# Composable Linear Solvers for Multiphysics

Jed Brown*, Matthew G. Knepley†, David A. May‡, Lois Curfman McInnes*, Barry Smith*

*Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL USA
[jedbrown,mcinnes,bsmith]@mcs.anl.gov

†Computation Institute
University of Chicago
Chicago, IL USA
knepley@ci.uchicago.edu

‡Department of Earth Sciences
ETH Zürich
Zürich, Switzerland
dave.may@erdw.ethz.ch

***Abstract*—The Portable, Extensible Toolkit for Scientific computing (PETSc), which focuses on the scalable solution of problems based on partial differential equations, now incorporates new components that allow full composability of solvers for multiphysics and multilevel methods. Through strong encapsulation, we achieve arbitrary, dynamic composition of hierarchical methods for coupled problems and allow customization of all components in composite solvers. For example, we support block decompositions with nested multigrid as well as multigrid on the fully coupled system with block-decomposed smoothers. This paper provides an overview of PETSc's new multiphysics capabilities, which have been used in parallel applications including lithosphere dynamics, subduction and mantle convection, ice sheet dynamics, subsurface reactive flow, fusion, mesoscale materials modeling, and power networks.**

***Keywords*-multiphysics, hierarchical solvers, composable solvers, Schur complement**

## I. Introduction

Multiphysics applications are of increasing interest as computational science communities strive to address broad questions about complex physical and engineered systems characterized by multiple, interacting physical processes that have traditionally been considered separately [1]. The Portable, Extensible Toolkit for Scientific computing (PETSc) [2], which focuses on the solution of large-scale problems based on partial differential equations (PDEs), includes several relatively recent components that have substantially improved composability for multiphysics and multilevel methods. The strong encapsulation of the PETSc design facilitates runtime composition of hierarchical methods for coupled problems without sacrificing the ability to customize certain components [3], [4], [5]. Applications using these multiphysics features include lithosphere dynamics [6], subduction and mantle convection [7], [8], [9], [10], [11], ice sheet dynamics [12], [13], [14], subsurface reactive flow [15], tokamak fusion [16], [17], mesoscale materials modeling [18], and power networks [19], [20].

This paper provides an overview of these multiphysics capabilities, along with illustrative examples. We emphasize our software strategy that enables users to experiment with these composable algorithms on parallel architectures. We refer readers to indicated references for discussion of performance for particular applications. Software with related capabilities includes MOOSE [21], the Teko package of Trilinos [22], and FEniCS [23]. PETSc also includes new capabilities for multiphysics time integration and nonlinear solvers [24], but these are outside the scope of this paper.

The remainder of this document is organized as follows. Section II introduces coupled problems and the concept of solver composition by relaxation and factorization. Section III introduces the `DM` object, which PETSc uses to define suitable decompositions and to provide the algebraic solvers with any necessary information that might depend on discretization, geometry, and physics. Section IV introduces the `FieldSplit` preconditioner and `Nest` matrix format, which implement relaxation and factorization splittings with efficient and flexible storage. Section V demonstrates this composability for applications in geodynamics, ice sheet modeling, and materials science. Section VI discusses conclusions and directions of future work.

## II. Composition of Linear Solvers

We begin by introducing notation and the various ways that solvers for individual physics components may be com-

bined to compute efficient solvers for the coupled physics. The algebraic system obtained by the discretization of a time-independent nonlinear PDE dependent on $p$ types of variables (physical quantities) can be written as

$$\begin{pmatrix} F_1(u_1, u_2, \ldots, u_p) \\ F_2(u_1, u_2, \ldots, u_p) \\ \vdots \\ F_p(u_1, u_2, \ldots, u_p) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

We make no particular assumptions about the relative sizes of the subvectors $u_i$. In addition, the subvectors themselves may consist of two or more subvectors related to different physical quantities; to avoid the textual complexity, we do not represent that case here. The Jacobian of the nonlinear function described above may be expressed in block form as

$$\begin{pmatrix} J_{11} & J_{12} & \cdots & J_{1p} \\ J_{21} & J_{22} & \cdots & J_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ J_{p1} & J_{p2} & \cdots & J_{pp} \end{pmatrix}.$$

Newton's method for solving $F(u) = 0$ is then simply $u^{n+1} = u^n - \lambda \hat{J}^{-1}(u^n)F(u^n)$, where $\hat{J}^{-1}(u^n)$ represents some approximate linear solve using some approximation of the true Jacobian. PETSc has a full suite of sophisticated Newton solvers, including line search techniques to compute $\lambda$, convergence determiners to stop the linear solver at an appropriate accuracy, sophisticated code to apply the Jacobian without explicitly forming it, colorings to compute the Jacobian efficiently [25], and procedures to allow lagging the computation of the (approximate) Jacobian and/or preconditioner [26]. However, none of those capabilities will be discussed here; this paper focuses on efficient techniques for approximate linear solution using preconditioned Krylov methods for problems arising in multiphysics simulations.

We begin the discussion by assuming that the entire Jacobian has been computed and is stored by using some sparse matrix representation. Later we will discuss how the computation can be optimized by removing the need to compute all of the Jacobian explicitly. Note that $J_{ii}$ represents coupling within a given physics submodel, while $J_{ij}$ represents coupling between two different physics submodels (for example, between a Stokes-like problem and thermal transport, as discussed in Section V). The simple block Jacobi preconditioner that involves only separate solves for each physics is given by

$$\begin{pmatrix} \hat{J}_{11}^{-1} & 0 & \cdots & 0 \\ 0 & \hat{J}_{22}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{J}_{pp}^{-1} \end{pmatrix}.$$

When the values from the previously computed physics are used before solving for the next physics, the result is block Gauss-Seidel, as given by

$$\begin{pmatrix} \hat{J}_{11} & 0 & \cdots & 0 \\ J_{21} & \hat{J}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ J_{p1} & J_{p2} & \cdots & \hat{J}_{pp} \end{pmatrix}^{-1}.$$

Note that applying block Gauss-Seidel requires applying $J_{ij}$ for *i>j*, while applying block Jacobi requires no off-diagonal Jacobian blocks. If one uses a symmetric sweep of block Gauss-Seidel, then one must be able to apply all off-diagonal blocks of $J_{ij}$, *i≠j*. If the off-diagonal blocks have not been computed and stored, their action on a vector may be obtained by differencing $F_i(u_1, \ldots, u_j, \ldots, u_p)$ over the $u_j$ variables. This requires computing the function $F_i(\ldots)$ associated with the $i^{\text{th}}$ physics without also computing all the other functions. An alternative that is computationally expensive, but useful for testing algorithms, is to difference the entire function and extract $F_i$ from the result. These features, combined with user-defined matrix-free computation of off-diagonal blocks, allow block methods to be implemented by using only *partial assembly of the Jacobian*, in which some blocks are explicitly computed and stored while others are not. The MOOSE package [21] is built on this principle.

We describe this solution process in more detail in order to provide a foundation for its subsequent extension. When the Jacobian is symmetric positive definite, the solution of any single block represents the projection (and hence correction) of the current error (in the $J$-norm) onto the subspace represented by that block's degrees of freedom. These concepts are well explained in the domain decomposition literature [27]. If the subspaces represented by the various physics are $J$-orthogonal, then the off-diagonal blocks of $J$ are zero and the iteration will converge in one iteration. The "strength" of the coupling in the off-diagonal blocks is a measure of the lack of orthogonality between the various subspaces. If one proceeds by orthogonalizing the second subspace with respect to the first, the third with respect to the first and second, and so on, then one obtains a block Cholesky factorization. To improve the convergence rate of block iterative methods, one then desires a preconditioner that works with subspaces that are more orthogonal than the original subspaces. For simplicity, we consider the block $2 \times 2$ case (the general case can be handled by recursion). The block LU factorization can be written as

$$J = LDU = \begin{pmatrix} I & 0 \\ J_{21}J_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} J_{11} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & J_{11}^{-1}J_{12} \\ 0 & I \end{pmatrix},$$

where $S = J_{22} - J_{21}J_{11}^{-1}J_{12}$. The inverse of the factorization can be written as

$$\begin{aligned} J^{-1} = U^{-1}(LD)^{-1} &= \begin{pmatrix} I & -J_{11}^{-1}J_{12} \\ 0 & I \end{pmatrix} \begin{pmatrix} J_{11} & 0 \\ J_{21} & S \end{pmatrix}^{-1} \\ &= (DU)^{-1}L^{-1} = \begin{pmatrix} J_{11} & J_{12} \\ 0 & S \end{pmatrix}^{-1} \begin{pmatrix} I & 0 \\ -J_{21}J_{11}^{-1} & I \end{pmatrix}, \end{aligned}$$

each of which requires two solves with $J_{11}$ and one solve with $S$ (the application of which embeds a solve with $J_{11}$). This is known as Schur complement reduction and is generally robust provided good preconditioners for $J_{11}$ and $S$ are available. In practice, one commonly iterates in the full space; uses only the lower $(LD)^{-1}$, upper $(DU)^{-1}$, or diagonal $D^{-1}$ part of the factorization; and solves only approximately with $S$. These shortcuts are justified by observing that the block triangular variants produced preconditioned operators with all eigenvalues equal to 1 and with minimal polynomial of degree 2; thus GMRES with exact solves for $J_{11}$ and $S$ would converge in two iterations [28], [29]. In practice, when the solves with $J_{11}$ and $S$ are approximate, the clustering of eigenvalues is usually more helpful to GMRES convergence than is the degradation because of non-normality introduced by the block triangular form. A similar result holds for a block diagonal preconditioner, which is popular for symmetric problems because it can be used with MINRES and has only three eigenvalues when applied exactly. These full-space iterations are often more efficient than Schur complement reduction but are more sensitive to relative scaling of each block.

The preconditioner for $J_{11}$ utilizes standard components, but $S$ is usually dense, thus requiring extra attention. Schur complement preconditioners can be obtained in a variety of ways, including several variants of approximate commutator arguments [30], [31], [32], [33], [34], Green's functions [35], and $\mathcal{H}$-matrices [36], [37]. We demonstrate the least squares commutator preconditioner in Section V.

Rather than splitting the matrix into large blocks and forming a preconditioner from solvers (for example, multigrid) on each block, one can perform multigrid on the entire system, basing the smoother on solves coming from the tiny blocks coupling the degrees of freedom at a single point (or small number of points). This approach is also handled in PETSc, but we will not elaborate on it here.

## III. The PETSc DM Object

Solvers based purely on provided matrix entries are limited in their ability to perform well since one cannot take advantage of geometric or modeling information. Hence, a solver framework that allows access to this information is vital. The difficulty is creating a flexible, hierarchical way to provide this information that is nonintrusive yet powerful. One unique feature of PETSc is the `DM` abstract class, which provides information to the algebraic solvers regarding the mesh and physics but does not impose constraints on their management. We note that `DM` is not a mesh management class and does not provide an interface to low-level mesh details; rather, `DM` provides an interface for accessing information relevant to and needed by the solver.

Before introducing the `DM` class, we introduce the index set, which is an abstraction of a parallel set of integers used for indexing into array-type objects. Several implementations are available, and more can be added easily. The first stores on each process an array of integers, each representing a single location; the second stores on each process an array of integers, each representing blocks of locations. Both implemenations are rather heavyweight but provide the most flexibility in indicating entries. The third representation is stride access, indicated by a first entry, the size between entries (the stride), and the number of entries; this representation is similar to MATLAB notation, for example, `[3:2:10]`. A subfield of a vector can then be represented by an index set that represents the indices of the subfield. A subsubfield can similarly be represented by an index set applied to the subfield.

What functionality does a `DM` need to provide to the algebraic solvers? These are represented as methods in the `DM` class in PETSc with capabilities to create

- a parallel global vector that can contain an entire field (PDE solution, for example);
- a local, ghosted vector for local function evaluations;
- a parallel sparse matrix that incorporates all or some of the coupling inherent in the discretization;
- a "refined-mesh" version of the given `DM`;
- a "coarsened-mesh" version of the given `DM`;
- a restriction of the `DM` to a subset of the physics (subset of the fields) or a subset of the domain;
- operators that interpolate, restrict, or inject between fields defined by two different `DM`s;
- an operator that maps between the global representation of a field and its local representation; and
- a way of naming and representing subfields of the global field by using index sets.

Using these operations, the PETSc linear and nonlinear solver infrastructure can automatically generate all work vectors, sparse matrices, and transfer operators needed by the multigrid solvers and composite (block) solvers. In addition, since `DM`s can recursively return `DM`s associated with subsets of physics, this process can be done in a hierarchical manner, with block solvers being embedded in multigrid solvers that are in turn embedded in multigrid solvers, and so on.

To make the management of user-provided code as simple as possible, we provide additional methods in `DM` that users implement to form initial guesses, compute nonlinear functions, and compute Jacobians. These enable an inner solver (which has access to an inner `DM`), for example, to explicitly compute a needed piece of the Jacobian, while leaving the entire large Jacobian unassembled.

## IV. FieldSplit Preconditioner

PETSc has a wide variety of composable linear solvers. Here we focus on one important class of preconditioners in PETSc, `FieldSplit`, which is designed for the flexible, hierarchical construction of block solvers. The `FieldSplit` preconditioner solves linear block systems

using either block relaxation or block factorization (Schur complement approaches). Arbitrary $p \times p$ block systems are supported, with the user specifying only index sets to identify each block (which may overlap). These index sets may be provided in the `DM` or passed directly to the `FieldSplit` preconditioner. Any algebraic solver can be used in each block; geometric multigrid may also be used if coarsening information is provided through the `DM` object or sub-`DM` objects representing fewer of the fields. As usual in PETSc, the `FieldSplit` preconditioner can be nested inside other preconditioners, such as multigrid or a higher-level `FieldSplit`, with construction of the hierarchy and other algorithmic choices as runtime options.

Fundamental in `FieldSplit` preconditioning is the concept of extracting submatrices from a global matrix acting on the coupled system. If a monolithic matrix format is used at the top level, the submatrix must be extracted and copied, a process that may involve excessive communication and increase memory requirements. The `Nest` matrix format alleviates this overhead by storing the submatrix associated with each physics independently. Although `Nest` is a more efficient format for `FieldSplit`, however, it cannot be used with certain solution methods, such as a sparse direct solver. To avoid an application depending on a specific matrix format (which would limit the available algorithms), PETSc provides a function that returns a generalized matrix view that is capable of assembly with the API for block problems regardless of the type of the larger matrix. When the larger matrix uses the `Nest` format, the submatrix is simply returned. When the larger matrix uses a monolithic format, a lightweight proxy is returned in the new matrix object. This proxy matrix is not fully functional, but it implements the assembly routines. If multiple levels of nesting are used, index translation in the proxy is flattened so that only one index translation can reach the top level. Since the matrix format can be chosen at runtime, a multiphysics application can assemble the diagonal blocks of a Jacobian in `Nest` or any monolithic format simply by calling the single-physics assembly routines. The off-diagonal blocks can be assembled with no global knowledge, only "local" indexing for the two physics submodels that the block couples.

To make this process concrete, we consider the case of three fields and the PETSc runtime options for various algorithms; in all cases the user code remains the same.

- Direct solve on the entire system:
```
-mat_type aij -ksp_type preonly -pc_type lu
```
- Coupled multigrid (without a Krylov accelerator):
```
-mat_type aij -ksp_type preonly -pc_type mg
-mg_levels_pc_type sor
-mg_levels_ksp_type richardson
```
- Jacobian-free (matrix-free) Newton-Krylov with one V-cycle of multigrid on the first two blocks and Jacobi on the third block:
```
-mat_type nest -snes_mf_operator
-ksp_type gmres -pc_type fieldsplit
-fieldsplit_0_pc_type mg
-fieldsplit_0_ksp_type preonly
-fieldsplit_0_mg_levels_pc_type sor
-fieldsplit_0_mg_levels_ksp_type preonly
-fieldsplit_1_pc_type mg
-fieldsplit_1_ksp_type preonly
-fieldsplit_1_mg_levels_pc_type sor
-fieldsplit_1_mg_levels_ksp_type preonly
-fieldsplit_2_pc_type jacobi
-fieldsplit_2_ksp_type preonly
```
- A Schur complement-based preconditioner that approximately eliminates the first two fields by using GMRES preconditioned by one V-cycle of uncoupled multigrid applied to each field.
```
-mat_type aij -ksp_type gmres -pc_type fieldsplit
-pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type full
-pc_fieldsplit_schur_precondition self
-pc_fieldsplit_0_fields 0,1
-pc_fieldsplit_1_fields 2
-fieldsplit_0_ksp_type gmres
-fieldsplit_0_pc_type fieldsplit
-fieldsplit_0_fieldsplit_0_pc_type mg
-fieldsplit_0_fieldsplit_1_pc_type mg
-fieldsplit_0_fieldsplit_0_ksp_type preonly
-fieldsplit_0_fieldsplit_1_ksp_type gmres
```

We note the following conventions for PETSc runtime options. Each linear (sub)solver has two parts: the preconditioner (indicated by `-pc_type`) and the Krylov accelerator (indicated by `-ksp_type`). A single application of the preconditioner is denoted by `-ksp_type preonly`, while `-ksp_type richardson` denotes applying the preconditioner until a sufficient reduction in the residual norm or a certain number of iterations is achieved. For example, `-ksp_type richardson -pc_type sor -ksp_max_it 3` indicates three iterations of SOR. The solvers for subproblems are prefixed to distinguish them. For example, `-fieldsplit_0_ksp_type preonly` indicates that no Krylov accelerator should be used for the first `FieldSplit` subsolve, while `-fieldsplit_0_mg_levels_pc_type sor` indicates the smoother for the levels of multigrid used in the first block solver. We note that the last example demonstrates the recursive use of the fieldsplit preconditioner.

## V. Numerical Examples

To demonstrate the flexibility offered by the simple definition of blocks in PETSc, we present problems in geodynamics, ice dynamics, and materials science. Only the operators themselves are defined in the application code, along with either a `DM` definition, a block size, or explicit marking of index sets. Applications make a single call to the PETSc linear (or nonlinear) solver regardless of the solution method or matrix format that they will use. We dynamically construct the hierarchical solver and preconditioner from the command line, allowing a wide range of methods to be easily compared. This approach removes the need for application scientists to explicitly select, design, and code a particular solver. Rather, they specify only the physical division between fields in the problem, through index sets

given to the DM; an appropriate solver can be constructed at runtime. The indicated references discuss performance of various algorithmic choices, which is beyond the scope of this paper.

### A. Geodynamics: Variable-viscosity Stokes

Our first example solves a variable coefficient Stokes problem from geodynamics for velocity $\boldsymbol{u}$ and pressure $p$,

$$-\nabla \cdot (\eta D\boldsymbol{u} - p\mathbf{1}) - \rho \boldsymbol{g} = 0$$
$$\nabla \cdot \boldsymbol{u} = 0,$$

where $\rho \boldsymbol{g}$ is the gravitational body force, $\eta$ is the effective viscosity (which is frequently discontinuous with jumps of several orders of magnitude), and $D$ is the symmetric gradient, that is, $D\boldsymbol{u} = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right)$. When discretized by using a mixed finite-element method from Underworld [38], the algebraic equations have the block form

$$\begin{pmatrix} J_{uu} & J_{pu}^T \\ J_{pu} & 0 \end{pmatrix},$$

where $J_{uu}$ is symmetric positive definite and $J_{pu}$ is the discrete divergence operator. Following an effective solution strategy from [8], we use a flexible GMRES method with upper triangular factorization, where inner solves with $J_{uu}$ are performed inexactly. Our preconditioner for $S = -J_{pu}J_{uu}^{-1}J_{pu}^T$ is the least squares commutator (LSC),

$$\hat{S}_{\mathrm{LSC}}^{-1} = L_p^{-1} J_{pu} M_d^{-1} J_{uu} M_d^{-1} J_{pu}^T L_p^{-1},$$

where $L_p = J_{pu} M_d^{-1} J_{pu}^T$ is a discrete scaled Laplacian with Neumann boundary conditions, and the scaling matrix $M_d = \mathrm{diag}(J_{uu})$. Since this problem has full Dirichlet boundary conditions, the constant pressure mode is in the null space of $S$ and $L^p$, so we remove it by projecting the Krylov iterates to have zero mean. The preconditioner $\hat{S}_{\mathrm{LSC}}$ is provided by PCLSC, which allows the user to provide $L_p$ and $M_d$ (e.g,. by rediscretization) but constructs them algebraically otherwise. Our code uses only the assembled operators $J_{uu}$, $J_{pu}$, and $J_{pu}^T$; optimizes storage by selecting the Nest matrix type; and configures the solver entirely with command line options. The outer flexible GMRES [39] method, scaling, and upper-triangular preconditioner structure are configured as follows:

```
-ksp_type fgmres -ksp_diagonal_scale
-ksp_rtol 1.0e-10
-pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type upper
```

with an iterative solve for $J_{uu}$

```
-fieldsplit_velocity_ksp_type cg
-fieldsplit_velocity_ksp_rtol 1.0e-6
-fieldsplit_velocity_pc_type bjacobi
-fieldsplit_velocity_sub_pc_type cholesky
-fieldsplit_velocity_sub_pc_factor_mat_ordering_type
  nd
```

an iterative solve for $S$

```
-fieldsplit_pressure_ksp_type fgmres
-fieldsplit_pressure_ksp_constant_null_space
-fieldsplit_pressure_ksp_monitor_short
-fieldsplit_pressure_pc_type lsc
```

and a low-accuracy solve with the scaled Laplacian $L_p$ appearing in the LSC preconditioner.

```
-fieldsplit_pressure_lsc_ksp_type cg
-fieldsplit_pressure_lsc_ksp_rtol 1.0e-2
-fieldsplit_pressure_lsc_ksp_constant_null_space
-fieldsplit_pressure_lsc_ksp_converged_reason
-fieldsplit_pressure_lsc_pc_type bjacobi
-fieldsplit_pressure_lsc_sub_pc_type icc
```

For a different discretization in which geometric interpolants are available, we choose geometric multigrid on the coupled problem, Galerkin coarse grid operators so that the problem does not need to be rediscretized, and a simpler field splitting (based on a scaled pressure "mass" matrix) used as a smoother on each level. The following options set up multigrid, the (multiplicative) splitting of interlaced variables into velocity and pressure parts, and the velocity part of the smoother (using default values for the pressure smoother).

```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_block_size 3
-mg_levels_pc_fieldsplit_0_fields 0,1
-mg_levels_pc_fieldsplit_1_fields 2
-mg_levels_fieldsplit_0_pc_type sor
```

The relaxation factor for the smoother on each level is tuned by estimating the largest eigenvalue of the preconditioned operator and optimizing a first-order Chebychev polynomial for the interval $[0.2\lambda_{\max}, 1.1\lambda_{\max}]$, which is achieved by the following options.

```
-mg_levels_ksp_type chebychev
-mg_levels_ksp_max_it 1
-mg_levels_ksp_chebychev_estimate_eigenvalues
   0,0.2,0,1.1
```

Note that the fieldsplit division can also be accomplished for mixed discretizations by looking for the saddle point (null block in the matrix).

```
-mg_levels_pc_fieldsplit_detect_saddle_point
```

These solvers are discussed in the context of mantle convection in [10], [11].

### B. Ice sheet dynamics: Polythermal ice flow

Nested fieldsplits have proved useful for several problems that couple variable-viscosity Stokes problems to other physics, including Lagrangian-free surfaces, viscoplasticity, and viscoelasticity. We consider a non-Boussinesq formulation of polythermal ice flow, which solves for momentum density $\rho \boldsymbol{u}$, pressure $p$, and total energy density $E$,

$$(\rho \boldsymbol{u})_t + \nabla \cdot \left(\rho \boldsymbol{u} \otimes \boldsymbol{u} - \eta D\boldsymbol{u}_i + p\mathbf{1}\right) - \rho \boldsymbol{g} = 0$$
$$\rho_t + \nabla \cdot \rho \boldsymbol{u} = 0$$
$$E_t + \nabla \cdot \left((E+p)\boldsymbol{u} - k_T \nabla T - k_\omega \nabla \omega\right)$$
$$-\eta D\boldsymbol{u}_i : D\boldsymbol{u}_i - \rho \boldsymbol{u} \cdot \boldsymbol{g} = 0,$$

where $T$ is temperature, $\omega$ is the volumetric melt fraction (porosity), and $\boldsymbol{u}_i$ is the velocity of the ice (not including the velocity of the melt). These terms are computed from $(\rho \boldsymbol{u}, p, E)$ by using the equation of state and additional

nonlinear constitutive relations defining viscosity $\eta$, thermal diffusivity $k_T$, and melt diffusivity $k_\omega$; see [40] for details. Similar models are used for other viscous, porous-media problems. The Jacobian has the block form

$$J = \begin{pmatrix} J_{uu} & J_{up} & J_{uE} \\ J_{pu} & 0 & 0 \\ J_{Eu} & J_{Ep} & J_{EE} \end{pmatrix},$$

representing the Stokes-like problem coupled to thermal transport. Although $J_{uE}$ represents important nonlinear coupling, it contributes little linear stiffness relative to the lower-triangular terms $J_{Eu}$ and $J_{Ep}$. Therefore, we choose lower-triangular preconditioning,

$$P^{-1} = \begin{pmatrix} \begin{pmatrix} J_{uu} & J_{up} \\ J_{pu} & 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} J_{Eu} & J_{Ep} \end{pmatrix} & J_{EE} \end{pmatrix}^{-1},$$

which, if applied exactly, ensures that the preconditioned operator has minimal polynomial of degree 2. The lower-triangular form is further from normal, which degrades GMRES convergence despite well-clustered eigenvalues [41], [42]. We solve the Stokes block using a block triangular preconditioner with a scaled mass matrix preconditioner for the Schur complement. Our 3D polythermal ice flow solver uses high-order, mixed finite-element methods on unstructured hexahedral meshes, where all high-order operators are applied matrix-free and only necessary low-order operators are assembled. The matrix-free methods have better asymptotics with increasing element order and use much less memory than assembled methods [43], [40]. First we create the top-level splitting and forward the (full accuracy) matrix-free diagonal blocks to the inner split.

```
-pc_type fieldsplit
-pc_fieldsplit_type multiplicative
-pc_fieldsplit_real_diagonal
```

Then we configure the solver for the inner Stokes problem to use GCR [44].

```
-fieldsplit_s_ksp_type gcr
-fieldsplit_s_ksp_rtol 1e-1
-fieldsplit_s_ksp_monitor_vht
-fieldsplit_s_ksp_monitor_singular_value
-fieldsplit_s_pc_type fieldsplit
-fieldsplit_s_pc_fieldsplit_type schur
-fieldsplit_s_pc_fieldsplit_real_diagonal
-fieldsplit_s_pc_fieldsplit_schur_factorization_type
  lower
-fieldsplit_s_fieldsplit_u_ksp_type gmres
-fieldsplit_s_fieldsplit_u_ksp_max_it 10
-fieldsplit_s_fieldsplit_u_pc_type asm
-fieldsplit_s_fieldsplit_u_sub_pc_type ilu
-fieldsplit_s_fieldsplit_u_sub_pc_factor_levels 1
-fieldsplit_s_fieldsplit_u_ksp_converged_reason
-fieldsplit_s_fieldsplit_p_ksp_type preonly
-fieldsplit_s_fieldsplit_p_ksp_max_it 1
-fieldsplit_s_fieldsplit_p_pc_type jacobi
```

Finally, we configure the solver for energy transport.

```
-fieldsplit_e_ksp_type gmres
-fieldsplit_e_ksp_converged_reason
-fieldsplit_e_pc_type asm
-fieldsplit_e_sub_pc_type ilu
-fieldsplit_e_sub_pc_factor_levels 2
```

This flexible configuration allows shifting work into the preconditioner to avoid the need for GMRES restarts and to enable parts of the problem with different spectral structure to be treated independently. In practice, the outermost linear solve converges in about three Krylov iterations, with most work occurring in the viscous part of the Stokes problem (prefixed by `-fieldsplit_s_fieldsplit_u_`) and the energy solve (prefixed by `-fieldsplit_e_`). We have found this configuration to be significantly more robust in realistic parameter ranges than is the $3 \times 3$ block preconditioner used by [45] for Boussinesq buoyancy-driven flows,

$$P_1^{-1} = \begin{pmatrix} J_{uu} & J_{up} & J_{uE} \\ 0 & B_{pp} & 0 \\ 0 & 0 & J_{EE} \end{pmatrix}^{-1},$$

where $B_{pp}$ is a scaled pressure mass matrix (or, for non-vanishing Reynolds number, "pressure convection diffusion" or "least squares commutator" preconditioners).

### C. Materials science: Phase field models

Our final example solves the Allen-Cahn variational inequalities that arise in phase field models in the mesoscale modeling of irradiated materials [18]. We consider isothermal, multicomponent phase transitions in a polygonal domain $\Omega \subset \mathbb{R}^d, d = 1, 2, 3$. Each phase at a particular point $(x, t) \in Q = \Omega \times [0, T_0], T_0 > 0$, is represented by the value of a component $u_i(x, t)$ of the order parameter $\boldsymbol{u} = (u_1, ..., u_N)^T$. In practical applications the components $u_i$ may represent concentrations or volume fractions of different phases in the system; that is, if $u_i = 0$, then phase $i$ is absent in that region, while if $u_i = 1$, only phase $i$ is present in that region. Since the concentrations are non-negative and sum to unity, the order parameter satisfies the constraints

$$\boldsymbol{u}(x, t) \in G = \{\boldsymbol{v} \in \mathbb{R}^d | v_i \geqslant 0, \sum_{i=1}^{N} v_i = 1\}, \quad \forall (x, t) \in Q.$$

The closed convex set $G \subset \mathbb{R}^N$ is often called Gibbs simplex. The Ginsburg-Landau total free energy of our system is assumed to take the form

$$E(\boldsymbol{u}) = \int_\Omega \frac{\epsilon}{2} \sum_{i=1}^{N} |\nabla u_i|^2 + \frac{1}{\epsilon} \Psi(\boldsymbol{u}) dx, \quad \epsilon > 0.$$

The quadratic term describes the interfacial energy, and $\Psi$ is a free energy that gives rise to phase separation. We focus on a multiphase version of the logrithmic free energy with the classical obstacle potential,

$$\Psi_0(\boldsymbol{u}) = \theta \sum_{i=1}^{N} u_i \ln(u_i) + \theta_c \frac{N}{2} \sum_{i=1}^{N} u_i (1 - u_i), \quad \boldsymbol{u} \in G,$$

where $\theta$ is the absolute temperature and $\theta_c$ is the critical temperature. The vector-valued Allen-Cahn equation

$$\epsilon \boldsymbol{u}_t = \epsilon \Delta \boldsymbol{u} - \frac{1}{\epsilon} P \nabla_{\boldsymbol{u}} \Psi(\boldsymbol{u})$$

is the projected $L^2-$gradient flow of the total free energy $E$. The orthogonal projection $P : \mathbb{R}^N \to \Sigma_0 = \{\boldsymbol{v} \in \mathbb{R}^N | \sum_i \boldsymbol{v}_i = 0\}$, defined by

$$P\boldsymbol{v} = \boldsymbol{v} - \frac{1}{N}(\boldsymbol{v} \cdot 1)1,$$

accounts for the fact that the values $\boldsymbol{u}(x,t) \in G \subset \Sigma = \{\boldsymbol{v} \in \mathbb{R}^N | \sum_i \boldsymbol{v}_i = 1\}$ must vary only on the affine hyperplane $\Sigma$. We also prescribe suitable initial conditions $\boldsymbol{u}(x,0) \in G$ on $\Omega$ and impose periodic boundary conditions.

The system can be written as the saddle-point matrix,

$$\begin{pmatrix} J & 0 & 0 & -I \\ 0 & J & 0 & -I \\ 0 & 0 & J & -I \\ -I & -I & -I & 0 \end{pmatrix}.$$

After the reduced-space active-set method is applied, the reduced system matrix has the same structure but with certain rows or columns of the matrices removed (those associated with actively constrained variables).

The options below use the block Schur complement preconditioner and the hypre [46] BoomerAMG algebraic multigrid preconditioner on the first blocks.

```
-ksp_type fgmres -pc_type fieldsplit
-pc_fieldsplit_detect_saddle_point
-pc_fieldsplit_type schur
-pc_fieldsplit_schur_precondition self
-fieldsplit_0_ksp_type preonly
-fieldsplit_0_pc_type hypre
-fieldsplit_1_ksp_type fgmres
-fieldsplit_1_pc_type lsc
```

The following commands invert the roles of the block preconditioner and multigrid and hence run (geometric) multigrid on the entire Stokes problem using a Schur complement `FieldSplit` preconditioner as the smoother on each level:

```
-ksp_type fgmres -pc_type mg -pc_mg_galerkin
-mg_levels_ksp_type fgmres
-mg_levels_ksp_max_it 2
-mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition user
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
-mg_levels_fieldsplit_ksp_max_it 5
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

## VI. Conclusion

This paper has demonstrated the power of new PETSc infrastructure called `FieldSplit` for constructing multiphysics solvers. Its hierarchical design and interaction with the PETSc `DM` class enable the runtime selection of a variety of nested block and multigrid preconditioners. Moreover, users can develop custom strategies that exploit operator-specific knowledge. We are incorporating new features to leverage emerging extreme-scale architectures, as well as complementary capabilities for multiphysics nonlinear solvers and time integration.


## Acknowledgment

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.


## References


[1] D. E. Keyes, L. C. McInnes, C. Woodward, W. D. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, A. Peters, D. Reynolds, B. Riviere, U. Rüde, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel, B. Smith, X. Tang, C. Wilson, and B. Wohlmuth, "Multiphysics Simulations: Challenges and Opportunities," Argonne National Laboratory, Tech. Rep. ANL/MCS-TM-321, Dec 2011, Report of workshop sponsored by the Institute for Computing in Science (ICiS), Park City, Utah, July 30 - August 6, 2011.

[2] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, "PETSc users manual," Argonne National Laboratory, Tech. Rep. ANL-95/11 - Revision 3.3, 2012.

[3] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *CRPC Handbook of Parallel Computing*. Morgan Kaufmann Publishers, 2002, ch. Software for the Scalable Solution of PDEs.

[4] ——, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing*, E. Arge, A. M. Bruaset, and H. P. Langtangen, Eds. Birkhauser Press, 1997, pp. 163–202.

[5] B. Smith, *Encyclopedia of Parallel Computing*. Springer, 2011, ch. PETSc, the Portable, Extensible Toolkit for Scientific computing.

[6] B. Aagaard, S. Kientz, M. G. Knepley, S. Somala, L. Strand, and C. Williams, "Pylith user manual version 1.6.1," 2011.

[7] R. F. Katz, M. G. Knepley, B. Smith, M. Spiegelman, and E. Coon, "Numerical simulation of geodynamic processes with the Portable Extensible Toolkit for Scientific Computation," *Phys. Earth Planet. In.*, vol. 163, pp. 52–68, 2007.

[8] D. A. May and L. Moresi, "Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics," *Physics of the Earth and Planetary Interiors*, vol. 171, no. 1-4, pp. 33–47, 2008, Recent Advances in Computational Geodynamics: Theory, Numerics and Applications.

[9] R. F. Katz, M. Spiegelman, and B. Holtzman, "The dynamics of melt and shear localization in partially molten aggregates," *Nature*, vol. 442, pp. 676–679, August 2006.

[10] S. Weatherley and R. Katz, "Melting and channelized magmatic flow in chemically heterogeneous, upwelling mantle," *Geochem. Geophys. Geosys.*, vol. 13, no. 1, p. Q0AC18, 2012.

[11] R. Katz and S. Weatherley, "Consequences of mantle heterogeneity for melt extraction at mid-ocean ridges," *Earth Planet. Sci. Lett.*, vol. 335-336, pp. 226–237, 2012.

[12] R. Katz and M. Worster, "The stability of ice-sheet grounding lines," *Proc. Roy. Soc. A*, vol. 466, pp. 1597–1620, 2010.

[13] T. Tautges et al., "SISIPHUS: Scalable ice-sheet solvers and infrastructure for petascale, high-resolution, unstructured simulations," http://trac.mcs.anl.gov/projects/sisiphus/wiki.

[14] J. Brown, B. Smith, and A. Ahmadia, "Achieving textbook multigrid efficiency for hydrostatic ice sheet flow," 2012, submitted to SIAM Journal on Scientific Computing.

[15] P. Lichtner et al., "PFLOTRAN project," http://ees.lanl.gov/pflotran/.

[16] A. Hakim, J. Cary, J. Candy, J. Cobb, R. Cohen, T. Epperly, D. Estep, S. Krasheninnikov, A. Malony, D. McCune, L. McInnes, A. Pankin, S. B. J. Carlsson, M. Fahey, R. Groebner, S. Kruger, M. Miah, A. Pletzer, S. Shasharina, S. Vadlamani, D. Wade-Stein, T. Rognlein, A. Morris, S. Shende, G. Hammett, K. Indareshkumar, A. Pigarov, and H. Zhang, "Coupled whole device simulations of plasma transport in tokamaks with the FACETS code," in *Proceedings of SciDAC 2010 Conference*, 2010.

[17] M. McCourt, T. D. Rognlien, L. C. McInnes, and H. Zhang, "Improving parallel scalability for edge plasma transport simulations with neutral gas species," in *Proceedings of the Twenty Second International Conference on Numerical Simulations of Plasmas, Sept. 7-9, 2011, Long Branch, NJ (to appear)*, 2012, also available as preprint ANL/MCS-P2018-0112.

[18] L. Wang, J. Lee, M. Anitesu, A. E. Azab, L. C. McInnes, T. Munson, and B. Smith, "A differential variational inequality approach for the simulation of heterogeneous materials," in *Proceedings of SciDAC 2011 Conference*, 2011.

[19] S. Abhyankar, "Development of an implicitly coupled electromechanical and electromagnetic transients simulator for power systems," Ph.D. dissertation, Illinois Institute of Technology, 2011.

[20] S. Abhyankar, B. Smith, H. Zhang, and A. Flueck, "Using PETSc to develop scalable applications for next-generation power grid," in *Proceedings of the 1st International Workshop on High Performance Computing, Networking and Analytics for the Power Grid*. ACM, 2011.

[21] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandié, "Moose: A parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering and Design*, vol. 239, no. 10, pp. 1768–1778, 2009.

[22] M. A. Heroux and J. M. Willenbring, "Trilinos users guide," Sandia National Laboratories, Tech. Rep. SAND2003-2952, 2003.

[23] A. Logg, K. Ølgaard, M. Rognes, G. Wells, J. Jansson, R. Kirby, M. Knepley, D. Lindbo, and O. Skavhaug, "The FEniCS project," 2011, a continually updated technical report. http://fenicsproject.org.

[24] P. Brune, M. Knepley, B. Smith, and X. Tu, "Composing scalable nonlinear solvers," Argonne National Laboratory, Preprint ANL/MCS-P2010-0112, 2012.

[25] P. Hovland, B. Norris, and B. Smith, "Making automatic differentiation truly automatic: Coupling PETSc with ADIC," *Future Generation Computer Systems*, vol. 21, no. 8, pp. 1426–1438, 2005.

[26] W. D. Gropp, L. C. McInnes, and B. F. Smith, "Scalable libraries for solving systems of nonlinear equations and unconstrained minimization problems," in *Proceedings of the Scalable Parallel Libraries Conference*. Mississippi State University: IEEE, 1995, pp. 60–67.

[27] B. F. Smith, P. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[28] M. Murphy, G. Golub, and A. Wathen, "A note on preconditioning for indefinite linear systems," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 1969–1972, 2000.

[29] I. C. F. Ipsen, "A note on preconditioning nonsymmetric matrices," *SIAM J. Sci. Comput.*, vol. 23, pp. 1050–1051, 2001.

[30] H. Elman, V. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, "A taxonomy and comparison of parallel block multilevel preconditioners for the incompressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 227, no. 1, pp. 1790–1808, 2008.

[31] D. Silvester, H. Elman, D. Kay, and A. Wathen, "Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow," *Journal of Computational and Applied Mathematics*, vol. 128, no. 1-2, pp. 261–279, 2001.

[32] H. Elman, "Preconditioning for the steady-state Navier-Stokes equations with low viscosity," *SIAM Journal on Scientific Computing*, vol. 20, no. 4, pp. 1299–1316, 1999.

[33] H. Elman, V. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, "Block preconditioners based on approximate commutators," *SIAM Journal on Scientific Computing*, vol. 27, no. 5, pp. 1651–1668, 2006.

[34] H. Elman and R. Tuminaro, "Boundary conditions in approximate commutator preconditioners for the Navier-Stokes equations," *Electronic Transactions on Numerical Analysis*, vol. 35, pp. 257–280, 2009.

[35] D. Kay and D. Loghin, "A Green's function preconditioner for the steady-state Navier-Stokes equations," Oxford University Computing Laboratory, Tech. Rep. 99/06, 1999.

[36] M. Bebendorf and W. Hackbusch, "Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with $L^\infty$-coefficients," *Numerische Mathematik*, vol. 95, no. 1, pp. 1–28, 2003.

[37] W. Hackbusch, B. Khoromskij, and R. Kriemann, "Direct Schur complement method by domain decomposition based on $\mathcal{H}$-matrix approximation," *Computing and Visualization in Science*, vol. 8, no. 3, pp. 179–188, 2005.

[38] L. Moresi, S. Quenette, V. Lemiale, C. Meriaux, B. Appelbe, and H. Mühlhaus, "Computational approaches to studying non-linear dynamics of the crust and mantle," *Physics of the Earth and Planetary Interiors*, vol. 163, no. 1-4, pp. 69–82, 2007.

[39] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM Journal on Scientific Computing*, vol. 14, no. 2, pp. 461–469, 1993.

[40] J. Brown, "Computational methods for ice flow simulation," Ph.D. dissertation, ETH Zürich, 2011.

[41] N. Nachtigal, S. Reddy, and L. Trefethen, "How fast are nonsymmetric matrix iterations?" *SIAM Journal on Matrix Analysis and Applications*, vol. 13, p. 778, 1992.

[42] M. Embree, "How descriptive are GMRES convergence bounds," Oxford University Computing Laboratory, Tech. Rep. 08, 1999.

[43] J. Brown, "Efficient nonlinear solvers for nodal high-order finite elements in 3D," *Journal of Scientific Computing*, vol. 45, pp. 48–63, 2010.

[44] S. Eisenstat, H. Elman, and M. Schultz, "Variational iterative methods for nonsymmetric systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 20, no. 2, pp. 345–357, 1983.

[45] H. Elman, M. Mihajlovic, and D. Silvester, "Fast iterative solvers for buoyancy driven flow problems," *Journal of Computational Physics*, vol. 230, no. 10, pp. 3900 – 3914, 2011.

[46] R. Falgout, "hypre Web page," http://www.llnl.gov/CASC/hypre.