ^D and Γ^N, respectively. In this work, we adopt the following notation: bold symbols denote vectors/tensors, non-bold symbols denote scalars, Latin alphabet is used for vectors, and Greek alphabet for tensors. In discussing matrix-vector algebra later, upper-case Latin is used for matrices, and lower-case Latin for vectors. We present all governing equations using 2D notation due to its simplicity (e.g., $\boldsymbol{m}$ instead of $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$). Extending the equations to 3D is straightforward.

To represent fractures Γ^F in Ω^o_s, we use a damage (or phase-field) variable ψ defined on Ω^o_s. If a point $\boldsymbol{x}$ is fully damaged (i.e., cracked), $\psi(\boldsymbol{x}) = 1$, and if it is intact (i.e., not cracked), $\psi(\boldsymbol{x}) = 0$. Given Γ^F, which is a lower-dimensional manifold representing sharp cracks, we use the approach by Borden et al.[49] to compute the corresponding ψ as follows:

$$\psi(\boldsymbol{x}) = \begin{cases} \dfrac{BG_c}{2l}\left(1 - \dfrac{\text{dist}(\boldsymbol{x}, \Gamma^F)}{l/2}\right), & \text{dist}(\boldsymbol{x}, \Gamma^F) < l/2 \\ 0, & \text{dist}(\boldsymbol{x}, \Gamma^F) \geq l/2 \end{cases} \tag{3}$$

where G_c is the fracture toughness, l a length scale that controls the thickness of the diffuse cracks, $\text{dist}(x,\Gamma^F)$ the distance of a point $\boldsymbol{x}$ from Γ^F, and B a constant taken here to be 10^3. The impact of ψ on the stiffness tensor $\mathbb{C}$ is captured via:

$$\mathbb{C} = g(\psi)\,\mathbb{C}_0 \tag{4}$$

where $g(\psi) = (1 - \psi)^2$ is the degradation function, and $\mathbb{C}_0$ the stiffness of the intact solid. Here, we assume $\mathbb{C}_0$ is isotropic. Eq.4 describes the isotropic degradation of Ω_s, which is valid under tensile loading only. More sophisticated expressions for $g(\psi)$,[50] with anisotropic degradation of $\mathbb{C}_0$ along

specific directions (i.e., "stress splits"),[18,19] also exist and may be considered. The multiscale preconditioner presented later is not affected by the particular choice of $g(\psi)$, or the way $\mathbb{C}_0$ is degraded. However, if an anisotropic degradation model of $\mathbb{C}_0$ is chosen, Eq.1 becomes nonlinear,[19] and our preconditioner must be applied to the linearized form of Eq.1.

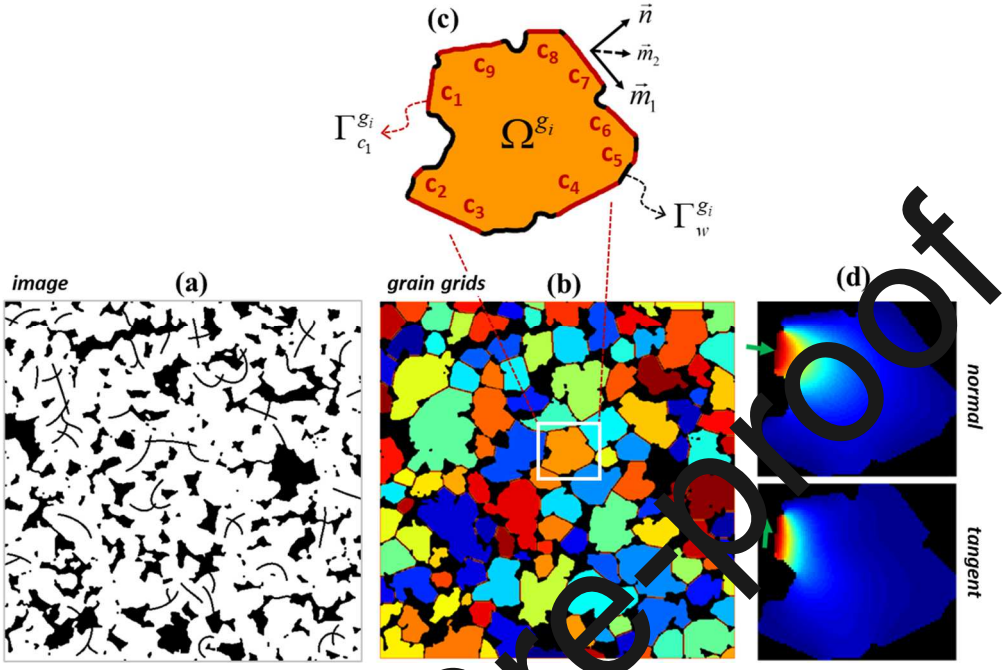


Fig 1. Schematic of the domain decomposition and basis construction. (a) Image of a fractured porous domain (Ω_s: white, Ω_v: black, Γ^F: black lines). (b) Decomposition into grain grids (randomly colored). (c) Zoom-in of a grain grid, Ω^{gi}, sharing nine contact interfaces (red) with neighboring grain grids (Γ^{gi}_{cj}). The index set of these interfaces is $C^{gi} = \{c_1, c_2, \ldots, c_9\}$. Black lines represent the void-solid interface (Γ^{gi}_{w}). (d) Two basis functions associated with Γ^{gi}_{c1} are shown. Some cracks in (a) intersect the contact interfaces, while others are confined to the interior of grain grids.

The benefit of introducing the damage variable ψ to replace Γ^F is that $\boldsymbol{u}$ becomes continuous. Moreover, the propagation of cracks due to incremental loading of Ω_s is easily captured by solving an equation for the evolution of ψ. In phase-field models, this takes the form:

$$l^2\Delta\psi + \psi = 2lG_c^{-1}(1-\psi)H^+ \tag{5}$$

where H^+ is called the "history variable"[18] and is equal to the maximum tensile energy experienced at each point in the solid since the beginning of loading. Once ψ is updated via Eq.5, Eq.1 can be solved again to obtain $\boldsymbol{u}$ at the next loading step. The result is a sequential scheme. In this work, we develop a preconditioner for the linear(ized) system arising from discretizing Eq.1 only:

$$\hat{A}\hat{x} = \hat{b} \tag{6}$$

where $\hat{A}$ is the coefficient matrix, $\hat{b}$ is the right-hand side (RHS) vector, and $\hat{x}$ the unknown vector of displacements. While we do not solve Eq.5 (i.e., cracks assumed to be static), the implications of having to repeatedly update ψ in evolving-crack problems on the preconditioner is considered. At the pore scale, Ω_s is often represented by a segmented image acquired from an X-ray μCT scanner (Fig.1a). We refer to the Cartesian mesh over which Eq.1 is discretized, to obtain Eq.6, as

the *fine grid*, whose size is an integer fraction of the image pixels. The discretization is performed using FEM with bilinear shape functions as our DNS solver. We note, however, that the multiscale preconditioner is neither limited to FEM nor Cartesian fine grids, as other DNS solvers (e.g., FVM) and arbitrarily complex unstructured grids are equally applicable.

3. Multiscale preconditioner

In the following sections, we present a two-stage preconditioner for accelerating the solution of Eq.6 in linear solvers. The preconditioner is an algebraic reinterpretation of the pore-level multiscale method (PLMM),[23] a rough outline of which was sketched in an appendix therein for *intact* solids discretized via FVM. Here, we extend this to fractured solids using FEM as the DNS solver. Section 3.1 reviews the domain decomposition that is central to the preconditioner. We then briefly review PLMM in Section 3.3 to provide some intuition for the algebraic formulation presented later. In Sections 3.4-6, we discuss how the global preconditioner M_G is built, how M_G is coupled to the smoother M_L, and how M_G is updated adaptively in evolving-crack simulations. In Appendix B, we formulate prolongation and restriction matrices associated with M_G.

3.1. Domain decomposition

Fig.1b illustrates the decomposition of Ω_s into randomly colored subdomains, or *grain grids*, denoted by Ω^{gi}. The decomposition algorithm was originally proposed by Mehmani et al.[23] and is based on the watershed transform, a morphological operation in image analysis,[15,28] of the domain image. The key difference between the decomposition here and that in Mehmani et al.[23] is that the interfaces between grain grids, called *contact interfaces* and denoted by Γ^{cj}, are not made to conform to Γ^F here. Concretely, the watershed transform is applied to Ω_s while neglecting Γ^F, which allows contact interfaces to intersect the cracks. The result is a partitioning of fine grids into those that belong to Ω^{gi} versus Γ^{cj}. Each grain grid, Ω^{gi}, and interface, Γ^{cj}, is therefore an aggregate of many fine grids. In particular, Γ^{cj} has a finite thickness (i.e., roughly 1-2 fine grids). Since fine grids correspond to an integer fraction of image pixels, and FEM is used as the DNS solver, each fine grid is a rectangular *element* in 2D and a hexahedral element in 3D. The fine-scale displacement unknowns on Γ^{cj} belong to nodes located at the corners of the elements comprising Γ^{cj}. Similarly, the unknowns on Ω^{gi} correspond to nodes in the interior of Ω^{gi}. The unknowns assigned to Γ^{cj} and Ω^{gi} are therefore mutually exclusive, implying a non-overlapping partition. We remark that watershed transform cuts Ω_s across local constrictions, which depends on the porous microstructure at hand. This property is a major reason for PLMM's accuracy in prior work, [23] and that of M_G herein. While refining the decomposition is difficult without degrading accuracy, multi-level coarsening and load balancing of subdomains are possible but not pursued.

3.2. Mathematical notation

To simplify the presentation, we adopt a notation similar to Mehmani et al.[23] Let indices g_i and c_j enumerate entities associated with Ω^{gi} and Γ^{cj}, respectively. The boundary $\partial\Omega^{gi}$ consists of $\Gamma^{gi}{}_w = \partial\Omega^{gi} \cap \Gamma^w$, representing the void-solid interface, and $\Gamma^{gi}{}_{cj} = \partial\Omega^{gi} \cap \Gamma^{cj}$, representing contact interfaces shared between Ω^{gi} and its neighboring grain grids. For some Ω^{gi}, $\Gamma^{gi}{}_{ex} = \partial\Omega^{gi} \cap \Gamma^{ex}$ is non-empty and $\partial\Omega^{gi}$ also consists of a portion on Γ^{ex}. We define C^{gi} as the index set of all interfaces Γ^{cj} that are subsets of $\partial\Omega^{gi}$, and G^{cj} as the index set of all grain grids that share Γ^{cj}. Since each Γ^{cj} is shared by exactly two grain grids, G^{cj} has only two members. We use the terms *fine-scale* and *coarse-scale* to refer to entities associated with fine grids and coarse grids (i.e., Ω^{gi}), respectively.

Coarse-scale variables are marked with a superscript *o*, and fine-scale variables with superscript *f*. We use the term *local* to refer to entities associated with a single coarse grid, and *global* to refer to entities defined on the entire domain, Ω_s. The problem dimension is denoted by D (= 2 or 3).

3.3. Review of the pore-level multiscale method (PLMM)

In PLMM, Eq.1 is solved by restricting it to each grain grid Ω^{gi}:

$$\nabla\cdot\left(\mathbb{C}:\nabla^s\boldsymbol{u}^f_{g_i}\right)=\boldsymbol{f} \qquad s.t. \qquad \begin{cases}\boldsymbol{u}^f_{g_i}\cdot\boldsymbol{n}\,|_{\Gamma^{g_i}_{c_j}}=h_{g_ic_j1}(\boldsymbol{x})\\ \boldsymbol{u}^f_{g_i}\cdot\boldsymbol{m}\,|_{\Gamma^{g_i}_{c_j}}=h_{g_ic_j2}(\boldsymbol{x})\end{cases} \tag{7}$$

where $\boldsymbol{u}^f_{gi}$ denotes the fine-scale displacement on Ω^{gi}. Dirichlet BCs are imposed on Γ^{gi}_{cj} $\forall c_j \in C^{gi}$, consisting of normal, $h_{gi,cj,1}(\boldsymbol{x})$, and tangential, $h_{gi,cj,2}(\boldsymbol{x})$, displacements at each point $\boldsymbol{x}$ on Γ^{gi}_{cj}. The BCs on Γ^{gi}_{w} are stress-free, and the BCs on Γ^{gi}_{ex} (if $\neq \emptyset$) are inherited from the global BCs in Eq.2. If $h_{gi,cj,1}(\boldsymbol{x})$ and $h_{gi,cj,2}(\boldsymbol{x})$ were known, $\boldsymbol{u}^f_{gi}$ could be solved for all subdomains and the global solution assembled. But they are not known. Hence, the central idea in PLMM is to approximate $h_{gi,cj,1}(\boldsymbol{x})$ and $h_{gi,cj,2}(\boldsymbol{x})$ as scalars $u^o_{gi,cj,1}$ and $u^o_{gi,cj,2}$, referred to as *coarse-scale unknowns*. This approximation is called the *localization assumption*. We define $\boldsymbol{u}^o_{gi,ck} = [u^o_{gi,ck,1},\ u^o_{gi,ck,2}]$ as the (here 2D) vector made up of these scalars. The result is an approximated form of Eq.7:

$$\nabla\cdot\left(\mathbb{C}:\nabla^s\boldsymbol{u}^f_{g_i}\right)=\boldsymbol{f} \qquad s.t. \qquad \begin{cases}\boldsymbol{u}^f_{g_i}\cdot\boldsymbol{n}\,|_{\Gamma^{g_i}_{c_j}}=u^o_{g_ic_j1}\\ \boldsymbol{u}^f_{g_i}\cdot\boldsymbol{m}\,|_{\Gamma^{g_i}_{c_j}}=u^o_{g_ic_j2}\end{cases} \tag{8}$$

whose solution can be expressed as follows:

$$\boldsymbol{u}^f_{g_i}=\tilde{\boldsymbol{\varphi}}^f_{g_i}+\sum_{\forall c_k\in C^{g_i}}\sum_{d=1}^{D}u^o_{g_ic_kd}\boldsymbol{\varphi}^f_{g_ic_kd} \tag{9}$$

In Eq.9, $\boldsymbol{\varphi}^f_{gi,ck,d}$ are the *basis functions* and $\tilde{\boldsymbol{\varphi}}^f_{gi}$ the *correction function* on Ω^{gi}. They satisfy:

$$\nabla\cdot\left(\mathbb{C}:\nabla^s\boldsymbol{\varphi}^f_{g_ic_kd}\right)=0 \qquad s.t. \qquad \begin{cases}\boldsymbol{\varphi}^f_{g_ic_kd}\cdot\boldsymbol{n}\,|_{\Gamma^{g_i}_{c_j}}=\delta_{kj}\delta_{1d}\\ \boldsymbol{\varphi}^f_{g_ic_kd}\cdot\boldsymbol{m}\,|_{\Gamma^{g_i}_{c_j}}=\delta_{kj}\delta_{2d}\end{cases} \tag{10a}$$

$$\nabla\cdot\left(\mathbb{C}:\nabla\tilde{\boldsymbol{\varphi}}^f_{g_i}\right)=\boldsymbol{f} \qquad s.t. \qquad \begin{cases}\tilde{\boldsymbol{\varphi}}^f_{g_i}\cdot\boldsymbol{n}\,|_{\Gamma^{g_i}_{c_j}}=0\\ \tilde{\boldsymbol{\varphi}}^f_{g_i}\cdot\boldsymbol{m}\,|_{\Gamma^{g_i}_{c_j}}=0\end{cases} \tag{10b}$$

Eq.10a is referred to as the *basis problem*, and Eq.10b as the *correction problem* on Ω^{gi}. The parameters δ_{ab} in the RHS of Eq.10a are the Kronecker delta. If $d = 1$, $\boldsymbol{\varphi}^f_{gi,ck,d}$ corresponds to setting the normal displacement at Γ^{gi}_{ck} to one, and all other displacement components at Γ^{gi}_{ck} and other contacts, Γ^{gi}_{cj} $\forall j \neq k \in C^{gi}$, to zero. If $d = 2$, only the tangential displacement at Γ^{gi}_{ck} is set to one, while the remaining displacements at all contacts are set to zero. Fig.1d depicts two such basis functions for the grain grid in Fig.1c. Note that Eq.10a entails solving $\#C^{gi}\times D$ local problems on Ω^{gi}, one per interface per normal/tangential BC; the symbol # denotes the number of members in a set. Eq.10b, by contrast, entails solving only one local problem on Ω^{gi}. In both Eqs.10a and 10b,

the BCs on Γ^{gi}_{w} are stress-free. If $\Gamma^{gi}_{ex} \neq \emptyset$, some of the global BCs from Eq.2 are inherited by Eq.10a-b. However, the RHS of these BCs are set to zero (homogeneous) for Eq.10a, but not for Eq.10b (inhomogeneous). Note the basis problem accounts for inhomogeneities due to the BCs in Eq.8 on Γ^{gi}_{cj}, and the correction problem for the inhomogeneity caused by the body force $\boldsymbol{f}$.

The final step in PLMM is to compute $u^o_{gi,ck,d}$ in Eq.9, for which a global problem must be formulated. Here, the global problem consists of continuity of momentum and displacement:

$$\boldsymbol{t}^o_{g_1c_j} = \boldsymbol{t}^o_{g_2c_j} \tag{11a}$$

$$\boldsymbol{u}^o_{g_1c_j} = \boldsymbol{u}^o_{g_2c_j} \tag{11b}$$

imposed on all Γ^{cj}. Subscripts g_1 and g_2 denote the two grain grids that share Γ^{cj}. Eq.11a equates the integrated tractions computed on either side of Γ^{cj}. Given the diffuse representation of cracks in Section 2, the continuity of displacements in Eq.11b is consistent. However, even if a sharp-crack representation were adopted, Eq.11b would still result in a convergent (but not consistent) global preconditioner. Substituting Eq.9 into Eq.11a-b, and following steps similar to Mehmani et al.[23] (not repeated), we obtain a small, $(N^o_cD)\times(N^o_cD)$ system in terms of $u^o_{i,ck,d}$, where N^o_c denotes the number of contact interfaces. This concludes our review of PLMM.

Remark 1. To correct errors caused by the localization assumption, Mehmani et al.[23] introduced a second set of coarse grids, called *contact grids*. These did not cover all of Ω_s, only a small region around Γ^{cj}, and enabled very fast convergence to the DNS solution. However, in low-porosity media, contact grids tend to overlap and must be merged. This causes some contact grids to span large regions in Ω_s, rendering local problems defined on them costly. The preconditioner of Section 3.5 addresses this drawback by obviating the need for contact grids altogether.

Remark 2. If Ω^{gi} is intact, the stiffness tensor $\mathbb{C}$ in Eq.10a corresponds to $\mathbb{C}_0$, and PLMM reduces to the formulation of Mehmani et al.[23] But if Ω^{gi} is cracked, two choices exist for $\mathbb{C}$ in Eq.10a: (1) the damaged stiffness in Eq.4 ($= g(\phi)\mathbb{C}_0$); or (2) the intact stiffness $\mathbb{C}_0$. The former is consistent and more accurate, but computationally costly for evolving-crack problems. This is because if Γ^F evolves during loading, so does $g(\phi)$ and the basis functions must be updated. Since Eq.10a entails multiple solves on Ω^{gi} per update, the cost can be prohibitive. The second choice of $\mathbb{C}_0$ does not require updating the basis functions but is inconsistent unless correction functions are computed iteratively. We explore this approach in a separate paper,[24] where we show convergence is slower. For the preconditioner discussed next, we outline an optimal way of updating the bases.

3.4. Building the global preconditioner

Building the global preconditioner, M_G, is equivalent to solving the global fine-scale system in Eq.6 *approximately*. We proceed as follows: Given the decomposition of Section 3.1, the fine-scale displacement unknowns are partitioned into those that belong to Ω^{gi} versus Γ^{cj}. We then construct a permutation matrix, W, such that when applied to Eq.6, yields the equivalent system:

$$\hat{\mathrm{A}}\hat{x} = \hat{b} \;\Rightarrow\; \underbrace{\mathrm{W}^T\hat{\mathrm{A}}\mathrm{W}}_{\mathrm{A}}\underbrace{\mathrm{W}^T\hat{x}}_{x} = \underbrace{\mathrm{W}^T\hat{b}}_{b} \;\Rightarrow\; \mathrm{A}x = b \tag{12}$$

with the following block structure:

$$\mathrm{A} = \begin{bmatrix} \mathrm{A}^g_g & \mathrm{A}^g_c \\ \mathrm{A}^c_g & \mathrm{A}^c_c \end{bmatrix} \qquad b = \begin{bmatrix} b^g \\ b^c \end{bmatrix} \qquad x = \begin{bmatrix} x^g \\ x^c \end{bmatrix} \tag{13}$$

where

$$\mathrm{A}^{g}_{g}=\begin{bmatrix}\mathrm{A}^{g_1}_{g_1} & \cdots & \mathrm{O}\\ \vdots & \ddots & \vdots\\ \mathrm{O} & \cdots & \mathrm{A}^{g_{Ng}}_{g_{Ng}}\end{bmatrix}_{(N^f_g D)\times(N^f_g D)} \qquad \begin{aligned}\mathrm{A}^{c}_{g}&=[\mathrm{A}^{c_i}_{g_j}]_{(N^f_c D)\times(N^f_g D)}\\ \mathrm{A}^{g}_{c}&=[\mathrm{A}^{g_i}_{c_j}]_{(N^f_g D)\times(N^f_c D)}\\ \mathrm{A}^{c}_{c}&=[\mathrm{A}^{c_i}_{c_j}]_{(N^f_c D)\times(N^f_c D)}\end{aligned} \tag{14}$$

The matrix W is unitary ($\mathrm{WW^T = I}$) and has 0 or 1 entries. In Eqs.12-14, N^f_{gi} and N^f_{cj} are the number of unknowns associated with Ω^{gi} and Γ^{cj}, respectively, and $N^f_g = \Sigma_i N^f_{gi}$ and $N^f_c = \Sigma_j N^f_{cj}$. The symbol N^g denotes the total number of grain grids. The permuted matrix A consists of blocks with super/subscripts g_i and c_j that correspond to Ω^{gi} and Γ^{cj}, respectively. $\mathrm{A^g_g}$ is square and block-diagonal, with $\mathrm{A^{gi}_{gi}}$ representing the stiffness matrix on Ω^{gi}. In contrast, $\mathrm{A^g_c}$ and $\mathrm{A^c_g}$ are thin and rectangular. Because A is symmetric, a result of our FEM discretization of Eq.1 with bilinear test/trial functions, $\mathrm{A^g_c} = (\mathrm{A^c_g})^T$ and $\mathrm{A^{gi}_{cj}} = (\mathrm{A^{cj}_{gi}})^T$.

We next transform $\mathrm{A}x = \mathrm{b}$ using a *reduction matrix*[1] Q that embeds the localization assumption in Eq.8. The transformation is as follows:

$$\mathrm{A}x=b,\quad x\simeq \mathrm{Q}x_M \quad\Rightarrow\quad \mathrm{Q^TAQ}x_M=\mathrm{Q^T}b \quad\Rightarrow\quad \mathrm{A_M}x_M=b_M \tag{15}$$

where

$$\mathrm{Q}=\begin{bmatrix}\mathrm{I}_{(N^f_g D)\times(N^f_g D)} & \mathrm{O}\\ \mathrm{O} & \mathrm{Q^o}\end{bmatrix} \qquad \mathrm{Q^o}=\begin{bmatrix}\mathbf{1}^{c_1} & & \mathrm{O}\\ & \ddots & \\ \mathrm{O} & & \mathbf{1}^{c_{Nc}}\end{bmatrix}_{(N^f_c D)\times(N^o_c D)} \qquad \mathbf{1}^{c_i}=\begin{bmatrix}\mathrm{I}_{D\times D}\\ \vdots\\ \mathrm{I}_{D\times D}\end{bmatrix}_{(N^f_{c_i} D)\times D} \tag{16}$$

Notice Q is block-diagonal and has 0 or 1 entries. The off-diagonal blocks are zero and the (1,1)-block is an identity matrix of size $(N^f_g D)\times(N^f_g D)$. The (2,2)-block, $\mathrm{Q^o}$, is itself block diagonal and consists of sub-matrices $\mathbf{1}^{ci}$ on its (i,i)-blocks. The matrix $\mathbf{1}^{ci}$ consist of N^f_{ci} identity matrices of size $D\times D$ (i.e., $\mathrm{I}_{D\times D}$) concatenated vertically.[2] Notice $\mathrm{Q^o}$ is $(N^f_c D)\times(N^o_c D)$, where N^o_c is the total number of contact interfaces. It is clear that constructing Q is trivial. The reduced system $\mathrm{A_M}\,x_M = b_M$, with $\mathrm{A_M} = \mathrm{Q^TAQ}$ and $b_M = \mathrm{Q^T}b$, has now the following block structure:

$$\mathrm{A}_M=\begin{bmatrix}\mathrm{A}^{g}_{g} & \overline{\mathrm{A}}^{g}_{c}\\ \overline{\mathrm{A}}^{c}_{g} & \overline{\mathrm{A}}^{c}_{c}\end{bmatrix} \qquad b_M=\begin{bmatrix}b^g\\ \overline{b}^c\end{bmatrix} \qquad x_M=\begin{bmatrix}x^g\\ x^o\end{bmatrix} \tag{17}$$

where x^o corresponds to the coarse-scale unknowns in PLMM (i.e., $u^o_{gi,ck,d}$ in Eq.9).

To interpret Eqs.15-16, note that left-multiplying a $(N^f_g + N^f_c)D\times X$ matrix by $\mathrm{Q^T}$, where X is an arbitrary integer, sums all rows associated with the fine-scale unknowns defined on Γ^{cj} along each coordinate direction separately. The result is a $(N^f_g + N^o_c)D\times X$ matrix. In particular, $\mathrm{Q^TA}$ yields the "integrated" momentum balance equations on Γ^{cj}; same as Eq.11a. Similarly, right-multiplying a $X\times(N^f_g + N^f_c)D$ matrix by Q, sums all columns associated with Γ^{cj}, resulting in a $X\times(N^f_g + N^o_c)D$ matrix. In particular, $\mathrm{Q^TAQ}$ entails that the fine-scale displacement unknowns defined on each Γ^{cj} are equal; same as the localization assumption in Eq.8. Lastly, left-multiplying a $(N^f_g + N^f_c)D\times X$

[1] This was called the "prolongation matrix" in Mehmani et al.[23] (Appendix C therein), which is a misnomer.

[2] The matrix $\mathbf{1}^{ci}$ was specified incorrectly in Mehmani et al.[23] as an all-one matrix (Appendix C therein).

matrix by Q, duplicates the reduced rows associated with Γ^{cj} over all fine grids comprising Γ^{cj}; on a per-coordinate-direction basis. In summary, the reduced system in Eq.15 consists of fine-scale unknowns in the interior of grain grids, x^g, and coarse-scale unknowns on contact interfaces, x^o. The reduced blocks of A_M and b_M are specified by overbars in Eq.17. While A_M is only slightly smaller than A, the reduction *decouples* the sub-systems associated with each Ω^{gi}. Our remaining task is to compute the approximate solution x_M. But first, we need a definition:

Definition. Let $E^{gi}{}_g$ and $R^{gi}{}_g$ be the *grain-grid* extension and restrictions matrices of Ω^{gi}, and $E^{ci}{}_c$ and $R^{ci}{}_c$ the *contact* extension and restrictions matrices of Γ^{ci}, respectively, defined as:

$$\mathrm{E}_g^{g_i} = [\Delta_{g_1}^{g_i}, \Delta_{g_2}^{g_i}, \ldots, \Delta_{g_{Ng}}^{g_i}]^{\mathrm{T}}_{(N_g^f D)\times(N_{g_i}^f D)} \qquad \mathrm{R}_g^{g_i} = \left(\mathrm{E}_g^{g_i}\right)^{\mathrm{T}} \tag{18a}$$

$$\mathrm{E}_c^{c_i} = [\Delta_{c_1}^{c_i}, \Delta_{c_2}^{c_i}, \ldots, \Delta_{c_{N_c^o}}^{c_i}]^{\mathrm{T}}_{(N_c^o D)\times D} \qquad \mathrm{R}_c^{c_i} = \left(\mathrm{E}_c^{c_i}\right)^{\mathrm{T}} \tag{18b}$$

where

$$\Delta_{g_j}^{g_i} = \begin{cases} \mathrm{I}_{(N_{g_i}^f D)\times(N_{g_i}^f D)} & if \quad j = i \\ \mathrm{O}_{(N_{g_i}^f D)\times(N_{g_j}^f D)} & if \quad j \neq i \end{cases} \qquad \Delta_{c_j}^{c_i} = \begin{cases} \mathrm{I}_{D\times D} & if \quad j = i \\ \mathrm{O}_{D\times D} & if \quad j \neq i \end{cases} \tag{18c}$$

Multiplying a $(N^f_{gi}D)\times 1$ vector on Ω^{gi} by $E^{gi}{}_g$ extends it to a $(N^f_g D)\times 1$ vector on all grain grids (i.e., $\cup_j \Omega^{gj}$). Similarly, multiplying a $D\times 1$ vector defined on Γ^{ci} by $E^{ci}{}_c$ extends it to a $(N^o_c D)\times 1$ vector on all interfaces (i.e., $\cup_j \Gamma^{cj}$). Multiplication by $R^{gi}{}_g$ or $R^{ci}{}_c$ maps in the opposite direction.

Now, to solve $A_M x_M = b_M$, we form the following Schur complement system:[22]

$$\mathrm{S}_M x^o = s_M \tag{19}$$

where

$$\mathrm{S}_M = \overline{\mathrm{A}}_c^c - \overline{\mathrm{A}}_g^c \left(\mathrm{A}_g^g\right)^{-1} \overline{\mathrm{A}}_c^g \qquad s_M \quad \overline{b}^c - \overline{\mathrm{A}}_g^c \left(\mathrm{A}_g^g\right)^{-1} b^g \tag{20}$$

by performing block-Gaussian elimination on Eq.17. Solving Eq.19 for x^o allows x^g, and thereby x_M, to be computed via:

$$x^g = \left(\mathrm{A}_g^g\right)^{-1}\left(b^g - \overline{\mathrm{A}}_c^g x^o\right) \tag{21}$$

In classical domain decomposition, it is ill-advised to construct the Schur complement matrix, S_M, explicitly as it requires inverting $A^g{}_g$. Instead, only the *action* of S_M on a vector is desired. The case here is different because S_M is very small, i.e., $(N^o{}_c D)\times(N^o{}_c D)$. We can therefore adopt the atypical strategy of actually forming, S_M, column-by-column as follows:

$$\mathrm{S}_M = \overline{\mathrm{A}}_c^c - \overline{\mathrm{A}}_g^c \mathrm{B} \tag{22}$$

where

$$\mathrm{col}_k(\mathrm{B}) = -\left(\mathrm{A}_g^g\right)^{-1} \overline{\mathrm{A}}_c^g e_k \qquad e_k = [0, \ldots, \overbrace{0,1,0}^{k-1,k,k+1}, \ldots, 0]^{\mathrm{T}}_{(N_c^o D)\times 1} \tag{23}$$

We refer to B as the *basis matrix*, whose k^{th} column is obtained by multiplying $-(A^g{}_g)^{-1}\overline{A}_c^g$ by the unit vector e_k in Eq.23. Because $A^g{}_g$, and thus its inverse, are block-diagonal, $\mathrm{col}_k(B)$ requires

solving a local sub-problem on only two grid grids. To determine which grain grids, notice the k^{th} entry in e_k (i.e., the only non-zero entry) corresponds to the interface Γ^{ci}, where:

$$c_i = \left\lceil \frac{k}{D} \right\rceil \tag{24}$$

If Γ^{ci} is shared between grain grids $\Omega^{g_{k1}}$ and $\Omega^{g_{k2}}$, then:

$$\text{col}_k(\text{B}) = -\text{E}_g^{g_{k_1}} \underbrace{\left(\text{A}_{g_{k_1}}^{g_{k_1}}\right)^{-1} \overline{\text{A}}_{c_i}^{g_{k_1}} \text{R}_c^{c_i} e_k}_{-p_k^{g_{k1}}} - \text{E}_g^{g_{k_2}} \underbrace{\left(\text{A}_{g_{k_2}}^{g_{k_2}}\right)^{-1} \overline{\text{A}}_{c_i}^{g_{k_2}} \text{R}_c^{c_i} e_k}_{-p_k^{g_{k2}}} \tag{25}$$

where the red highlighted terms require, from left to right, solving local problems on $\Omega^{g_{k1}}$ and $\Omega^{g_{k2}}$, respectively. Notice we have used here the contact restriction and grain-grid extension operators defined above. We denote the highlighted terms by $p_k^{g_{k1}}$ and $p_k^{g_{k2}}$ and refer to them as *basis vectors*. These are precisely the basis functions obtained from solving Eq. 10a in PLMM. All basis vectors, and S_M assembled therefrom, are computed once and stored to computer memory.

We next proceed to compute the RHS vector s_M in Eq.19. This is done by writing Eq.20 as:

$$s_M = \overline{b}^c - \overline{\text{A}}_g^c c^g \qquad c^g = \left(\text{A}_g^g\right)^{-1} b^g \tag{26}$$

where c^g is called the *correction vector*. Once again, because A_g^g is block-diagonal, c^g can be obtained by concatenating smaller correction vectors $c^{gi} = (\text{A}^{gi}_{gi})^{-1} b^{gi}$ defined on each Ω^{gi}. We note that c^{gi} corresponds precisely to the correction function obtained from solving Eq.10b in PLMM. If $b^{gi} = 0$, then $c^{gi} = 0$ and no computations are required to obtain the correction vector on Ω^{gi}.

Given S_M and s_M, we can now solve Eq.19 to obtain x_M. Next, we can compute x^g by substituting x^o into Eq.21. However, a careful inspection of Eq.21 reveals:

$$x^g = c^g + \text{B}x^o \tag{27}$$

which means calculating x^g does not incur any extra costs because B and c^g have already been computed. Eq.27 is the algebraic equivalent of the reconstruction step in Eq.9. Given x^g and x^o, we get x_M from Eq.17. Lastly, x_M is transformed into an approximate solution of the original system, Eq.6, by undoing the initial permutation and reduction steps via $\hat{x}_{\text{aprx}} = \text{WQ}x_M$.

The above approximate way of solving $\widehat{\text{A}}\hat{x} = \hat{b}$ can be cast as a global preconditioner, M_G. In iterative solvers, preconditioning means repeatedly solving systems like $\widehat{\text{A}}w = v$ approximately for w, given an arbitrary v. Building M_G, therefore, consists of constructing the permutation matrix W, reduction matrix Q, basis vectors $p^{gi}{}_k$, and basis matrix B once, and storing them all to computer memory. We note the computations of $p^{gi}{}_k$ and B can be fully parallelized across Ω^{gi}. In applying M_G for each new vector v, we only need to compute c^{gi}, which is also parallelizable across Ω^{gi}.

Remark 3. In Appendix B, we show that Eq.26 is equivalent to $s_M = \overline{b}^c + \text{B}^\text{T} b^g$, which seems to obviate computing c^g. However, c^g is still needed in Eq.27 to reconstruct x^g. If our goal is to merely obtain an approximate solution, then c^g must be computed once and used in Eq.27. But if our goal is to precondition $\widehat{\text{A}}\hat{x} = \hat{b}$ in an iterative solver, where $\widehat{\text{A}}w = v$ is repeatedly solved for different v, recomputing c^g can become cumbersome. In Appendix B, we show how this cost can be eliminated by reinterpreting c^{gi} as yet another basis vector that is computed once.

Remark 4. The above global preconditioner can be expressed as $\text{M}_\text{G} = \widehat{\text{R}}\widehat{\text{A}}\widehat{\text{P}}$, where $\widehat{\text{P}} = \text{WQP}$ is a prolongation matrix and $\widehat{\text{R}} = \widehat{\text{P}}^\text{T}$ a restriction matrix. Note P is the prolongation matrix associated

with the reduced system $A_M x_M = b_M$ in Eq.15, which we formulate in Appendix B. Using this multigrid notation, the approximate solution can be expressed as $\hat{x}_{aprx} = M_G^{-1}\hat{b} = \widehat{P}(\widehat{R}\widehat{A}\widehat{P})^{-1}\widehat{R}\hat{b}$. In Appendix B and Section 6.4, we discuss the relation of M_G to algebraic multigrid methods.

3.5. Combining the global preconditioner with a smoother

To apply M_G in iterative solvers, we must pair it with a local preconditioner, or *smoother*, denoted by M_L. The pairing is done as follows:[22]

$$M^{-1} = M_G^{-1} + M_L^{-1}\left(I - AM_G^{-1}\right) \tag{28}$$

yielding the multiscale preconditioner M. The rationale for Eq.28 is that M_G, by itself, attenuates only long-range, or low-frequency, errors. By contrast, M_L can efficiently remove high-frequency errors, but is slow to correct long-range errors on its own. The pairing targets both error frequencies thereby accelerating convergence. We consider smoothers that are constructed as follows:

$$M_L^{-1} = \sum_{i=1}^{n_{st}} M_l^{-1} \prod_{j=1}^{i-1}\left(I - AM_l^{-1}\right) \tag{29}$$

where M_l is an incomplete LU-factorization of $\widehat{A}$, i.e., ILU(k) with k denoting the fill level. Eq.29 entails a multi-stage application of M_l, where n_{st} is the number of smoothing stages. More details on Eqs.28-29 can be found elsewhere.[22,51] In Section 5, we test M within a GMRES solver.

3.6. Updating the global preconditioner

Suppose a global preconditioner M_G has been built for Ω_s. Now let Ω_s be subject to progressive loading, during which cracks nucleate and evolve, altering the stiffness matrix $\widehat{A}$ via $g(\psi)$ in Eq.4. How should M_G be updated? And how often? If M_G is not updated at all, the Krylov solver will require more and more iterations to converge as cracks grow with loading steps, thus increasing cost. If M_G is updated at each loading step, the cost of updates may override the time spent by the solver itself. Here, we present an efficient way of updating M_G that preserves rapid convergence.

Updating M_G means updating the basis matrix B and the Schur matrix S_M with respect to those grain grids Ω^{gi} in which new cracks have nucleated and/or old ones have grown. This entails recomputing the basis vectors p^{gi}_k on the impacted Ω^{gi}. However, just because cracks have evolved in Ω^{gi} does not mean its basis functions need updating. A more efficient strategy is to adopt the adaptivity criterion below, which is supported by numerical observations discussed later:

Criterion. If cracks, Γ^F, intersect a contact interface Γ^{ci} between two adjacent grain grids Ω^{g1} and Ω^{g2}, then update M_G by recomputing *all* the basis vectors defined on Ω^{g1} and Ω^{g2}.

This criterion ensures that basis vectors, hence M_G, are updated infrequently and only if necessary. In other words, cracks confined to the interior of grain grids are ignored. To determine whether a crack intersects Γ^{ci}, we check if $g(\psi) < 10^{-2}$ at any point on Γ^{ci}. After identifying the grain grids satisfying the criterion, denoted by the index set $\tilde{G}$, we update M_G by executing Algorithm 1. The terms with an under-tilde denote vectors/sub-matrices altered by the evolved cracks. The red highlighted terms correspond to precomputed and stored variables of the old M_G, which need not be recomputed. The outputs of Algorithm 1 consist of updated matrices B and S_M.

Algorithm 1. Updating $\mathrm{M_G}$

Inputs: Coefficient matrix $\underset{\sim}{\mathrm{A}}$ altered by evolved cracks

Outputs: Updated Schur matrix $\mathrm{S_M}$ and basis matrix B
Note 1. *S_M and B, along with W and Q, comprise M_G*
Note 2. *Matrices W and Q remain unaltered by cracks*

Do $g_i \in \underset{\sim}{G}$
  Do $j \in C_{g_i}$
    Do $d \in \{1,\ldots,D\}$

$$k = (j-1)D + d$$
$$\underset{\sim}{p}_k^{g_i} = -\left(\underset{\sim}{\mathrm{A}}_{g_i}^{g_i}\right)^{-1} \bar{\underset{\sim}{\mathrm{A}}}_{c_j}^{g_i} \mathrm{R}_c^{c_j} e_k$$
$$\mathrm{col}_k(\mathrm{B}) := \mathrm{col}_k(\mathrm{B}) + \mathrm{E}_g^{g_i}\left(\underset{\sim}{p}_k^{g_i} - p_k^{g_i}\right)$$
$$\mathrm{col}_k(\mathrm{S_M}) := \mathrm{col}_k(\mathrm{S_M}) + \mathrm{E}_c^{c_j}\left(\bar{\underset{\sim}{\mathrm{A}}}_{g_i}^{c_j} \underset{\sim}{p}_k^{g_i} - \bar{\mathrm{A}}_{g_i}^{c_j} p_k^{g_i}\right)$$

    End do
$$\mathrm{S_M} := \mathrm{S_M} + \mathrm{E}_c^{c_j} \mathrm{R}_c^{c_j} (\bar{\underset{\sim}{\mathrm{A}}}_c^c - \bar{\mathrm{A}}_c^c) \mathrm{E}_c^{c_j} \mathrm{R}_c^{c_j}$$
  End do
End do

To apply the updated $\mathrm{M_G}$ to the evolved system, we proceed similar to Section 3.4 but with a key modification: we account for *all* cracks in calculating the correction vectors c^{gi}, not just those intersecting a contact interface. Let us adopt the following notation: variables with under-tilde (e.g. $\underset{\sim}{x}$) correspond to grain grids with cracks intersecting at least one contact interface; variables with underline (e.g., $\underline{x}$) correspond to grain grids with (interior) cracks that do not intersect any contacts; and variables with no under-tilde or underline correspond to intact grain grids. For example, the sets G, $\underline{\mathrm{G}}$, and $\underset{\sim}{\mathrm{G}}$ contain the indices of grain grids satisfying the above definitions. Similarly, $c^{gi} = (\mathrm{A}^{\mathrm{gi}}{}_{\mathrm{gi}})^{-1} b^{gi}$, $\underline{c}^{gi} = (\underline{\mathrm{A}}^{\mathrm{gi}}{}_{\mathrm{gi}})^{-1} b^{gi}$, and $\underset{\sim}{c}^{gi} = (\underset{\sim}{\mathrm{A}}^{\mathrm{gi}}{}_{\mathrm{gi}})^{-1} b^{gi}$. We now assemble s_M via:

$$s_M = \bar{b}^c - \sum_{g_i \in G} \bar{\mathrm{A}}_{g_i}^c c^{g_i} - \sum_{g_i \in \underset{\sim}{G}} \bar{\mathrm{A}}_{g_i}^c \underset{\sim}{c}^{g_i} - \sum_{g_i \in \underline{G}} \underline{\mathrm{A}}_{g_i}^c \underline{c}^{g_i} \tag{30}$$

Given s_M, we solve Eq.17 to obtain x^o. To compute x^g, we have two options: (1) Ignore interior cracks in $\underline{\mathrm{G}}$ and use Eq.27, which requires no extra computations. This means x^g is reconstructed using basis vectors that assume the grain grids in $\underline{\mathrm{G}}$ are intact; (2) Include cracks in $\underline{\mathrm{G}}$ in exchange for solving one extra subsystem per grain grid in $\underline{\mathrm{G}}$; outlined in Appendix A. This compensates for the fact that basis vectors defined on the grain grids in $\underline{\mathrm{G}}$ assume the solid is locally intact.

Remark 5. Numerical experiments revealed that both options for computing x^g lead to identical convergence rates in Krylov solvers. This is because fine-scale details of the solution around cracks associated with $\underline{\mathrm{G}}$, even if missed during the reconstruction of x^g via Eq.27, are captured by the smoother, $\mathrm{M_L}$ in Section 3.5. On the other hand, including such details is equivalent to having a block-Jacobi smoother built into $\mathrm{M_G}$, lessening the burden on $\mathrm{M_L}$ chosen in Section 3.5. In Section 5, we adopt the latter option of using Eq.A.2 to reconstruct x^g, because it is also useful for obtaining accurate first-pass solutions for comparison against DNS.

Remark 6. A single application of the global preconditioner, $\mathrm{M_G}$, to Eq.6 yields an approximate, or first-pass, solution, i.e., $\hat{x}_{\mathrm{aprx.}} = \mathrm{M_G}^{-1}\hat{b}$. The quality of this solution is a direct reflection of the ability of $\mathrm{M_G}$ to attenuate low-frequency errors. If $\mathrm{M_G}$ is built from scratch for a fractured domain,

following the procedure outlined in Section 3.4, we refer to the first-pass solution as A_0. The letter "A" indicates *all* cracks are included in building the basis vectors (hence B and S_M) of M_G. But if M_G is obtained by updating a previously built global preconditioner via Algorithm 1, we refer to the first-pass solution as U_0. The letter "U" indicates that M_G has been *updated*.

4. Problem set

We test the multiscale preconditioner for the domains in Fig.2. They include a 2D disk pack (P2D),[23] a 2D sandstone (S2D),[52] and a 3D sandstone (S3D).[52] The stiffness tensor $\mathbb{C}_0$ is isotropic with Lamé parameters $\lambda = 8.3$ GPa and $\mu = 44.3$ GPa corresponding to α-quartz.[53] In all domains, the boundary corresponding to $x = L_x$ is fixed and a constant displacement of $u_D = 1$ is imposed on $x = 0$ along the negative x-direction (i.e., tensile); where L_x is the domain size in the x-direction. All lateral boundaries satisfy roller BCs. The void-solid interface Γ^w is stress-free, implying zero pore pressure. The fine grids correspond to the image pixels in Fig.2, which have dimensions 716 × 576 in P2D, 541 × 546 in S2D, and 67 × 68 ×71 in S3D. Fig.2 also shows the decomposition of each domain into grain grids. Their number, N^g, is 76 in P2D, 120 in S2D, and 47 in S3D.

For each domain, we consider different crack patterns shown in Fig.3. For P2D and S2D, we consider three patterns: (1) c_1: all cracks are confined to the interior of grain grids (i.e., none intersects a contact interface); (2) c_2: all cracks intersect at least one contact interface; and (3) c_3: cracks are obtained by superposing the cracks from patterns c_1 and c_2. For S3D, we consider two crack patterns c_{50} and c_{100}, which correspond respectively to placing 50 and 100 elliptical cracks with different sizes and orientations randomly inside the domain. For brevity, we use P2D-c_1, P2D-c_2, P2D-c_3, S2D-c_1, S2D-c_2, S2D-c_3, S3D-c_{50}, and S3D-c_{100} to refer to each domain-pattern pair. In some cases, where we consider domains to be intact (i.e., containing no cracks), we use P2D-c_0, S2D-c_0, and S3D-c_0. The phase-field length-scale parameter l and fracture toughness G_c in Eq.3 are set to $l = 2h = 0.02$ mm and $G_c = 2.7\times10^{-3}$ GPa-mm, respectively, in all cases.

5. Results

In the following sections, we first compare the first-pass solutions A_0 and U_0 obtained from a single application of M_G against the exact DNS solution from a direct solver. Section 5.1.1 presents results for intact domains, and Section 5.1.2 for fractured domains. The intact-domain results are discussed in relation to the FFMM results of Mehmani et al.[23] We measure the errors of A_0 and U_0 via:

$$E_p^{\boldsymbol{u}} = \frac{\|\boldsymbol{u}_M - \boldsymbol{u}_S\|}{\sup_s \|\boldsymbol{u}_S\|}\times 100 \qquad E_2^{\boldsymbol{u}} = \left(\frac{1}{|\Omega_s|}\int_{\Omega_s}(E_p^{\boldsymbol{u}})^2\,\mathrm{d}\Omega\right)^{1/2} \tag{31}$$

where $E_p^{\boldsymbol{u}}$ and $E_2^{\boldsymbol{u}}$ quantify the pointwise and L_2 errors of the displacement field, respectively, both expressed in percentages.[23] In Section 5.2, we analyze the convergence rate of GMRES when right preconditioned by the multiscale preconditioner M in Eq.28, made up of M_G and M_L. For M_L, we consider ILU(k) with $k = 0, 1, 2$ as M_l, and $n_{st} = 1, 6, 12$ in Eq.29. For M_G, we consider two options: (1) M_G is built from scratch, including all cracks in the basis vectors, using the procedure outlined in Section 3.4; and (2) M_G is updated from a previously built M_G for the intact domain. We use the simple, but abusive, notation of A_0 to refer to the former M_G, and U_0 to refer to the latter M_G.

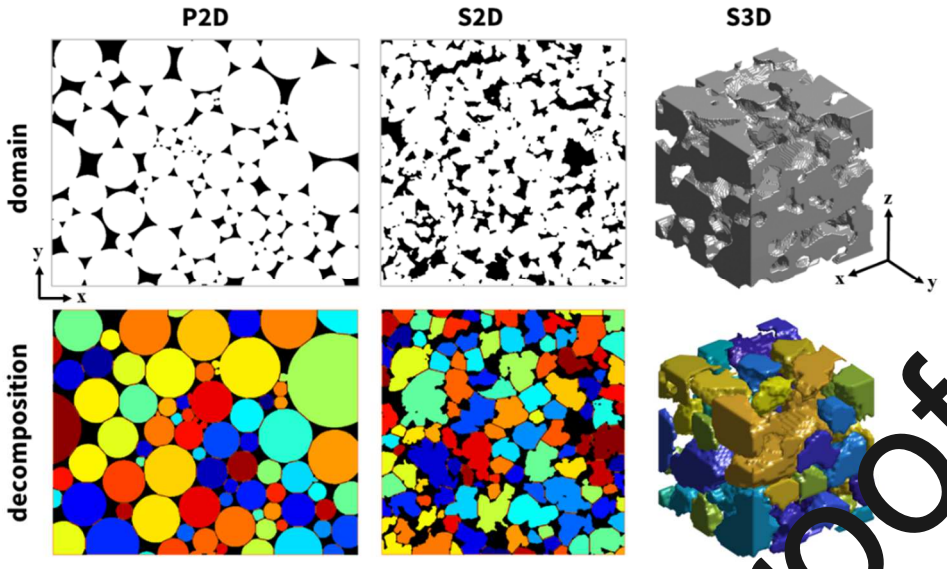


Fig 2. (Top row) Pore-scale domains used to test the multiscale preconditioner. (Bottom row) The decomposition of these domains into grain grids (randomly colored).

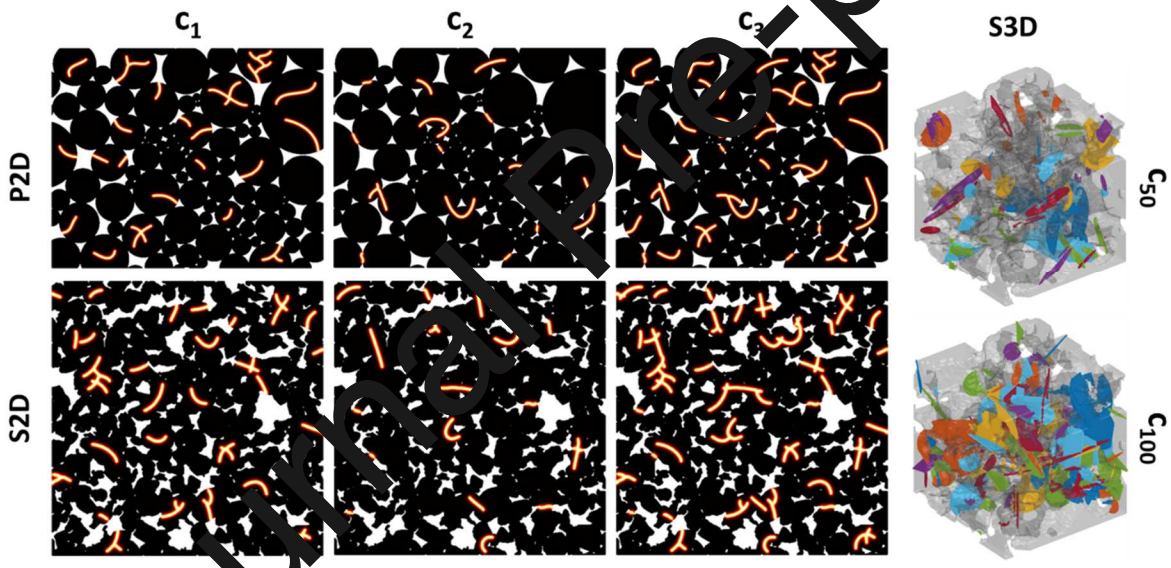


Fig 3. Crack patterns used to test the multiscale preconditioner. In P2D and S2D, c_1 denotes cracks confined solely to the interior of grain grids, c_2 denotes cracks intersecting at least one contact interface, and c_3 is a superposition of cracks in patterns c_1 and c_2. In S3D, c_{50} and c_{100} correspond to 50 and 100 elliptical cracks, respectively, with different sizes and orientations placed randomly in the domain.

5.1. Single application of the global preconditioner

5.1.1. Intact domains

Figs.4-5 compare the first-pass solutions obtained from a single application of M_G against DNS for P2D-c_0, S2D-c_0, and S3D-c_0. Because these domains are intact, the first-pass solutions A_0 and U_0 are equal, and only A_0 is plotted. Figs.4-5 show that the displacement magnitude $|\boldsymbol{u}|$, maximum shear stress, $\boldsymbol{\sigma}_m$, and volumetric strain, $\nabla\cdot\boldsymbol{u}$ of A_0 are in excellent agreement with DNS. Table 1

summarizes the relative L_2-errors (%) of the displacement field for A_0 (see the "c_0" columns). In all domains, $E^{u}_2 < 5\%$ which is comparable to the PLMM errors reported in Mehmani et al.[23] for the same intact domains. The small numerical discrepancy between the two is likely due to the fine-grid discretization method, i.e., FEM here but FVM there. The high accuracy of A_0 implies that it can serve as an attractive global preconditioner for intact domains.

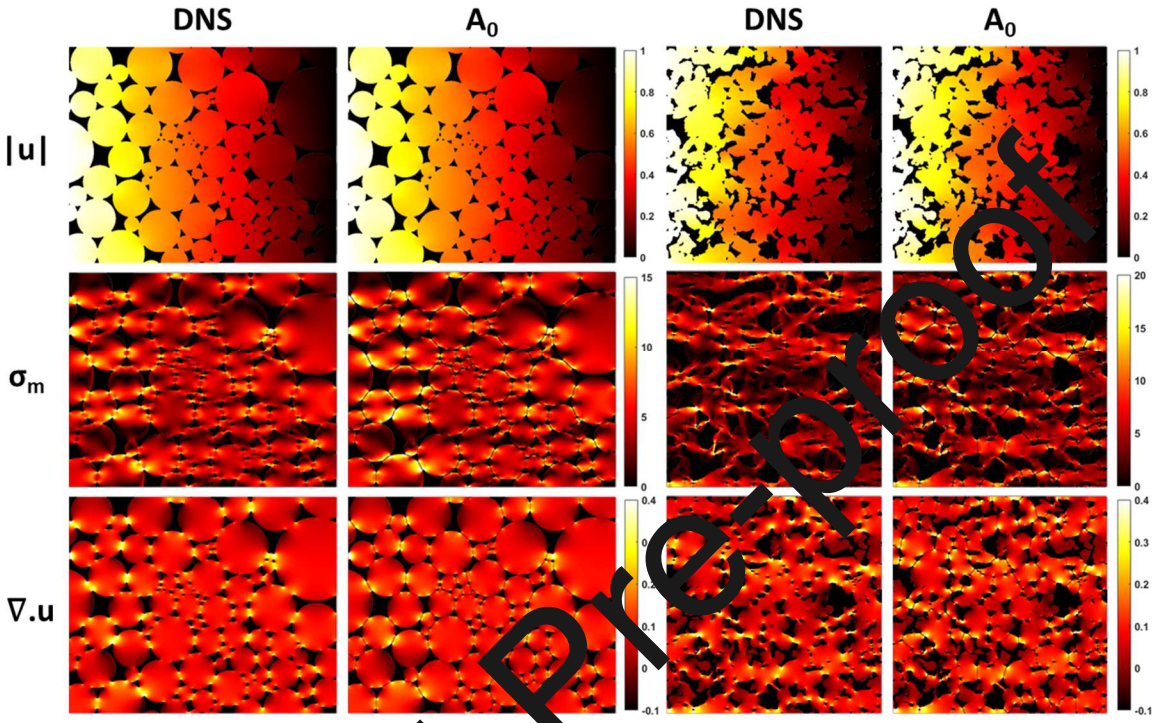


Fig 4. Single application of the global preconditioner A_0 against DNS for P2D-c_0 and S2D-c_0.

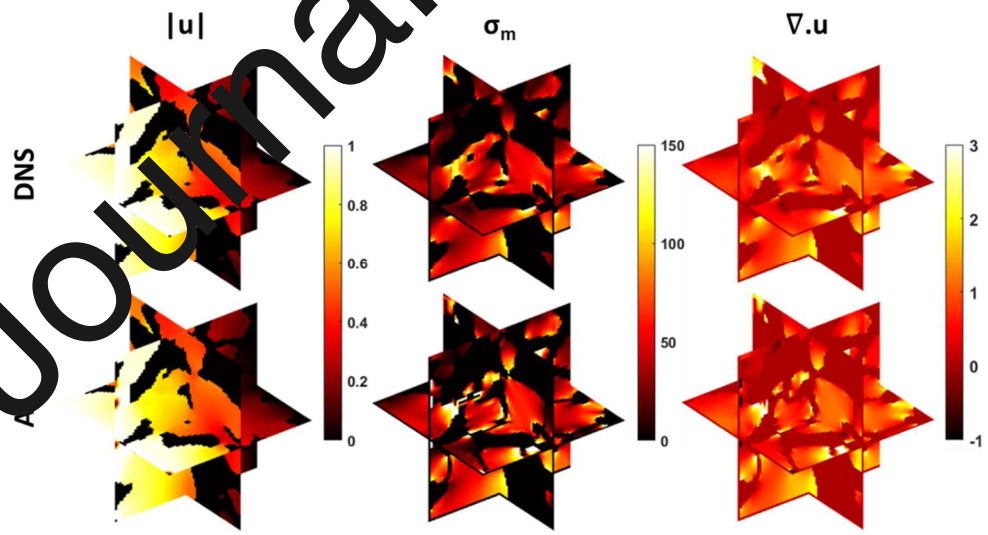


Fig 5. Single application of the global preconditioner A_0 against DNS for S3D-c_0.

Table 1. Relative L_2-errors (%) of displacement (E^u_2) between the first-pass solutions A_0 and U_0 obtained from a single application of the global preconditioner versus the exact DNS solution.

	P2D				S2D				S3D		
	c_0	c_1	c_2	c_3	c_0	c_1	c_2	c_3	c_0	c_{50}	c_{100}
A_0	0.46	0.64	3.07	3.58	1.8	2.34	5.42	6.04	4.57	6.54	6.05
U_0	0.46	5.88	3.18	5.11	1.8	8.61	4.99	8.49	4.57	6.54	6.05

5.1.2. Fractured domains

Similar to Section 5.1.1, Figs.6-8 compare the first-pass solutions obtained from a single application of M_G against DNS for P2D-c_3, S2D-c_3, and S3D-c_3. The results for crack patterns c_1 and c_2 are not shown as they are less challenging than c_3. Here, the approximate solutions A_0 and U_0 differ, hence, both are shown. As in Figs.6-8, we see that A_0 and U_0 are in excellent agreement with DNS. Unsurprisingly, A_0 is slightly more accurate than U_0 because it accounts for *all* cracks in the building of M_G, not just those that intersect contact interfaces. Table 1 summarizes the L_2-errors associated with A_0 and U_0, which are $E^u_2 < 6\%$ and $E^u_2 < 8.5\%$, respectively, for all domain-crack pattern pairs. Notice E^u_2 for A_0 is roughly equal to E^u_2 for U_0 in crack pattern c_2. This is expected because all cracks are included in computing U_0 through the adaptivity criterion defined in Section 3.6. By the same token, the errors of U_0 are larger than those of A_0 for pattern c_1 because no cracks are included in the construction of M_G for U_0. Despite this apparent superiority of A_0 over U_0, the next section shows they perform comparably as M_G within a GMRES solver. Figs.9-11 depict pointwise errors of A_0 and U_0, which tend to concentrate near the cracks and contact interfaces. The latter is due to the localization assumption in Eq.8. Removing these errors is the main function of the smoother M_L discussed in Section 3.5.

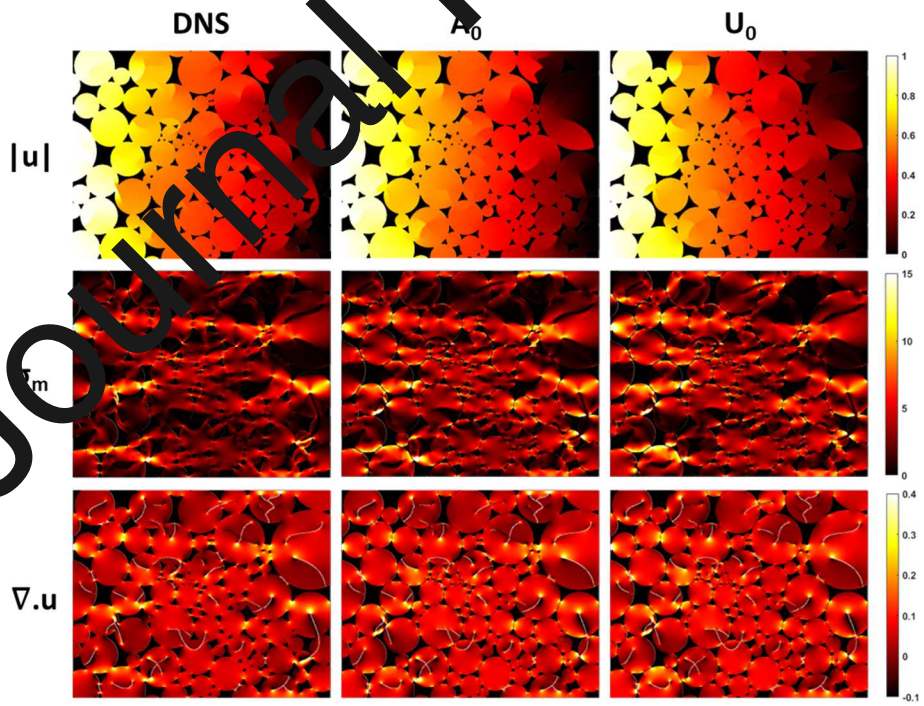


Fig 6. Single application of the global preconditioners A_0 and U_0 against DNS for P2D-c_3.

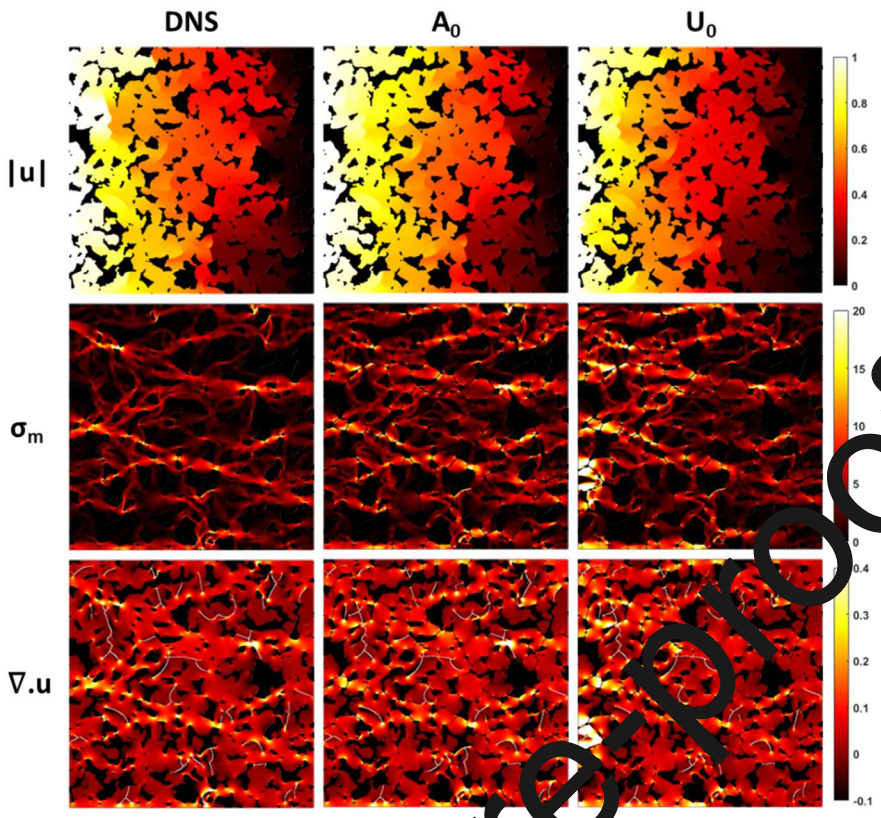


Fig 7. Single application of the global preconditioners A_0 and U_0 against DNS for S2D-c_3.

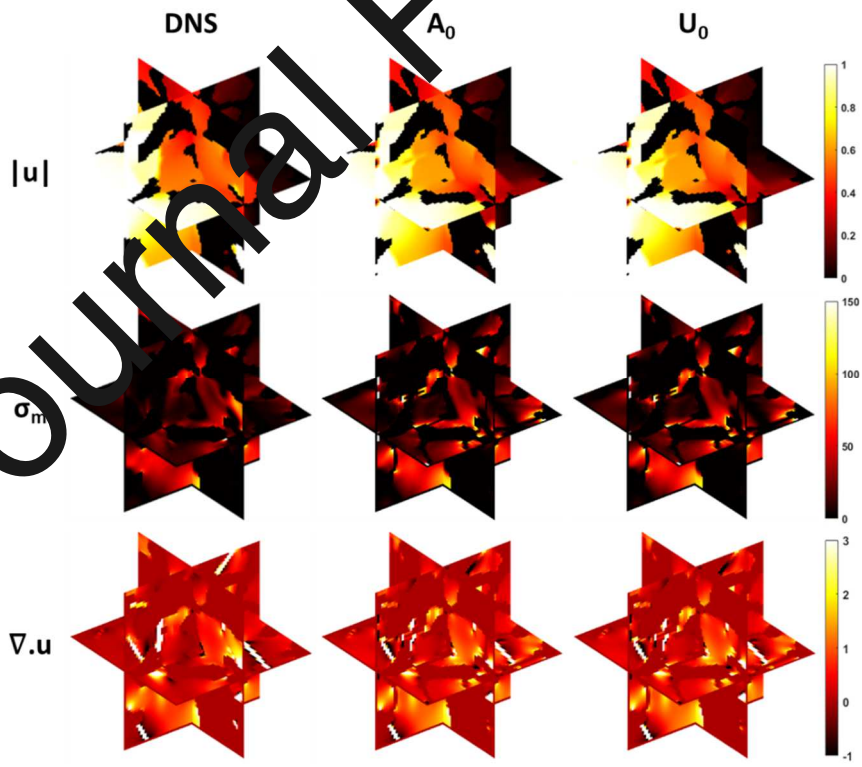


Fig 8. Single application of the global preconditioners A_0 and U_0 against DNS for S2D-c_3.

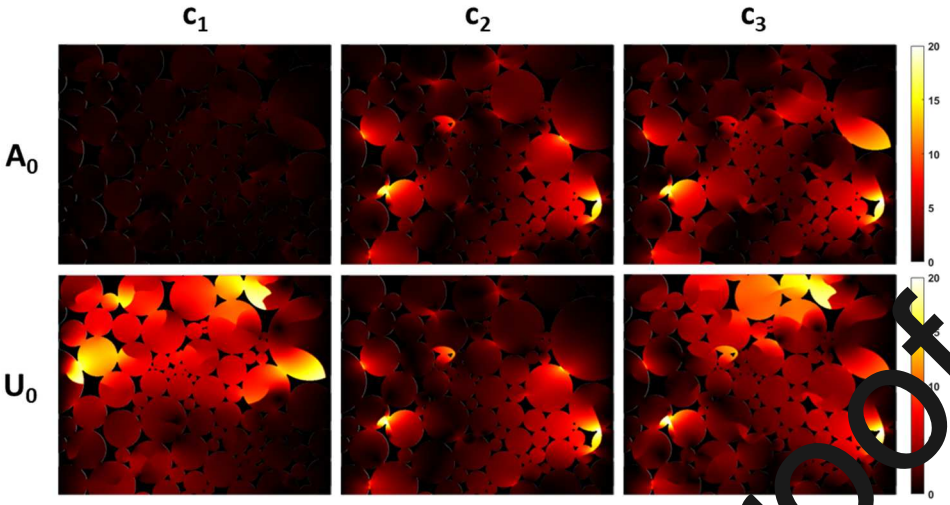


Fig 9. Pointwise displacement errors, E^{u}_{p} (%), of A_0 and U_0 for P2D-c_1, P2D-c_2, and P2D-c_3.

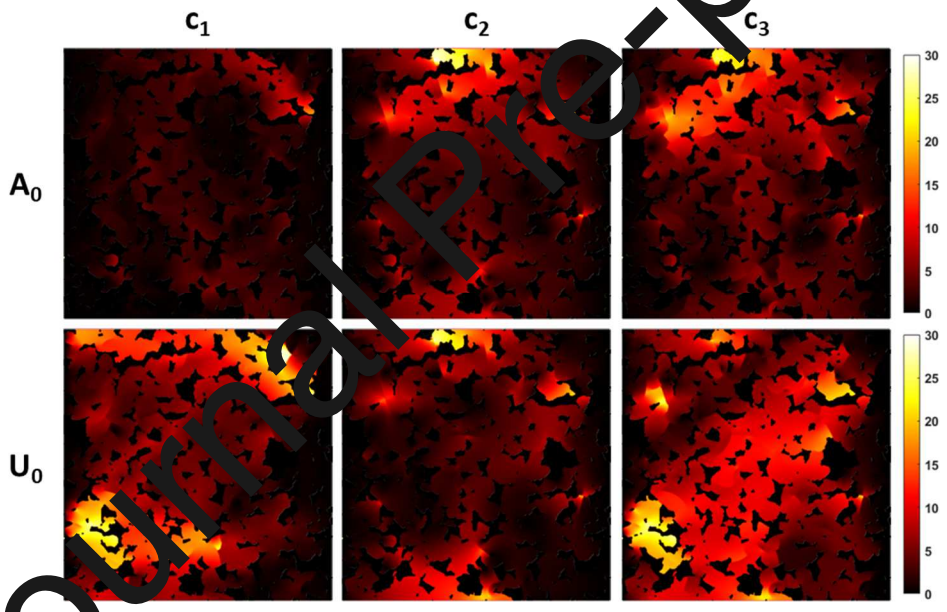


Fig 10. Pointwise displacement errors, E^{u}_{p} (%), of A_0 and U_0 for S2D-c_1, S2D-c_2, and S2D-c_3.

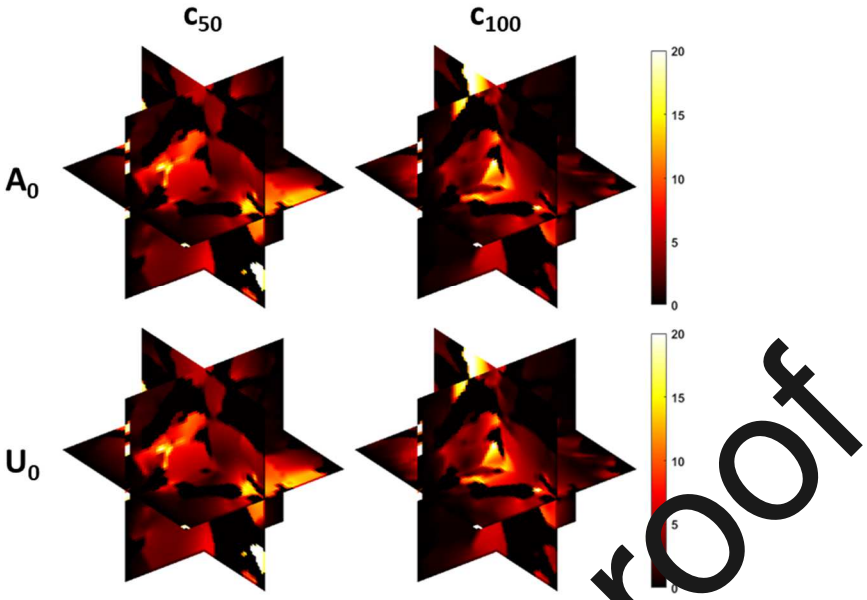


Fig 11. Pointwise displacement errors, E''_p (%), of A_0 and U_0 for S3D-c_{50} and S2D-c_{100}.

We conclude this section by noting the first-pass solutions A_0 and U_0 have one fundamental difference from early iterates produced by a generic AMG solver: the former conserve momentum whereas the latter do not. This property also applies to all later iterates A_m and U_m obtained from our global preconditioner. The implication is that A_0 and U_0, and all subsequent iterates of M_G, can be used readily in engineering applications without having to converge all the way to the exact DNS solution. The same cannot be said about AMG solvers. As an example, using an aggregation-based algebraic multigrid (AGMG) solver[54] on the S2D-c_3 domain, we obtained E''_2 = 48%, 41%, and 39% errors, respectively, for the 1st , 2nd, and 4th iterates. Not only do these not conserve momentum, they are 6-8 times less accurate than A_0 and U_0 in Table 1.

5.2. Preconditioning a Krylov solver (GMRES)

We next test the performance of the multiscale preconditioner M in Eq.28, which combines M_G and M_L in a right preconditioned GMRES solver. We probe different combinations of M_G, based on A_0 or U_0, with M_L based on Eq.29 with M_l = ILU(k), k = 0, 1, 2, and n_{st} = 1, 6, 12. We chose ILU(k) for M_L over other popular smoothers, such as Gauss-Seidel, because ILU(k) is more flexible in the way its accuracy is hierarchically controlled by the fill-in parameter k, and because preliminary tests revealed it to be more efficient; often by ×2 in CPU-time. We consider all pairings of the domains and crack patterns shown in Figs.2-3, as well as intact domains, i.e., P2D-$c_{0,1,2,3}$, S2D-$c_{0,1,2,3}$, and S3D-$c_{0,50,100}$. We declare GMRES to have *converged* if the normalized residual satisfies $||\hat{A}\hat{x} - \hat{b}||/||\hat{b}|| < 10^{-8}$ in less than 150 iterations. Otherwise, the solver is said to have "diverged." We set the restart value of GMRES equal to 20. Table 2 summarizes the number of iterations required to converge in all cases. Under S3D, "div" means "diverged". Table 3 also lists the total wall-clock time (WCT) associated with each case. This includes times spent on building M_G and M_L plus solving the linear system with GMRES. For comparison, Tables 2-3 also include the number of iterations and WCTs of an aggregation-based algebraic multigrid (AGMG) solver[54] that is known to outperform classical AMG solvers.[47] For AGMG, iterations are performed until $||\hat{A}\hat{x} - \hat{b}||/||\hat{b}|| < 10^{-8}$ without capping their number at 150. The AGMG source code was obtained

from *agmg.eu*. All cases were run in *series* and on an Intel Core(TM) i9-10980XE CPU @ 3.00GHz, 128 GB RAM machine. Each run was repeated three times to measure WCT.

Table 2. Number of GMRES iterations to converge ($||\hat{A}\hat{x} - \hat{b}||/||\hat{b}|| < 10^{-8}$) for different combinations of the global preconditioner M_G built from A_0 and U_0, smoothers with M_l = ILU(k) where k = 0, 1, 2, and number of smoothing stages is n_{st} = 1, 6, 12. All domains, cracked (c_1, c_2, c_3) and intact (c_0), are listed. Results of an aggregation-based algebraic multigrid (AGMG) solver are also included for comparison. GMRES runs marked "div" indicate iterations exceeded 150. No such limit was imposed on AGMG.

		A_0									U_0									
		ILU(0)			ILU(1)			ILU(2)			ILU(0)			ILU(1)			ILU(2)			
	n_{st}	1	6	12	1	6	12	1	6	12	1	6	12	1	6	12	1	6	12	AGMG
P2D	c_0	21	10	8	15	8	7	12	7	6	21	10	8	15	8	7	12	7	6	123
	c_1	21	11	9	16	9	7	13	8	7	22	13	12	17	12	11	15	11	1	206
	c_2	52	27	23	38	22	19	32	19	16	52	28	23	38	22	1	32	19	16	187
	c_3	59	32	27	44	26	22	37	22	19	60	32	27	45	26	22	37	22	19	228
S2D	c_0	25	13	11	18	11	10	15	10	11	25	13	11	18	11	0	15	10	11	283
	c_1	27	15	13	20	12	11	17	11	12	29	17	16	2	15	14	19	14	15	407
	c_2	44	26	22	32	21	18	30	18	17	45	26	22	32	1	18	30	18	17	501
	c_3	64	34	28	48	27	23	40	23	23	64	34	9	49	28	23	40	24	23	504
S3D	c_0	23	12	10	17	10	8	14	9	8	3	1	10	17	10	8	14	9	8	92
	c_{50}	38	21	20	29	23	25	34	2	d	3	21	20	29	23	25	34	24	div	696
	c_{100}	46	25	26	33	30	33	36	36	v	46	25	26	33	30	33	36	36	div	268
P2D-H		88	51	160	61	44	64	55	62	div	89	52	159	62	44	61	55	67	div	1315

The following key observations are made: (1) Whether M_G is built from A_0 or U_0 makes little difference in the number of GMRES iterations. This is noteworthy because it means updating M_G via Algorithm 1 is *as good as* building it from scratch following the steps in Section 3.4. Note the WCTs associated with U_0 are slightly higher than A_0 due to implementational differences in the way M_G is constructed. For U_0, M_G is first built for the intact domain, then updated to account for cracks, resulting in redundant subdomain solves that artificially inflate WCT. In practice, the WCT would only include that of updating an *existing* M_G; (2) The larger the fill-in value k is in ILU(k), and the more smoothing stages n_{st} are performed in Eq.29, the fewer iterations are required by GMRES to converge. However, there are diminishing returns in terms of WCT. For example, performance worsens in nearly all cases as n_{st} grows larger than one. The only exception is ILU(0), for which n_{st} = 6 is optimal. Another example is if we fix n_{st} and go from ILU(1) to ILU(2), WCT either stays constant or worsens (e.g., S3D). In contrast, we always see improvement when going from ILU(0) to ILU(1). We thus recommend ILU(0) with n_{st} = 6 or ILU(1) with n_{st} = 1 as the best smoothers; (3) In S3D-c_{50} and S3D-c_{100}, large values of k and n_{st} can lead to divergence, the patterns of which are illustrated in Fig.12f. It is interesting to note that the intact domain S3D-c_0 does not diverge, which may indicate that generic smoothers such as ILU(k) are not optimal for fractured porous materials. In Section 6.1, we discuss how specialized smoothers may be designed in the future that are more robust; Finally (4) if cracks are confined solely to the interior of the grain grids (pattern c_1), convergence rate is comparable to intact domains (pattern c_0). But if cracks

intersect contact interfaces (patterns c_2 and c_3), convergence is slower (by ×2) because the quality of the localization assumption in Eq.8, and thus M_G, is degraded by such cracks.

Table 3. Total wall-clock times (WCT) in seconds corresponding to the cases in Table 2. WCTs include the times spent on building M_G and M_L plus solving the linear system. All cases are run in series. AGMG results are obtained from a *compiled* FORTRAN code, whereas preconditioned GMRES results from an *uncompiled* MATLAB code. Despite the disadvantage, GMRES is on par with AGMG, and outperforms AGMG for large domain sizes such as P2D-H, depicted by Fig.13.

		A_0									U_0									
		ILU(0)			ILU(1)			ILU(2)			ILU(0)			ILU(1)			ILU(2)			
	n_{st}	1	6	12	1	6	12	1	6	12	1	6	12	1	6	12	1	6	12	AGMG
P2D	c_0	24	22	23	22	22	25	21	23	25	24	22	23	22	22	25	21	23	25	16
	c_1	23	23	25	21	22	24	20	22	25	33	32	36	31	33	37	30	33	41	31
	c_2	43	37	43	35	35	43	32	34	41	51	46	51	43	43	51	40	42	49	27
	c_3	49	43	51	39	41	50	35	39	49	63	56	64	53	53	62	48	51	61	36
S2D	c_0	15	14	15	13	14	17	13	15	19	15	14	15	13	14	17	13	15	19	22
	c_1	15	14	17	12	13	17	12	14	19	20	19	22	17	19	23	17	19	26	38
	c_2	22	20	24	17	19	23	17	18	24	26	24	28	21	22	27	21	22	28	44
	c_3	30	25	29	23	22	27	20	21	29	36	30	35	29	28	33	26	27	35	49
S3D	c_0	224	209	213	219	220	227	232	246	261	224	209	213	219	220	227	232	246	261	31
	c_{50}	233	225	248	231	253	305	260	302	div	404	390	416	397	425	493	436	477	div	278
	c_{100}	214	201	235	206	245	312	237	322	div	380	371	404	374	422	490	412	492	div	117
P2D-H		1367	1162	4933	1062	1148	2030	918	1693	div	1648	1402	5219	1241	1438	2142	1236	2059	div	2893

The solid lines in Fig.12 illustrate the convergence pattern of GMRES for a subset of the entries in Table 2. They correspond to P2D-c_3, S2D-c_3, and S3D-c_{50} with M_G built for U_0. In Figs.12a-c, we set M_l to ILU(0) and vary n_{st}, while in Figs.12d-f, we fix n_{st} = 12 and vary M_l between ILU(0), ILU(1), and ILU(2). These plots echo the same observations made above from Table 2. Namely, there are almost no improvements in the number of iterations from n_{st} = 6 to 12 in Figs.12a-c, or from ILU(1) to ILU(2) in Figs.12d-f, only diminishing returns in terms of WCTs (see Table 3). Fig.12f also shows that if k in ILU(k) and n_{st} in Eq.29 are set too large, convergence for S3D-c_{50} deteriorates or diverges. We now answer two important questions: (1) Is M_G necessary to achieve fast convergence? The black dashed lines in Figs.12a-c show the convergence patterns of GMRES if preconditioned by only M = M_L (excluding M_G). Here, M_L is built from Eq.29 using M_l = ILU(0) and n_{st} = 6. In all cases, GMRES converges very slowly (or "diverges" by our standard) if M = M_L compared to the solid black lines. The corresponding WCTs for P2D-c_3 and S2D-c_3 at the 150th iteration, when neither has converged, are 137s and 96s, respectively. These WCTs are 3-4 times larger than those in Table 3 where M_L is paired with M_G. The WCT for S3D-c_{50} with M = M_L is 205s and GMRES converges in 116 iterations. While this is on par with Table 3, we note this is because S3D is a rather small domain (i.e., 67×68×71; still memory demanding because the matrix is denser than 2D). Hence, a few smoothing steps can quickly propagate error corrections spatially. For the much larger domain P2D-H, described later, GMRES does not converge with M = M_L even after 500 iterations and ~3 hours. With M_G, on the other hand, GMRES converges in ~30 minutes

and ~50 iterations (Table 2-3). These observations demonstrate a need for M_G to accelerate Krylov solvers; (2) Is it necessary to update M_G as cracks evolve? The red dashed lines in Figs.12d-f show the convergence of GMRES if preconditioned by M in Eq.28, where M_G is built for *intact* domains, i.e., basis vectors of M_G are not updated via Algorithm 1. Here too, we see GMRES converges very slowly compared to the solid red lines in Figs.12d-f, highlighting the need for updating M_G in evolving-crack problems. In Section 6.4, we discuss this point further and hypothesize a potentially better way of updating M_G.

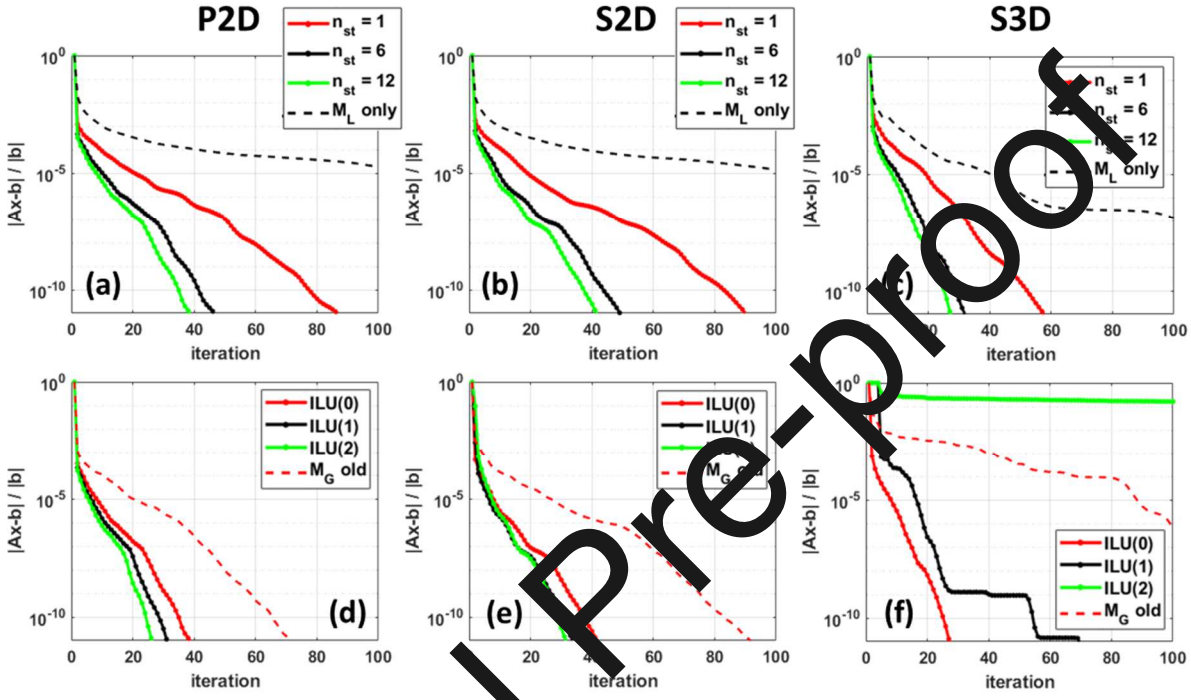


Fig 12. Normalized residual versus number of GMRES iterations for different number of smoothing stages (n_{st} = 1, 6, 12) and smoother types (M_l = ILU(0), ILU(1), ILU(2)). Plots correspond to P2D-c_3, S2D-c_3, and S3D-c_{50} using M_G based on ILU as the global preconditioner. In (a-c), we fix M_l = ILU(0) and vary n_{st}. In (d-f), we fix n_{st} = 12 and vary M_l. The black dashed lines in (a-c) correspond to using M_L in Eq.29, with M_l = ILU(0) and n_{st} = 6, as the sole preconditioner of GMRES (i.e., M_G is excluded). The red dashed lines in (d-f) correspond to using the preconditioner M in Eq.28 with M_G built for *intact* domains (i.e., no cracks included in the construction of M_G).

Finally, Tables 2-3 include results for AGMG. Compared to GMRES preconditioned by M_G, we see AGMG requires substantially more iterations to converge, often by a factor of 10-30. Its WCTs, however, are generally on par with GMRES, except S2D-$c_{0\text{-}3}$ where GMRES is twice as fast, and S3D-$c_{0,100}$ where AGMG outperforms GMRES. A very important caveat, however, is that *the AGMG solver corresponds to a compiled FORTRAN code, while the preconditioned GMRES solver corresponds to an uncompiled MATLAB code.* Through this lens, we see that despite being significantly disadvantaged, GMRES manages to perform as well or even better than AGMG. The difference amplifies dramatically as we increase the domain size, which is relatively small for the geometries in Fig.2. We thus consider a much larger version of P2D, called P2D-H. Fig.13a depicts its geometry, which contains 100 randomly placed fractures. The associated image has 2400×2400 pixels with 5,357,211 elements and 5,425,144 nodes. This amounts to ~11 million displacement

unknowns. The domain is pulled from the left side, while fixing the right boundary and leaving the top and bottom boundaries traction-free. The displacement field is solved on an Intel® Xeon® CPU E5-2690 v2 @ 3.00GHz, 126 GB RAM machine; shown by Fig.13b. Tables 2-3 summarize the number of iterations and WCTs of GMRES and AGMG for P2D-H. Not only does the preconditioned GMRES converge in significantly fewer iterations, by almost a factor of 26, its WCT is also smaller by over 2.5 times. If compiled, we expect GMRES to accelerate by another integer factor. Notice all calculations are in *series*. Added benefits can be reaped by exploiting the fact that the building and applying of M_G are amenable to parallelism. We conclude by noting that roughly 30-50% of the GMRES WCTs for P2D-$c_{0\text{-}3}$, S2D-$c_{0\text{-}3}$, and P2D-H, and ~70% for S3D-$c_{0,50,100}$, are spent on constructing M_G. Specifically, this cost is ~13s for P2D-$c_{0\text{-}3}$, ~7s for S2D-$c_{0\text{-}3}$, ~156s for S3D-$c_{0,50,100}$, and 310s for P2D-H, when M_G is based on A_0. The cost is slightly higher for U_0 due to implementational differences already discussed. Building M_L, by contrast, is cheap and costs ~1s for P2D-$c_{0\text{-}3}$ and S2D-$c_{0\text{-}3}$, ~19s for S3D-$c_{0,50,100}$, and 25-35s for P2D-H. Unlike AGMG, which must be executed from scratch at each loading step in an evolving-crack problem, M_G is reusable across loading steps and incurs a small cost in updating via Algorithm 1.

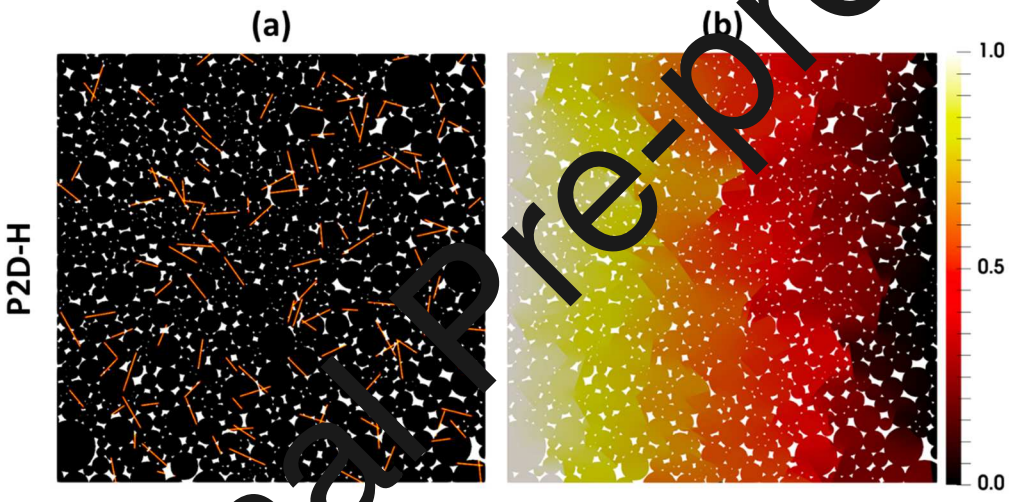


Fig 13. (a) Geometry and crack pattern of the P2D-H domain used to test the scalability of the multiscale preconditioner versus AGMG. (b) Displacement field associated with the simulation.

6. Discussion

6.1. Fine-scale smoothers

The smoothers used in this work, ILU(k) with k = 0, 1, and 2, are rather generic, or black-box, uninformed by the specific microstructure of the domains being targeted. We believe one could do much better by designing more specialized smoothers. In the geometric interpretation of PLMM,[23] the authors had introduced a second set of coarse grids, called *contact grids*. These straddled and covered a narrow region around each contact interface, where errors were known to be large due to the localization assumption in Eq.8 (see Figs.9-11). Contact grids are attractive because they do not cover all of Ω_s, and the cost of solving local problems on them is low. We hypothesize that a successful algebraic reinterpretation of these local problems as a smoother would result in even faster convergence rates than those listed in Table 2; after pairing with the M_G developed herein. The challenge is to develop this preconditioner without having to merge overlapping contact grids,

as was done previously;[23] a key drawback of PLMM in low-porosity domains. The formulation and testing of such a smoother is subject to ongoing research.

6.2. Frequency of preconditioner updates

Recall that crack patterns considered in Section 4 were *fixed*. If the cracks are evolving, due incremental loading of Ω_s, then the M_G constructed at one loading step can become ineffective in later loading steps. Specifically, the convergence rate of the linear solver would likely deteriorate progressively. The remedy this, M_G must be periodically updated. The adaptivity criterion in Section 3.6 ensures such updates are sufficiently infrequent and local to keep computational costs low. Each time cracks, Γ^F, intersect a contact interface between two grain grids, the basis functions of those grain grids only are recomputed. This implies that cracks nucleating and propagating within the sole confinement of a grain grid are ignored. In Section 5.2, we tested this criterion by updating M_G initially built for intact domains to fractured domains via Algorithm 1. The results in Table 2 confirmed that the criterion would preserve the convergence rate of iterative solvers across loading steps. In other words, the updated M_G is as good as an M_G built from scratch for the cracked domain. Future work will focus on applying the adaptivity criterion and Algorithm 1 in evolving-crack problems and quantify the computational gains obtained therefrom.

6.3. Computational complexity

The initial construction of M_G and its subsequent updates via Algorithm 1 are parallelizable. This is because computations associated with the basis vectors, $p^{gi}{}_k$, and correction vectors, c^{gi}, are fully decoupled across all grain grids. While a rigorous quantification of computational cost and parallel scalability is, as of now, outside our scope due to limitations in implementation, we provide a brief analysis of the computational complexity of M_G. Let Ω_s consists of N^f fine grids, N^g grain grids, and N^c contact interfaces. Also let $p^{gi}{}_k$ and c^{gi} be computed using a linear solver on Ω^{gi} that scales as $O(n^\beta)$, where n is the number of displacement unknowns and $\beta \in (1,3)$. Hence, the wall-clock time (WCT) associated with building M_G is:

$$\mathcal{T}_{build} = O(N^f D / N^g)^\beta \times (\overline{\#C^{gi}} D+1) N^g / P_{prc} \tag{32}$$

and the WCT associated with updating M_G is:

$$\mathcal{T}_{updat} = O(N^f D / N^g)^\beta \times (2\overline{\#C^{g_i}} D) N^c f^c / P_{prc} \tag{33}$$

where $\overline{\#C^{gi}}$ denotes the average number contact interfaces per grain grid, P_{prc} is the number of parallel processors employed, and f^c is the fraction of contact interfaces newly intersected by cracks (and not yet accounted for in M_G). Eq.32 multiplies the cost of a single fine-scale problem on Ω^{gi}, $O(N^f D/N^g)^\beta$, by the number of basis vectors, $\overline{\#C^{gi}} D$, and correction vectors, 1, to be built on Ω^{gi}. Multiplying the result by the number of grain grids yields the total cost of building M_G in series. Because all basis and correction vectors are decoupled, a maximum number of $P_{prc} = (\overline{\#C^{gi}} D+1) N^g$ processors may be used, which would reduce $\mathcal{T}_{build}$ to $O(N^f D/N^g)^\beta$. With reference to Eq.33, since $N^c f^c$ contact interfaces are intersected by Γ^F, and all basis vectors of the grain grids straddling such contacts must be updated, a total of $(2\overline{\#C^{gi}} D) N^c f^c$ local problems must be solved. For a maximum number of processors, the costs of updating M_G reduces, again, to $O(N^f D/N^g)^\beta$.

The WCT associated with applying M_G to precondition $\widehat{\mathrm{A}} w = v$ once is at most:

$$\mathcal{T}_{apply} = O(N^f D / N^g)^\beta \times (N^g / P_{prc}) \tag{34}$$

which corresponds to computing a correction vector c^{gi} per grain grid Ω^{gi}. Eq.34 is an upper bound because if v is zero on some Ω^{gi}, so is c^{gi} and no correction vector must be computed. In the next section, we discuss how the cost of $\mathcal{T}_{apply}$ may be eliminated. Lastly, M_G was developed here using the isotropic degradation model in Eq.4, which yields the linear system in Eq.6. If an anisotropic degradation model were adopted, M_G would be applied to the *linearized* system at each *Newton* iteration. Since M_G is held fixed between such iterations, computational gains can be significant.

6.4. Relation to algebraic multigrid (AMG)

The global preconditioner M_G is related to recent multiscale finite element, and more generally algebraic multigrid, preconditioners developed for Darcy-scale linear elastic deformation in porous media. The link is made explicit in Appendix B by formulating appropriate prolongation, P, and restriction, $R = P^T$, matrices that allow the derivation of the following coarse-scale system:

$$A_M x_M = b_M \quad \Rightarrow \quad \underbrace{P^T A_M P}_{A^o} x^o = \underbrace{P^T b_M}_{b^o} \quad \Rightarrow \quad A^o x^o = b^o \tag{35}$$

from the reduced system $A_M\, x_M = b_M$ in Eq.15. In Appendix B, we prove that $A^o = S_M$ and $b^o = s_M$ only if $\underline{G} = \emptyset$, which makes the coarse-scale system in Eq.35 equivalent to the Schur system in Eq.19. However, if $\underline{G} \neq \emptyset$, the two formulations differ by a residual derived in Appendix B. The above use of prolongation/restriction matrices is attractive for a number of reasons: (1) updating M_G can be accomplished by recomputing a select few columns of P; (2) the cost associated with recomputing the correction vectors c^{gi} per Krylov iteration, as noted in Remark 3, can be eliminated by augmenting the columns of P through the inclusion of c^{gi} as extra "basis vectors" (see Appendix B); (3) preliminary tests seem to suggest even less stringent adaptivity criteria than the one in Section 3.6 may be applied to preserve rapid convergence in Krylov solvers. For example: "update basis vectors on grain grids that straddle a *sample-spanning* fracture." Notice the cracks in Fig.3 were relatively short; and (4) the cost of applying M_G ($\mathcal{T}_{apply}$ in Eq.34) may be removed because solving the coarse-scale problem in Eq.35 is as expensive as solving the Schur system in Eq.19, which is negligible. Future work will systematically test these hypotheses.

7. Conclusion

We developed a multiscale preconditioner for solving the linear-elastic deformation of porous domains with arbitrary microstructures and crack patterns. It combines a global preconditioner, M_G, with a local smoother, M_L, to simultaneously attenuate low- and high-frequency error modes, respectively. Our main contribution was in the construction of M_G, which we based on an algebraic reinterpretation of a recent pore-level multiscale method (PLMM) developed by the authors.[23,24] The preconditioner interpretation has two advantages: (1) its application within existing codes is fully non-intrusive, and (2) key limitations associated with PLMM in reducing approximation errors via contact grids are removed through the flexible use of smoothers.[23] We verified and tested the convergence of the preconditioner in a Krylov solver for a range of 2D/3D microstructures and crack patterns and found favorable performance. Even so, the use of generic ILU(k) smoothers was found to be sub-optimal. If the fill-in level, k, or number of smoothing stages, n_{st}, are too large, the solver either converges slowly or diverges in our 3D domain. This calls for more specialized smoothers, which we think can be based on an algebraic reinterpretation of so-called "contact

problems" in PLMM.[23] Our second contribution was to propose an economic and adaptive way of updating M_G for evolving-crack problems that preserves rapid convergence of Krylov solvers. We also established a direct link between M_G (thus PLMM) and AMG (or MsFE) preconditioners by defining appropriate prolongation and restriction matrices in Appendix B. While a rigorous demonstration of parallel scalability was outside of the scope of our paper, a complexity analysis was provided that shows the construction and updating of M_G are fully parallelizable.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-2145222. The Department of Energy and Mineral Engineering (EME), the College of Earth and Mineral Sciences (EMS), and the Institutes of Energy and the Environment (IEE) are acknowleged for providing partial support. The Institute for Computational and Data Sciences at The Pennsylvania State University is also thanked for providing access to computational resources.

Appendix A. Accounting for interior cracks during reconstruction

To account for interior cracks in $\underline{\mathrm{G}}$ while reconstructing x^g, Eq.27 must be modified to:

$$\begin{aligned} x^g = c^g &- \sum_{g_i \in G} \sum_{c_j \in C_{g_i}} \mathrm{E}_g^{g_i} \left(\mathrm{A}_{g_i}^{g_i} \right)^{-1} \overline{\mathrm{A}}_{c_j}^{g_i} \mathrm{R}_c^{c_j} x^o \\ &- \sum_{g_i \in \underset{\sim}{G}} \sum_{c_j \in C_{g_i}} \mathrm{E}_g^{g_i} \left(\underset{\sim}{\mathrm{A}}_{g_i}^{g_i} \right)^{-1} \overline{\underset{\sim}{\mathrm{A}}}_{c_j}^{g_i} \mathrm{R}_c^{c_j} x^o \\ &- \sum_{g_i \in \underline{\underset{\sim}{G}}} \sum_{c_j \in C_{g_i}} \mathrm{E}_g^{g_i} \left(\underline{\underset{\sim}{\mathrm{A}}}_{g_i}^{g_i} \right)^{-1} \overline{\underline{\underset{\sim}{\mathrm{A}}}}_{c_j}^{g_i} \mathrm{R}_c^{c_j} x^o \end{aligned} \tag{A.1}$$

where summations are over the grain grids in $\underline{\mathrm{G}}$, and $\underline{\underset{\sim}{G}}$. We note that basis vectors associated with grain grids in $\underset{\sim}{G}$ already capture local cracks, because they are updated via Algorithm 1. Using the definitions of basis vectors $p^{g_i}{}_k$ in Eq.25, we rewrite Eq.A.1 as follows:

$$\begin{aligned} x^g = c^g &+ \sum_{g_i \in G} \sum_{c_j \in C_{g_i}} \sum_{k=(j-1)D+1}^{(j-1)D+D} [x^o]_k \mathrm{E}_g^{g_i} p_k^{g_i} \\ &+ \sum_{g_i \in \underset{\sim}{G}} \sum_{c_j \in C_{g_i}} \sum_{k=(j-1)D+1}^{(j-1)D+D} [x^o]_k \mathrm{E}_g^{g_i} \underset{\sim}{p}_k^{g_i} \\ &- \sum_{g_i \in \underline{\underset{\sim}{G}}} \sum_{c_j \in C_{g_i}} \mathrm{E}_g^{g_i} \left(\underline{\underset{\sim}{\mathrm{A}}}_{g_i}^{g_i} \right)^{-1} \overline{\underline{\underset{\sim}{\mathrm{A}}}}_{c_j}^{g_i} \mathrm{R}_c^{c_j} x^o \end{aligned} \tag{A.2}$$

The red highlighted terms have been stored in computer memroy, and need not be recomputed. The green highlighted terms, however, require solving a local problem per grain grid in $\underline{\mathrm{G}}$. This added computation, as remarked in Section 3.6, is optional when using M_G as a preconditioner to accelerate the convergence of Krylov solvers; as opposed to computing a first-pass solution.

Appendix B. Formulating prolongation and restriction matrices

Starting from $\mathrm{A_M}\, x_M = b_M$ in Eq.15, global preconditioning in AMG or MsFE follows:

$$\mathrm{A_M} x_M = b_M \quad \Rightarrow \quad \underbrace{\mathrm{P^T A_M P}}_{\mathrm{A^o}} x^o = \underbrace{\mathrm{P^T} b_M}_{b^o} \quad \Rightarrow \quad \mathrm{A^o} x^o = b^o \tag{B.1}$$

where P is a *prolongation matrix*, and its transpose $\mathrm{P^T}$ a *restriction matrix*. Solving Eq.B.1, yields the corase-scale solution x^o. An approximate solution for x_M is obtained via $x_M = \mathrm{P}x^o$, which interpolates x^o onto the fine grid. Here, we formulate an expression for P that corresponds to the $\mathrm{M_G}$ in Section 3.4. Recalling the following definitions for basis and correction vectors:

$$p_k^{g_i} = -\left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \bar{\mathrm{A}}_{c_j}^{g_i} \mathrm{R}_c^{c_j} e_k \qquad \forall k = (j-1)D + d \quad s.t. \quad j \in C^{g_i},\ d \in \{1, \ldots, D\} \tag{B.2a}$$

$$c^{g_i} = \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} b^{g_i} \tag{B.2b}$$

the prolongation matrix below can be formulated:

$$\mathrm{P} = \begin{bmatrix} p_1^{g_1} & p_2^{g_1} & \cdots & p_n^{g_1} \\ p_1^{g_2} & p_2^{g_2} & \cdots & p_n^{g_2} \\ \vdots & \vdots & \ddots & \vdots \\ p_1^{g_m} & p_2^{g_m} & \cdots & p_n^{g_m} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} p^{g_1} \\ p^{g_2} \\ \vdots \\ p^{g_m} \\ \mathrm{I} \end{bmatrix} = \begin{bmatrix} \mathrm{B} \\ \mathrm{I} \end{bmatrix} \tag{B.3}$$

where $n = N^o{}_c D$ is the number of coarse-scale unknowns, and $m = N^f{}_g D$ the number of fine-scale unknowns associated with Ω^{gi} (not Γ^{cj}). The identity matrix I is $N^o{}_c D \times N^o{}_c D$. In Eq.B.3, we have used the abbrieviated notation $p^{gi} = [p^{gi}{}_1, p^{gi}{}_2, \ldots, p^{gi}{}_n]$, which equals $p^{gi} = -(\mathrm{A}^{gi}{}_{gi})^{-1} \bar{\mathrm{A}}^{gi}{}_c$. Note that $\mathrm{B} = [(p^{g1})^\mathrm{T}, (p^{g2})^\mathrm{T}, \ldots, (p^{gm})^\mathrm{T}]^\mathrm{T}$ is identical to the *basis matrix* in Section 3.4. Substituting Eq.B.3 into Eq.B.1, we obtain the following expressions for $\mathrm{A^o}$ and b^o:

$$\mathrm{A^o} = \mathrm{B^T A}_g^g \mathrm{B} + \bar{\mathrm{A}}_g^c \mathrm{B} + \mathrm{B^T} \bar{\mathrm{A}}^g + \bar{\mathrm{A}}^c \tag{B.4a}$$

$$b^o = \mathrm{B^T} b^g + \bar{b}^c \tag{B.4b}$$

where we have used the block-structures of $\mathrm{A_M}$ and b_M in Eq.17. We claim:

Proposition 1. If $\underline{G} = \emptyset$, then $\mathrm{A^o} = \mathrm{S_M}$ and $b^o = s_M$.

Proof. Since $\underline{G} = \emptyset$, none of the basis vectors p^{gi} neglect the presence of cracks in the grain grids. We therefore drop the underline/under-tilde notation of Section 3.6 for simplicity. By substituting Eq.B.3 into Eq.B.4b, we obtain:

$$\begin{aligned} b^o &= \mathrm{B^T} b^g + \bar{b}^c = \bar{b}^c + \sum_{g_i \in G \cup \tilde{G}} \left(p^{g_i}\right)^\mathrm{T} b^{g_i} = \bar{b}^c - \sum_{g_i \in G \cup \tilde{G}} \bar{\mathrm{A}}_{g_i}^c \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} b^{g_i} \\ &= \bar{b}^c - \sum_{g_i \in G \cup \tilde{G}} \bar{\mathrm{A}}_{g_i}^c c^{g_i} = \bar{b}^c - \sum_{g_i \in G \cup \tilde{G}} \bar{\mathrm{A}}_g^c \mathrm{E}_g^{g_i} \mathrm{R}_g^{g_i} c^g = \bar{b}^c - \bar{\mathrm{A}}_g^c c^g = s_M \end{aligned} \tag{B.5}$$

which proves $b^o = s_M$. Note that in Eq.B.5, we have used the fact that $\mathrm{A_M}$ is symmetric and have substituted the definitions of p^{gi} and c^{gi}. Focusing next on $\mathrm{A^o}$, we write the first term in the RHS of Eq.B.4a as follows:

$$\begin{aligned} \mathrm{B}^{\mathrm{T}}\mathrm{A}_g^g\mathrm{B} &= \sum_{g_i\in G\cup\underline{G}} \left(p^{g_i}\right)^{\mathrm{T}} \mathrm{A}_{g_i}^{g_i} p^{g_i} \\ &= \sum_{g_i\in G\cup\underline{G}} \overline{\mathrm{A}}_{g_i}^{c} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \mathrm{A}_{g_i}^{g_i} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \overline{\mathrm{A}}_{c}^{g_i} = \sum_{g_i\in G\cup\underline{G}} \overline{\mathrm{A}}_{g_i}^{c} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \overline{\mathrm{A}}_{c}^{g_i} \end{aligned} \tag{B.6}$$

Similarly, the second term in the RHS of Eq.B.4a can be written as follows:

$$\overline{\mathrm{A}}_g^c\mathrm{B} = \sum_{g_i\in G\cup\underline{G}} \overline{\mathrm{A}}_{g_i}^{c} p^{g_i} = -\sum_{g_i\in G\cup\underline{G}} \overline{\mathrm{A}}_{g_i}^{c} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \overline{\mathrm{A}}_{c}^{g_i} \tag{B.7}$$

Note that Eqs.B.6 and B.7 sum to zero. Hence, Eq.B.4a reduces to:

$$\mathrm{A}^{\mathrm{o}} = \mathrm{B}^{\mathrm{T}}\overline{\mathrm{A}}_c^g + \overline{\mathrm{A}}_c^c = \left(\overline{\mathrm{A}}_c^g\mathrm{B} + \overline{\mathrm{A}}_c^c\right)^{\mathrm{T}} = \mathrm{S}_{\mathrm{M}} \tag{B.8}$$

where we have used Eq.22 and the fact that A^{o} is symmetric. This completes the proof. ■

In Proposition 1, $\mathrm{A}^{\mathrm{o}} = \mathrm{S}_{\mathrm{M}}$ holds only if $\underline{G} = \emptyset$. If $\underline{G} \neq \emptyset$, then $\mathrm{A}^{\mathrm{o}} \neq \mathrm{S}_{\mathrm{M}}$ because the summation of the first two terms in the RHS of Eq.B.4a would no longer be zero. It is straightforward to show that the difference between A^{o} and S_{M} would then satisfy the following expression:

$$\begin{aligned} \mathrm{A}^{\mathrm{o}} - \mathrm{S}_{\mathrm{M}} &= \mathrm{B}^{\mathrm{T}}\mathrm{A}_g^g\mathrm{B} + \overline{\mathrm{A}}_g^c\mathrm{B} = \sum_{g_i\in\underline{G}} \left(p^{g_i}\right)^{\mathrm{T}} \underline{\mathrm{A}}_{g_i}^{g_i} p^{g_i} + \sum_{g_i\in\underline{G}} \overline{\mathrm{A}}_{g_i}^{c} p^{g_i} \\ &= \sum_{g_i\in\underline{G}} \overline{\mathrm{A}}_{g_i}^{c} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \underline{\mathrm{A}}_{g_i}^{g_i} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \overline{\mathrm{A}}_{c}^{g_i} - \sum_{\underline{G}} \overline{\mathrm{A}}_{g_i}^{c} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \overline{\mathrm{A}}_{c}^{g_i} \\ &= \sum_{g_i\in\underline{G}} \overline{\mathrm{A}}_{g_i}^{c} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \underbrace{\left(\underline{\mathrm{A}}_{g_i}^{g_i} - \mathrm{A}_{g_i}^{g_i}\right)}_{\neq 0} \left(\mathrm{A}_{g_i}^{g_i}\right)^{-1} \overline{\mathrm{A}}_{c}^{g_i} \neq 0 \end{aligned} \tag{B.9}$$

Once x^o is computed by solving Eq.B.1, x^g can be reconstructed via Eq.27. As noted in Remark 3 of Section 3.4, if our goal is to precondition a Krylov solver, as opposed to computing a first-pass solution to $\widehat{\mathrm{A}}x = \hat{b}$, then the correction vector c^g must be computed repeatedly. This can become computationally cumbersome. The next proposition shows how this cost can be eliminated by interpreting the c^{gi} from Eq.B.2 as yet another basis vector that is computed once:

Proposition 2. Suppose the prolongation matrix is augmented as follows:

$$\mathrm{P}_{\mathrm{aug}} = \begin{bmatrix} \mathrm{B}_{\mathrm{aug}} \\ \mathrm{I} \end{bmatrix}, \qquad \mathrm{B}_{\mathrm{aug}} = \begin{bmatrix} \mathrm{B} & \mathrm{C} \end{bmatrix}, \qquad \mathrm{C} = \begin{bmatrix} c^{g_1} & & & \mathrm{O} \\ & c^{g_2} & & \\ & & \ddots & \\ \mathrm{O} & & & c^{g_m} \end{bmatrix}_{(N_g^f D)\times N^g} \tag{B.10}$$

Then, the coarse-scale system $\mathrm{A}^{\mathrm{o}}x^o = b^o$ in Eq.B.1 remains unaltered, but Eq.27 is replaced by:

$$x^g = \mathrm{C}y^o + \mathrm{B}x^o \tag{B.11}$$

where y^o is a vector of additional coarse-scale unknonws associated with the correction vectors c^{gi}. To compute y^o, the following coarse-scale system must be solved (after $\mathrm{A}^{\mathrm{o}}x^o = b^o$):

$$C^T AC y^o = C^T b - C^T AP x^o \tag{B.12}$$

Alternatively, x^o and y^o can be computed by substituting P_{aug} for P and $x^o_{aug} = [(x^o)^T, (y^o)^T]^T$ for x^o in Eq.B.1, where x^o_{aug} is the augmented vector of coarse-scale unknowns.

Proof. The proof requires merely substituting P_{aug} for P and x^o_{aug} for x^o in Eq.B.1 and using the fact that $P^T AC = O$. The latter is straightforward to verify. ■

References

1. Shukla, R., Ranjith, P., Haque, A. & Choi, X. A review of studies on CO2 sequestration and caprock integrity. *Fuel* **89**, 2651–2664 (2010).
2. Osborn, S. G., Vengosh, A., Warner, N. R. & Jackson, R. B. Methane contamination of drinking water accompanying gas-well drilling and hydraulic fracturing. *Proc. Natl. Acad. Sci. U. S. A.* **108**, 8172–8176 (2011).
3. Marx, J. C., Robbins, S. J., Grady, Z. A., Palmieri, F. L., Wohl, C. J. & Rabiei, A. Polymer infused composite metal foam as a potential aircraft leading edge material. *Appl. Surf. Sci.* **505**, 144114 (2020).
4. Marx, J., Portanova, M. & Rabiei, A. Performance of composite metal foam armors against various threat sizes. *J. Compos. Sci.* **4**, 176 (2020).
5. Shi, F., Song, Z., Ross, P. N., Somorjai, G. A., Ritchie, R. O. & Komvopoulos, K. Failure mechanisms of single-crystal silicon electrodes in lithium-ion batteries. *Nat. Commun.* **7**, 1–8 (2016).
6. Cheng, X., Wang, C., Sastry, A. M. & Choi, S. B. Investigation of failure processes in porous battery substrates: part II—simulation results and comparisons. *J. Eng. Mater. Technol.* **121**, 514–523 (1999).
7. Barak, M. M. & Black, M. A. A novel use of 3D printing model demonstrates the effects of deteriorated trabecular bone structure on bone stiffness and strength. *J. Mech. Behav. Biomed. Mater.* **78**, 455–464 (2018).
8. Wirth, A. j., Müller, R. & van Lenthe, G. H. Computational analyses of small endosseous implants in osteoporotic bone. *Eur. Cell Mater.* **20**, 58–71 (2010).
9. Blunt, M. J., Bijeljic, B., Dong, H., Gharbi, O., Iglauer, S., Mostaghimi, P., Paluszny, A. & Pentland, C. Pore-scale imaging and modelling. *Adv. Water Resour.* **51**, 197–216 (2013).
10. O'Sullivan, C. *Particulate Discrete Element Modelling: A geomechanics perspective*. *Applied Geotechnics* vol. 4 (CRC Press, 2011).
11. Cundall, P. A. & Strack, O. D. L. A discrete numerical model for granular assemblies. *Geeotechnique* **29**, 47–65 (1979).
12. Potyondy, D. O. & Cundall, P. A. A bonded-particle model for rock. *Int. J. Rock Mech. Min. Sci.* **41**, 1329–1364 (2004).
13. Liu, C. & Sun, W. ILS-MPM: An implicit level-set-based material point method for frictional particulate contact mechanics of deformable particles. *Comput. Methods Appl. Mech. Eng.* **369**, (2020).
14. Hu, M. & Rutqvist, J. Microscale mechanical modeling of deformable geomaterials with dynamic contacts based on the numerical manifold method. *Comput. Geosci.* **24**, 1783–1797 (2020).
15. Wildenschild, D. & Sheppard, A. P. X-ray imaging and analysis techniques for quantifying pore-scale structure and processes in subsurface porous medium systems. *Adv. Water Resour.* **51**, 217–246 (2013).

16. Moës, N., Dolbow, J. & Belytschko, T. A finite element method for crack growth without remeshing. *Int. J. Numer. Methods Eng.* **46**, 131–150 (1999).

17. Li, K., Atallah, N. M., Rodríguez-Ferran, A., Valiveti, D. M. & Scovazzi, G. The shifted fracture method. *Int. J. Numer. Methods Eng.* **122**, 6641–6679 (2021).

18. Miehe, C., Hofacker, M. & Welschinger, F. A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits. *Comput. Methods Appl. Mech. Eng.* **199**, 2765–2778 (2010).

19. Ambati, M., Gerasimov, T. & De Lorenzis, L. A review on phase-field models of brittle fracture and a new fast hybrid formulation. *Comput. Mech.* **55**, 383–405 (2015).

20. Mehmani, Y., Castelletto, N. & Tchelepi, H. A. Nonlinear convergence in contact mechanics: Immersed boundary finite volume. *Comput. Methods Appl. Mech. Eng.* **383**, 113929 (2021).

21. Javili, A., Morasata, R., Oterkus, E. & Oterkus, S. Peridynamics review. *Math. Mech. Solids* **24**, 3714–3739 (2019).

22. Saad, Y. *Iterative methods for sparse linear systems*. (SIAM, 2003). doi:doi:10.1137/1.9780898718003.

23. Mehmani, Y., Castelletto, N. & Tchelepi, H. A. Multiscale formulation of frictional contact mechanics at the pore scale. *J. Comput. Phys.* **450**, 110792 (2021).

24. Li, K. & Mehmani, Y. A pore-level multiscale method for the elastic deformation of fractured porous media (in revision). *J. Comput. Phys.* (2023).

25. Mehmani, Y. & Tchelepi, H. A. Multiscale computation of pore-scale fluid dynamics: single-phase flow. *J. Comput. Phys.* **375**, 1469–1487 (2018).

26. Mehmani, Y. & Tchelepi, H. A. Multiscale formulation of two-phase flow at the pore scale. *J. Comput. Phys.* **389**, 164–188 (2019).

27. Guo, B., Mehmani, Y. & Tchelepi, H. A. Multiscale formulation of pore-scale compressible Darcy-Stokes flow. *J. Comput. Phys.* **397**, 108849 (2019).

28. Beucher, S. & Lantuejoul, C. Use of watersheds in contour detection. *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation* (1979) doi:citeulike-article-id:4083187.

29. Efendiev, Y. & Hou, T. Y. *Multiscale finite element methods, theory and applications*. (Springer Science & Business Media, 2008). doi:10.1007/978-0-387-09496-0.

30. Hou, T. Y. & Wu, X. H. A multiscale finite element method for elliptic problems in composite materials and porous media. *J. Comput. Phys.* **134**, 169–189 (1997).

31. Castelletto, N., Hajibeygi, H. & Tchelepi, H. A. Multiscale finite-element method for linear elastic geomechanics. *J. Comput. Phys.* **331**, 337–356 (2017).

32. Xu, F., Hajibeygi, H. & Sluys, L. J. Multiscale extended finite element method for deformable fractured porous media. *J. Comput. Phys.* **436**, 110287 (2021).

33. Buck, M., Iliev, O. & Andrä, H. Multiscale finite element coarse spaces for the application to linear elasticity. *Cent. Eur. J. Math.* **11**, 680–701 (2013).

34. Babuška, I. & Osborn, J. E. Generalized finite element methods: their performance and their relation to mixed methods. *SIAM J. Numer. Anal.* **20**, 510–536 (1983).

35. Jenny, P., Lee, S. H. & Tchelepi, H. A. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *J. Comput. Phys.* **187**, 47–67 (2003).

36. Zhou, H. & Tchelepi, H. A. Two-stage algebraic multiscale linear solver for highly heterogeneous reservoir models. *SPE J.* **17**, 523–539 (2012).

37. Møyner, O. & Lie, K. A. A multiscale restriction-smoothed basis method for high contrast porous media represented on unstructured grids. *J. Comput. Phys.* **304**, 46–71 (2016).

38. Ţene, M., Wang, Y. & Hajibeygi, H. Adaptive algebraic multiscale solver for compressible flow in heterogeneous porous media. *J. Comput. Phys.* **300**, 679–694 (2015).

39. Bosma, S., Hajibeygi, H., Tene, M. & Tchelepi, H. A. Multiscale finite volume method for discrete fracture modeling on unstructured grids (MS-DFM). *J. Comput. Phys.* **351**, 145–164 (2017).

40. Nordbotten, J. M. & Bjørstad, P. E. On the relationship between the multiscale finite-volume method and domain decomposition preconditioners. *Comput. Geosci.* **12**, 367–376 (2008).

41. Arbogast, T., Pencheva, G., Wheeler, M. F. & Yotov, I. A multiscale mortar mixed finite element method. *SIAM Multiscale Model. Simul.* **6**, 319–346 (2007).

42. Wheeler, M. F., Wildey, T. & Xue, G. Efficient algorithms for multiscale modeling in porous media. *Numer. Linear Algebr. with Appl.* **17**, 771–785 (2010).

43. Ganis, B., Pencheva, G., Wheeler, M. F., Wildey, T. & Yotov, I. A Frozen Jacobian Multiscale Mortar Preconditioner for Nonlinear Interface Operators. *Multiscale Model. Simul.* **10**, 853–873 (2012).

44. Arbogast, T. & Xiao, H. Two-level mortar domain decomposition preconditioners for heterogeneous elliptic problems. *Comput. Methods Appl. Mech. Eng.* **292**, 221–242 (2015).

45. Mehmani, Y., Anderson, T., Wang, Y., Aryana, S. A., Battiato, I., Tchelepi, H. A. & Kovscek, A. R. Striving to translate shale physics across ten orders of magnitude: What have we learned? *Earth-Science Rev.* **223**, 103848 (2021).

46. Hajibeygi, H. Iterative multiscale finite volume method for multiphase flow in porous media with complex physics. **M**, 182 (2011).

47. Ruge, J. W. & Stüben, K. Algebraic Multigrid. in *Multigrid Methods* vol. 3 73–130 (Frontiers in Applied Mathematics, SIAM, 1987).

48. Wang, Y., Hajibeygi, H. & Tchelepi, H. A. Algebraic multiscale solver for flow in heterogeneous porous media. *J. Comput. Phys.* **259**, 284–303 (2014).

49. Borden, M. J., Verhoosel, C. V., Scott, M. A., Hughes, T. J. R. & Landis, C. M. A phase-field description of dynamic brittle fracture. *Comput. Methods Appl. Mech. Eng.* **217**–**220**, 77–95 (2012).

50. Sargado, J. M., Keilegavlen, E., Berre, I. & Nordbotten, J. M. High-accuracy phase-field models for brittle fracture based on a new family of degradation functions. *J. Mech. Phys.*

Solids **111**, 458–489 (2018).

51. Cao, H., Tchelepi, H. A., Wallis, J. & Yardumian, H. Parallel scalable unstructured CPR-type linear solver for reservoir simulation. in *SPE Annual Technical Conference and Exhibition* 9–12 (OnePetro, 2005). doi:10.2118/96809-MS.

52. Berg, S., Armstrong, R. & Wiegmann, A. Gildehauser sandstone. *http//www.digitalrocksportal .org /projects /134* (2018) doi:https://doi .org /10 .17612 /P7WW95.

53. Bass, J. D. Elasticity of minerals, glasses, and melts. *Miner. Phys. Crystallogr. A Handb. Phys. constants* **2**, 45–63 (1995).

54. Yvan Notay. An aggregation-based algebraic multigrid method. *Electron. Trans. Numer. Anal.* **37**, 123–146 (2010).

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:

Yashar Mehmani: Conceptualization, Methodology, Formal analysis, Writing - Original Draft, Funding acquisition

Kangan Li: Formal analysis, Validation, Writing - Review & Editing