DMSTAG: Native support to for staggered finite difference grid formulations within PETSc to

Patrick Sanan (1), Dave A. May (2), ***Boris J. P. Kaus (3),*** *Anton A. Popov (3), Paul J. Tackley (1), Taras Gerya (1)*

(1) Institute of Geophysics, ETH Zurich, Zurich, Switzerland, (2) Department of Earth Sciences,
University of Oxford, Oxford, United Kingdom, (3) Institute of Geosciences, JGU Mainz, Mainz, Germany

PETSc is the most widely used package within geodynamics to develop 3D scalable parallel codes. So far, however, it was no so straightforward to develop staggered grid discretizations within PETSc, as the different variables (such as velocity components or pressure), are located at different point in space. It thus usually takes some time to sort out the numbering of the different degrees of freedom, particularly in parallel. The same problem is also the main reason that students find it tricky to write a staggered finite difference code in MATLAB, and a finite element code is easier to develop.

To circumvent this, we have worked on the DMSTAG framework for PETSc, which encapsulates and optimizes operations essential to staggered-grid finite volume Stokes flow solvers. It provides a parallel staggered-grid abstraction with a high-level interface to C and Fortran natively build into PETSc. On top of this abstraction, tools are available to define boundary conditions and interact with particle systems (through DMSWARM, implemented in PETSc), such that you can add "markers" to your simulations with a few lines of codes. Tools and examples to efficiently solve Stokes systems defined on the grid are provided in small (direct solver), medium (simple preconditioners), and large (block factorization and multigrid) model regimes. These options, in addition to a wealth of solver options provided natively by PETSc, will make the most modern solution techniques available from a common interface.

We present the current stage of development of DMSTAG, which may be integrated into the PETSc packages in their next release (3.10), due this autumn. We discuss examples and show that this may actually make it easier to teach students how to write staggered finite difference codes from scratch, compared to the classical way of using MATLAB for this. The major advantage is that you directly have a massively parallel code that can be used for production runs. We discuss the various design aspects of DMSTAG, and future plans to create a higher level interface (StagBL), and coupling this with various application codes within geodynamics (LaMEM, I3ELVIS and StagYY).