Software Design in Computational Science

Matthew Knepley

Computation Institute
University of Chicago

Department of Molecular Biology and Physiology
Rush University Medical Center

Lindbergh Lecture
Department of Mechanical Engineering
University of Wisconsin – Madison Apr 21, 2011

Outline

1. What can PETSc do?
   - What is PETSc?
   - Who uses PETSc?
2. Value of Design

Outline

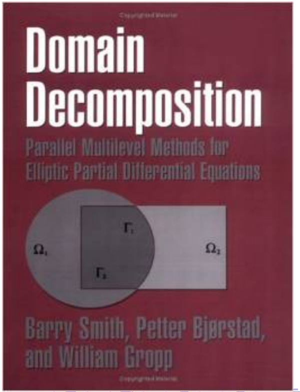
1. What can PETSc do?
   - What is PETSc?
   - Who uses PETSc?

How did PETSc Originate?

PETSc was developed as a Platform for Experimentation

We want to experiment with different

- Models
- Discretizations
- Solvers
- Algorithms
    - which blur these boundaries



The Role of PETSc

Developing parallel, nontrivial PDE solvers that deliver high performance is still difficult and requires months (or even years) of concentrated effort.

PETSc is a toolkit that can ease these difficulties and reduce the development time, but it is not a black-box PDE solver, nor a silver bullet.

— Barry Smith

Advice from Bill Gropp

You want to think about how you decompose your data structures, how you think about them globally. [...] If you were building a house, you'd start with a set of blueprints that give you a picture of what the whole house looks like. You wouldn't start with a bunch of tiles and say. "Well I'll put this tile down on the ground, and then I'll find a tile to go next to it." But all too many people try to build their parallel programs by creating the smallest possible tiles and then trying to have the structure of their code emerge from the chaos of all these little pieces. You have to have an organizing principle if you're going to survive making your code parallel.

(http://www.rce-cast.com/Podcast/rce-28-mpich2.html)

What is PETSc?

A freely available and supported research code for the parallel solution of nonlinear algebraic equations

Free

- Download from http://www.mcs.anl.gov/petsc
- Free for everyone, including industrial users

Supported

- Hyperlinked manual, examples, and manual pages for all routines
- Hundreds of tutorial-style examples
- Support via email: petsc-maint@mcs.anl.gov

Usable from C, C++, Fortran 77/90, Matlab, Julia, and Python

What is PETSc?

- Portable to any parallel system supporting MPI, including:
  - Tightly coupled systems
    - Cray XT6, BG/Q, NVIDIA Fermi, K Computer
  - Loosely coupled systems, such as networks of workstations
    - IBM, Mac, iPad/iPhone, PCs running Linux or Windows
- PETSc History
  - Begun September 1991
  - Over 60,000 downloads since 1995 (version 2)
  - Currently 400 per month
- PETSc Funding and Support
  - Department of Energy
    - SciDAC, MICS Program, AMR Program, INL Reactor Program
  - National Science Foundation
    - CIG, CISE, Multidisciplinary Challenge Program

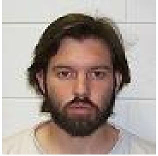
The PETSc Team

Bill Gropp

Barry Smith

Satish Balay

Jed Brown



Matt Knepley

Lisandro Dalcin

Hong Zhang

Mark Adams

Toby Issac

Outline

1. What can PETSc do?
   - What is PETSc?
   - Who uses PETSc?

Who Uses PETSc?

Computational Scientists

- Earth Science
  - PyLith (CIG)
  - Underworld (Monash)
  - Magma Dynamics (LDEO, Columbia, Oxford)
- Subsurface Flow and Porous Media
  - STOMP (DOE)
  - PFLOTRAN (DOE)

Who Uses PETSc?

Computational Scientists

- CFD
  - Firedrake
  - Fluidity
  - OpenFOAM
  - freeCFD
  - OpenFVM
- MicroMagnetics
  - MagPar
- Fusion
  - XGC
  - BOUT++
  - NIMROD

Who Uses PETSc?

Algorithm Developers

- Iterative methods
  - Deflated GMRES
  - LGMRES
  - QCG
  - SpecEst
- Preconditioning researchers
  - Prometheus (Adams)
  - ParPre (Eijkhout)
  - FETI-DP (Klawonn and Rheinbach)

Who Uses PETSc?

Algorithm Developers

- Finite Elements
  - libMesh
  - MOOSE
  - PETSc-FEM
  - Deal II
  - OOFEM
- Other Solvers
  - Fast Multipole Method (PetFMM)
  - Radial Basis Function Interpolation (PetRBF)
  - Eigensolvers (SLEPc)
  - Optimization (TAO)

What Can We Handle?

- PETSc has run implicit problems with over 500 billion unknowns
  - UNIC on BG/P and XT5
  - PFLOTRAN for flow in porous media
- PETSc has run on over 290,000 cores efficiently
  - UNIC on the IBM BG/P Jugene at Jülich
  - PFLOTRAN on the Cray XT5 Jaguar at ORNL
- PETSc applications have run at 23% of peak (600 Teraflops)
  - Jed Brown on NERSC Edison
  - HPGMG code

What Can We Handle?

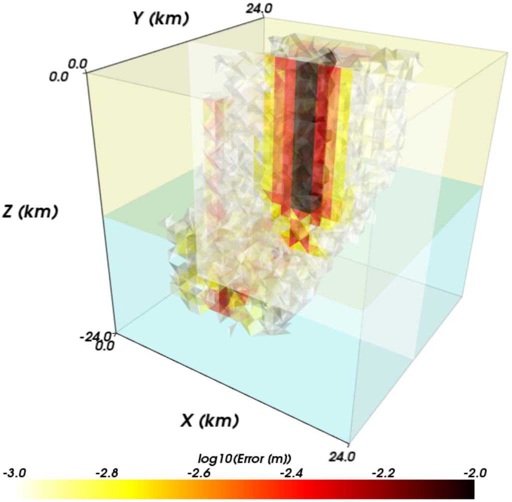
- PETSc has run implicit problems with over 500 billion unknowns
  - UNIC on BG/P and XT5
  - PFLOTRAN for flow in porous media
- PETSc has run on over 290,000 cores efficiently
  - UNIC on the IBM BG/P Jugene at Jülich
  - PFLOTRAN on the Cray XT5 Jaguar at ORNL
- PETSc applications have run at 23% of peak (600 Teraflops)
  - Jed Brown on NERSC Edison
  - HPGMG code

What Can We Handle?

- PETSc has run implicit problems with over 500 billion unknowns
  - UNIC on BG/P and XT5
  - PFLOTRAN for flow in porous media
- PETSc has run on over 290,000 cores efficiently
  - UNIC on the IBM BG/P Jugene at Jülich
  - PFLOTRAN on the Cray XT5 Jaguar at ORNL
- PETSc applications have run at 23% of peak (600 Teraflops)
  - Jed Brown on NERSC Edison
  - HPGMG code

PyLith

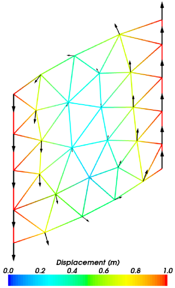
- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Nonlinear visco-plastic
  - Finite deformation
  - Fault constitutive models
- Multiple meshes
  - 1D, 2D, 3D
  - Hex and tet meshes
- Parallel
  - PETSc solvers
  - DMPlex mesh management

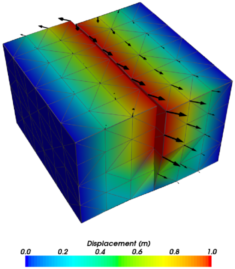


[a] Aagaard, Knepley, Williams

Multiple Mesh Types

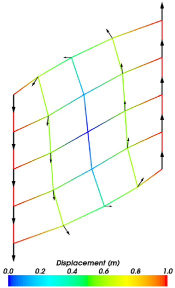
Triangular

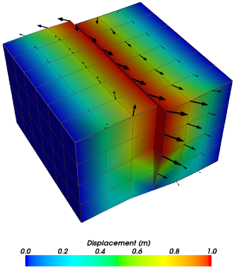




Tetrahedral

Rectangular

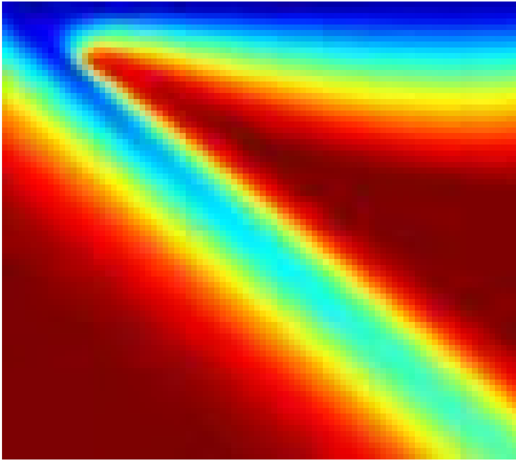




Hexahedral

Magma Dynamics

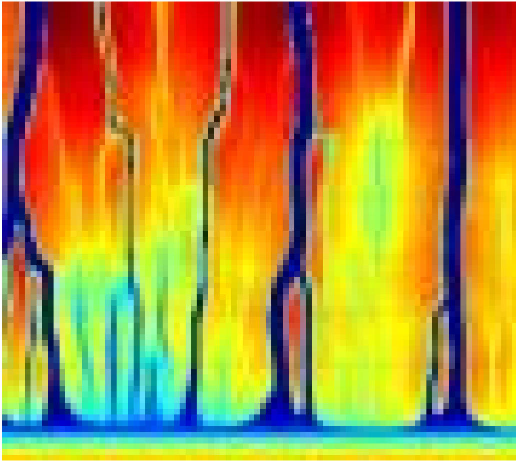
- Couples scales
  - Subduction
  - Magma Migration
- Physics
  - Incompressible fluid
  - Porous solid
  - Variable porosity
- Deforming matrix
  - Compaction pressure
- Code generation
  - FEniCS
- Multiphysics Preconditioning
  - PETSc FieldSplit



[a] Katz, Speigelman

Magma Dynamics

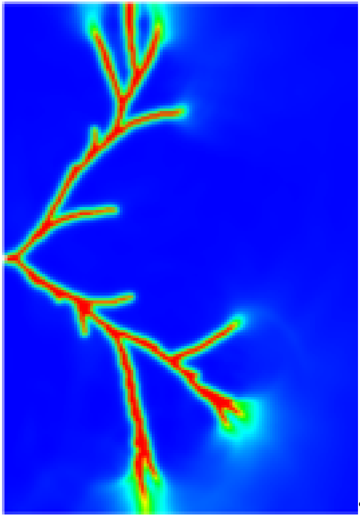
- Couples scales
  - Subduction
  - Magma Migration
- Physics
  - Incompressible fluid
  - Porous solid
  - Variable porosity
- Deforming matrix
  - Compaction pressure
- Code generation
  - FEniCS
- Multiphysics Preconditioning
  - PETSc FieldSplit



[a] Katz, Speigelman

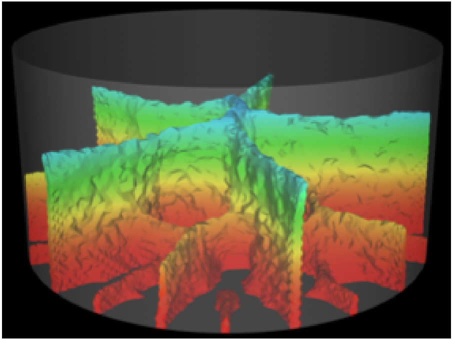
Fracture Mechanics

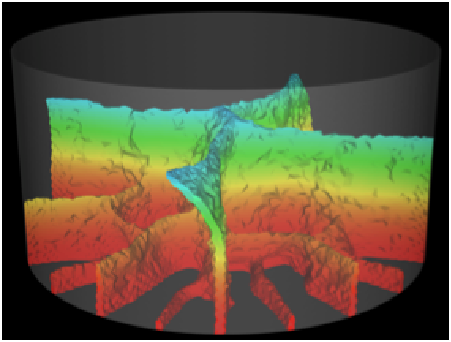
- Full variational formulation
  - Phase field
  - Linear or Quadratic penalty
- Uses TAO optimization
  - Necessary for linear penalty
  - Backtacking
- No prescribed cracks (movie)
  - Arbitrary crack geometry
  - Arbitrary intersections
- Multiple materials
  - Composite toughness


[a]

[a] Bourdin

Fracture Mechanics

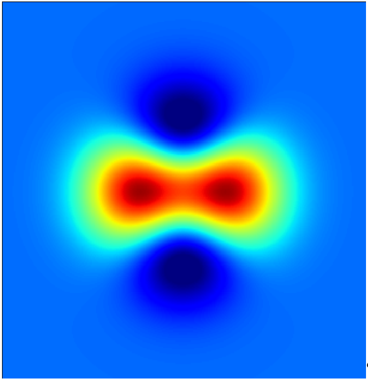


 [1]

[1] Bourdin

# Vortex Method
t = 000

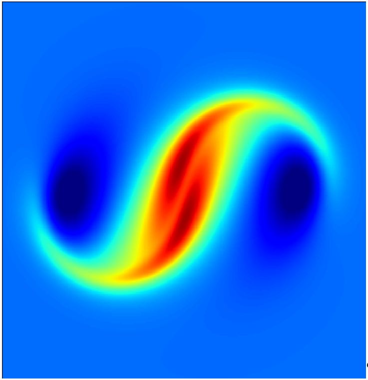
- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU


[a]

[a] Cruz, Yokota, Barba, Knepley

# Vortex Method
t = 100

- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU

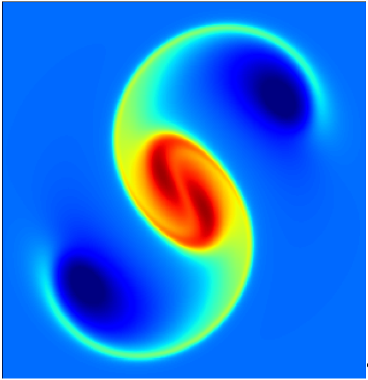

[a]

[a] Cruz, Yokota, Barba, Knepley

Vortex Method

t = 200

- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU

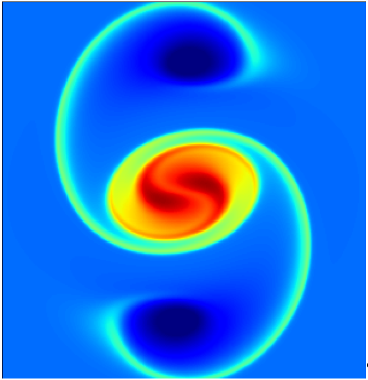

[a]

[a] Cruz, Yokota, Barba, Knepley

Vortex Method

t = 300

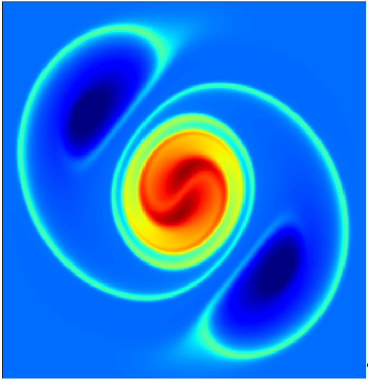
- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU

[a]

[a] Cruz, Yokota, Barba, Knepley

# Vortex Method
t = 400

- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU

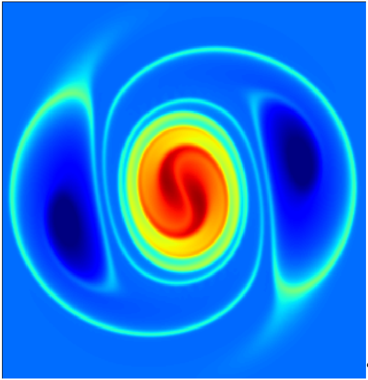

[a]

[a] Cruz, Yokota, Barba, Knepley

Vortex Method

t = 500

- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU

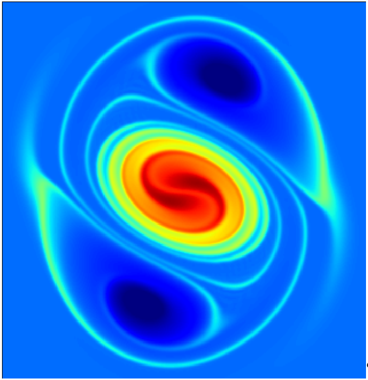

[a]

[a] Cruz, Yokota, Barba, Knepley

Vortex Method

t = 600

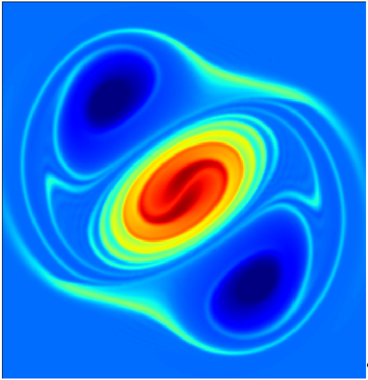
- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU

[a]

[a] Cruz, Yokota, Barba, Knepley

# Vortex Method
t = 700

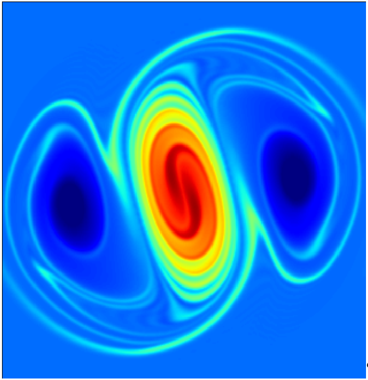
- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU

[a]

[a] Cruz, Yokota, Barba, Knepley

# Vortex Method
t = 800

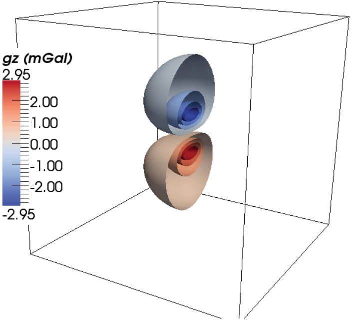
- Incompressible Flow
  - Gaussian vortex blobs
  - High Re
- PetFMM
  - 2D/3D domains
  - Automatic load balancing
  - Variety of kernels
  - Optimized with templates
- PetRBF
  - Variety of RBFs
  - Uses PETSc solvers
  - Scalable preconditioner
- Parallelism
  - MPI
  - GPU

 [a]

[a] Cruz, Yokota, Barba, Knepley

Gravity Anomaly Modeling

- Potential Solution
  - Kernel of inverse problem
  - Needs optimal algorithm
- Implementations
  - Direct Summation
  - FEM
  - FMM
- Parallelism
  - MPI
  - 4000+ cores
  - All methods scalable

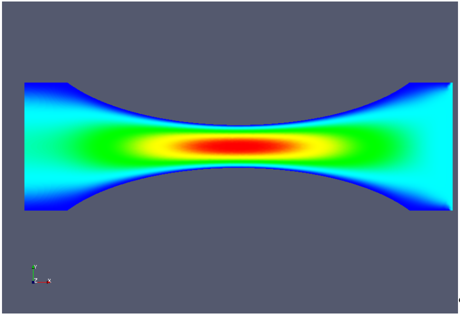


[a]

[a] May, Knepley

FEniCS-Apps
Rheagen

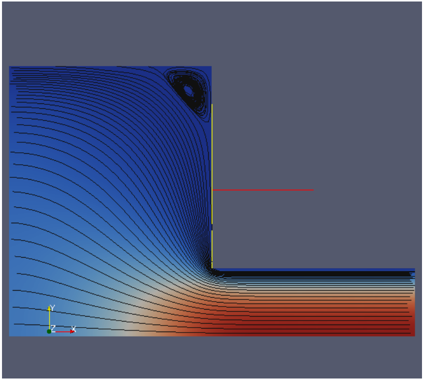
- Rheologies
  - Maxwell
  - Grade 2
  - Oldroyd-B
- Stabilization
  - DG
  - SUPG
  - EVSS
  - DEVSS
  - Macroelement
- Automation
  - FIAT (elements)
  - FFC (weak forms)

[a]

[a] Terrel

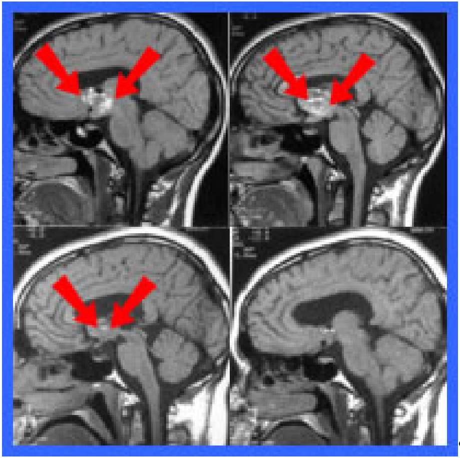
FEniCS-Apps
Rheagen

- Rheologies
  - Maxwell
  - Grade 2
  - Oldroyd-B
- Stabilization
  - DG
  - SUPG
  - EVSS
  - DEVSS
  - Macroelement
- Automation
  - FIAT (elements)
  - FFC (weak forms)



Real-time Surgery

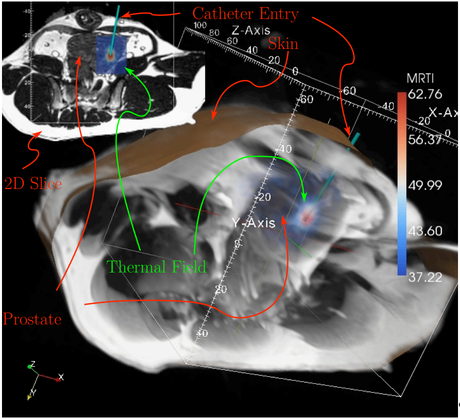
- Brain Surgery
  - Elastic deformation
  - Overlaid on MRI
  - Guides surgeon
- Laser Thermal Therapy
  - PDE constrained optimization
  - Per-patient calibration
  - Thermal inverse problem

[a]

[a] Warfield, Ferrant, et.al.

Real-time Surgery

- Brain Surgery
  - Elastic deformation
  - Overlaid on MRI
  - Guides surgeon
- Laser Thermal Therapy
  - PDE constrained optimization
  - Per-patient calibration
  - Thermal inverse problem



[a]

[a]Fuentes, Oden, et.al.

Outline

1 What can PETSc do?

2 Value of Design
- Portability
- Composability

Main Point

Separating the
Local from Global,

simplifies design,

and enables modern solvers.

Main Point

Separating the
Local from Global,

simplifies design,

and enables modern solvers.

Main Point

Separating the
Local from Global,

simplifies design,

and enables modern solvers.

Outline

2. Value of Design
   - Portability
   - Composability

Linear Algebra Interfaces

Strong interfaces mean:

- Easy code interoperability (LAPACK, Trilinos)
- Easy portability (GPU)
- Seamless optimization

VECCUDA

Strategy: Define a new **Vec** implementation

- Uses Thrust for data storage and operations on GPU
- Supports full PETSc **Vec** interface
- Inherits PETSc scalar type
- Can be activated at runtime, `-vec_type cuda`
- PETSc provides memory coherence mechanism

MATAIJCUDA

Also define new **Mat** implementations

- Uses Cusp for data storage and operations on GPU
- Supports full PETSc **Mat** interface, some ops on CPU
- Can be activated at runtime, `-mat_type aijcuda`
- Notice that parallel matvec necessitates off-GPU data transfer

Solvers

Solvers come for Free

Preliminary Implementation of PETSc Using GPU,

Minden, Smith, Knepley, 2010

- All linear algebra types work with solvers
- Entire solve can take place on the GPU
    - Only communicate scalars back to CPU
- GPU communication cost could be amortized over several solves
- Preconditioners are a problem
    - Cusp has a promising AMG

Example

Driven Cavity Velocity-Vorticity with Multigrid

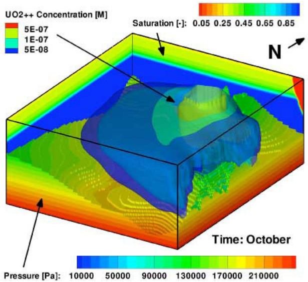
```
ex50 -da_vec_type seqcusp
  -da_mat_type aijcusp -mat_no_inode # Setup types
  -da_grid_x 100 -da_grid_y 100      # Set grid size
  -pc_type none -pc_mg_levels 1      # Setup solver
  -preload off -cuda_synchronize     # Setup run
  -log_summary
```

Example
PFLOTRAN

Flow Solver

$32 \times 32 \times 32$ grid

Routine	Time (s)	MFlops	MFlops/s
CPU			
KSPSolve	8.3167	4370	526
MatMult	1.5031	769	512
GPU			
KSPSolve	1.6382	4500	2745
MatMult	0.3554	830	2337



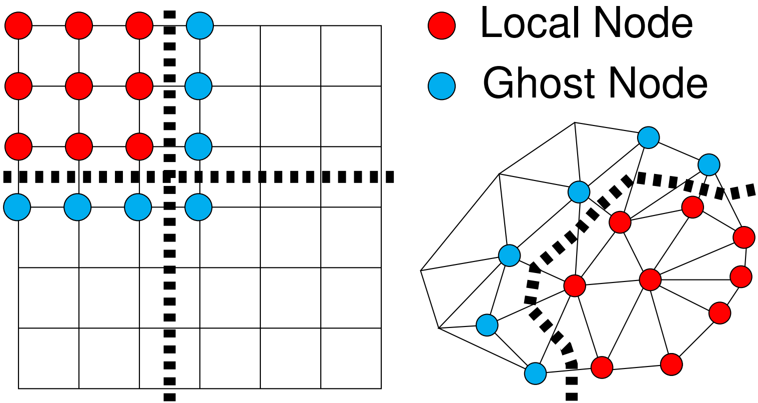
P. Lichtner, G. Hammond, R. Mills, B. Phillip

Outline

Ghost Values

To evaluate a local function $f(x)$, each process requires

- its local portion of the vector x
- its ghost values, bordering portions of x owned by neighboring processes



Residual Evaluation

The **DM** interface is based upon *local* callback functions

- FormFunctionLocal()
- FormJacobianLocal()

Callbacks are registered using

- SNESSetDM(), TSSetDM()
- DMSNESSetFunctionLocal(), DMTSSetJacobianLocal()

When PETSc needs to evaluate the nonlinear residual **F(x)**,

- Each process evaluates the local residual
- PETSc assembles the global residual automatically
  - Uses DMLocalToGlobal() method

Bratu Residual Evaluation

$$\Delta u + \lambda e^u = 0$$

```
ResLocal(DMDALocalInfo *info, PetscScalar **x, PetscScalar **f, void *ctx)
for(j = info->ys; j < info->ys+info->ym; ++j) {
  for(i = info->xs; i < info->xs+info->xm; ++i) {
    u = x[j][i];
    if (i==0 || j==0 || i == M || j == N) {
      f[j][i] = 2.0*(hydhx+hxdhy)*u; continue;
    }
    u_xx     = (2.0*u - x[j][i-1] - x[j][i+1])*hydhx;
    u_yy     = (2.0*u - x[j-1][i] - x[j+1][i])*hxdhy;
    f[j][i] = u_xx + u_yy - hx*hy*lambda*exp(u);
}}}
```

$PETSC_DIR/src/snes/examples/tutorials/ex5.c

Multigrid Paradigm

The **DM** interface uses the *local* callback functions to

- assemble global functions/operators from local pieces
- assemble functions/operators on coarse grids

Then **PCMG** organizes

- control flow for the multilevel solve, and
- projection and smoothing operators at each level.

Multigrid with DM

Allows multigrid with some simple command line options

- `-pc_type mg`, `-pc_mg_levels`
- `-pc_mg_type`, `-pc_mg_cycle_type`, `-pc_mg_galerkin`
- `-mg_levels_1_ksp_type`, `-mg_levels_1_pc_type`
- `-mg_coarse_ksp_type`, `-mg_coarse_pc_type`
- `-da_refine`, `-ksp_view`

Interface also works with GAMG and 3rd party packages like ML

MultiPhysics Paradigm

The **PCFieldSplit** interface

- extracts functions/operators corresponding to each physics
  - **VecScatter** and `MatGetSubMatrix()` for efficiency
- assemble functions/operators over all physics
  - Generalizes `LocalToGlobal()` mapping
- is composable with ANY PETSc solver and preconditioner
  - This can be done recursively

MultiPhysics Paradigm

The **PCFieldSplit** interface

- extracts functions/operators corresponding to each physics
  - **VecScatter** and `MatGetSubMatrix()` for efficiency
- assemble functions/operators over all physics
  - Generalizes `LocalToGlobal()` mapping
- is composable with ANY PETSc solver and preconditioner
  - This can be done recursively

FieldSplit provides the buildings blocks
for multiphysics preconditioning.

MultiPhysics Paradigm

The **PCFieldSplit** interface

- extracts functions/operators corresponding to each physics
  - **VecScatter** and `MatGetSubMatrix()` for efficiency
- assemble functions/operators over all physics
  - Generalizes `LocalToGlobal()` mapping
- is composable with ANY PETSc solver and preconditioner
  - This can be done recursively

Notice that this works in exactly the same manner as

- multiple resolutions (MG, FMM, Wavelets)
- multiple domains (Domain Decomposition)
- multiple dimensions (ADI)

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Block-Jacobi (Exact)

```
-ksp_type gmres -pc_type fieldsplit -pc_fieldsplit_type additive
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type lu
 -fieldsplit_pressure_ksp_type preonly -fieldsplit_pressure_pc_type jacobi
```

$$\begin{pmatrix} A & 0 \\ 0 & I \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Block-Jacobi (Inexact)

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type additive
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type gamg
 -fieldsplit_pressure_ksp_type preonly -fieldsplit_pressure_pc_type jacobi
```

$$\begin{pmatrix} \hat{A} & 0 \\ 0 & I \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Gauss-Seidel (Inexact)

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type multiplicative
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type gamg
 -fieldsplit_pressure_ksp_type preonly -fieldsplit_pressure_pc_type jacobi
```

$$\begin{pmatrix} \hat{A} & B \\ 0 & I \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Gauss-Seidel (Inexact)

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type multiplicative
 -pc_fieldsplit_0_fields 1 -pc_fieldsplit_1_fields 0
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type gamg
 -fieldsplit_pressure_ksp_type preonly -fieldsplit_pressure_pc_type jacobi
```

$$\begin{pmatrix} I & B^T \\ 0 & \hat{A} \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Diagonal Schur Complement

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type diag
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type gamg
 -fieldsplit_pressure_ksp_type minres  -fieldsplit_pressure_pc_type none
```

$$\begin{pmatrix} \hat{A} & 0 \\ 0 & -\hat{S} \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Lower Schur Complement

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type lower
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type gamg
 -fieldsplit_pressure_ksp_type minres  -fieldsplit_pressure_pc_type none
```

$$\begin{pmatrix} \hat{A} & 0 \\ B^T & \hat{S} \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Upper Schur Complement

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type upper
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type gamg
 -fieldsplit_pressure_ksp_type minres  -fieldsplit_pressure_pc_type none
```

$$\begin{pmatrix} \hat{A} & B \\ & \hat{S} \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Uzawa

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type upper
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type lu
 -fieldsplit_pressure_ksp_type richardson
 -fieldsplit_pressure_ksp_max_its 1
```

$$\begin{pmatrix} A & B \\ & \hat{S} \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Full Schur Complement

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type full
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type lu
 -fieldsplit_pressure_ksp_rtol 1e-10 -fieldsplit_pressure_pc_type jacobi
```

$$\begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

SIMPLE

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type full
 -fieldsplit_velocity_ksp_type preonly -fieldsplit_velocity_pc_type lu
 -fieldsplit_pressure_ksp_rtol 1e-10 -fieldsplit_pressure_pc_type jacobi
 -fieldsplit_pressure_inner_ksp_type preonly
  -fieldsplit_pressure_inner_pc_type jacobi
 -fieldsplit_pressure_upper_ksp_type preonly
  -fieldsplit_pressure_upper_pc_type jacobi
```

$$\begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & B^T D_A^{-1} B \end{pmatrix} \begin{pmatrix} I & D_A^{-1} B \\ 0 & I \end{pmatrix}$$

Solver Configuration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Least-Squares Commutator

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type full
-pc_fieldsplit_schur_precondition self
 -fieldsplit_velocity_ksp_type gmres -fieldsplit_velocity_pc_type lu
 -fieldsplit_pressure_ksp_rtol 1e-5 -fieldsplit_pressure_pc_type lsc
```

$$\begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & \hat{S}_{\mathrm{LSC}} \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

Solver Configuration: No New Code

ex31: P_2/P_1 Stokes Problem with Temperature on Unstructured Mesh

Additive Schwarz + Full Schur Complement

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type additive
-pc_fieldsplit_0_fields 0,1 -pc_fieldsplit_1_fields 2
 -fieldsplit_0_ksp_type fgmres -fieldsplit_0_pc_type fieldsplit
 -fieldsplit_0_pc_fieldsplit_type schur
 -fieldsplit_0_pc_fieldsplit_schur_factorization_type full
   -fieldsplit_0_fieldsplit_velocity_ksp_type preonly
   -fieldsplit_0_fieldsplit_velocity_pc_type lu
   -fieldsplit_0_fieldsplit_pressure_ksp_rtol 1e-10
   -fieldsplit_0_fieldsplit_pressure_pc_type jacobi
 -fieldsplit_temperature_ksp_type preonly
 -fieldsplit_temperature_pc_type lu
```

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} \hat{A} & 0 \\ 0 & \hat{S} \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix} & 0 \\ 0 & L_T \end{pmatrix}$$

Solver Configuration: No New Code

ex31: P_2/P_1 Stokes Problem with Temperature on Unstructured Mesh

Upper Schur Comp. + Full Schur Comp. + Least-Squares Comm.

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_0_fields 0,1 -pc_fieldsplit_1_fields 2
-pc_fieldsplit_schur_factorization_type upper
 -fieldsplit_0_ksp_type fgmres -fieldsplit_0_pc_type fieldsplit
 -fieldsplit_0_pc_fieldsplit_type schur
 -fieldsplit_0_pc_fieldsplit_schur_factorization_type full
   -fieldsplit_0_fieldsplit_velocity_ksp_type preonly
   -fieldsplit_0_fieldsplit_velocity_pc_type lu
   -fieldsplit_0_fieldsplit_pressure_ksp_rtol 1e-10
   -fieldsplit_0_fieldsplit_pressure_pc_type jacobi
 -fieldsplit_temperature_ksp_type gmres
 -fieldsplit_temperature_pc_type lsc
```

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} \hat{A} & 0 \\ 0 & \hat{S} \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix} & G \\ 0 & \hat{S}_{\text{LSC}} \end{pmatrix}$$

Conclusion

With good code design, PETSc can make Linear Algebra

Conclusion

With good code design, PETSc can make
Solving Equations

Conclusion

With good code design, PETSc can make Mesh and Data Management

Conclusion

With good code design, PETSc can make Multigrid

Conclusion

With good code design, PETSc can make Multiphysics

Conclusion

With good code design, PETSc can make FMM

Conclusion

With good code design, PETSc can make Scientific Computing