Interfacing MFiX with PETSc and HYPRE Linear Solver libraries

PI: Gautham Krishnamoorthy (UND)
Lauren Clarke (Grad Student, UND)

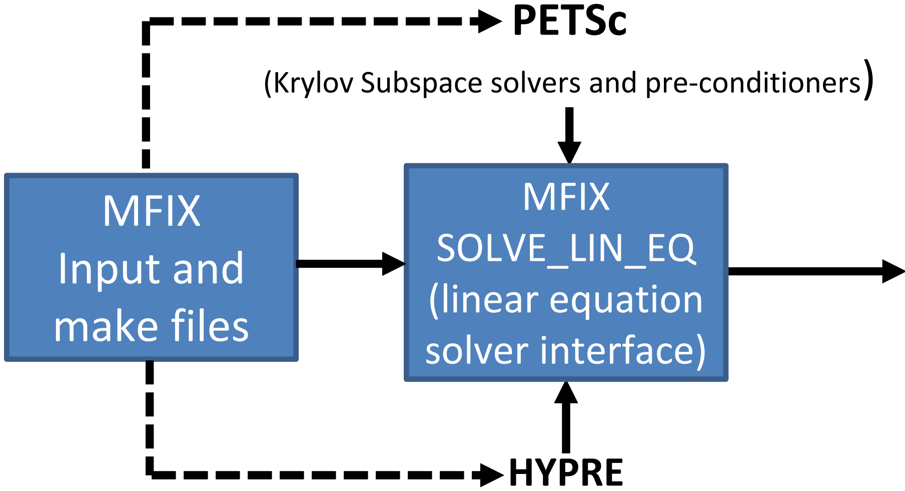
Co-PI: Jeremy Thornock (U.Utah)

March 23rd , 2017

Acknowledgement: This research is being funded by the University Coal Research Program which is administered by DOE-NETL (Award #: DE-FE0026191) PM: Jason Hissam

Objective/Vision

- Build a robust, well-abstracted, interface to the PETSc, HYPRE linear solver libraries from MFiX



PETSc

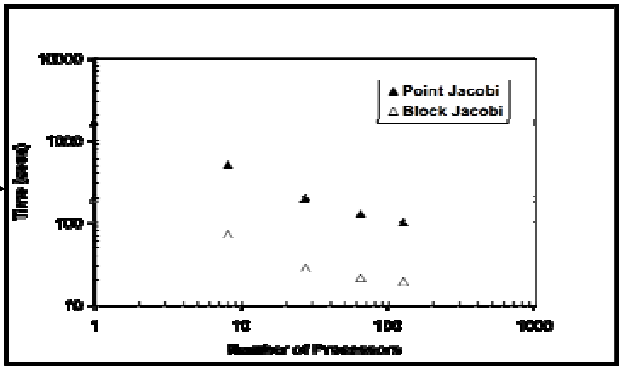
(Krylov Subspace solvers and pre-conditioners)

MFIX Input and make files

MFIX SOLVE_LIN_EQ (linear equation solver interface)

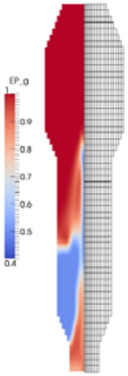
HYPRE

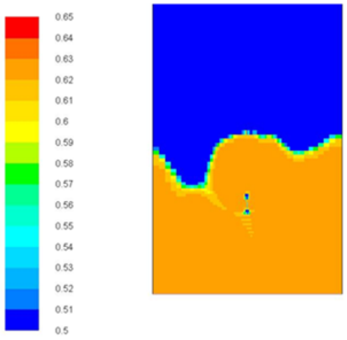
(Multigrid and Krylov solvers and pre-conditioners)

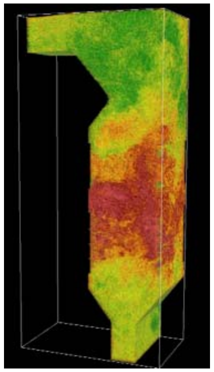


Improved speed and scaling performance

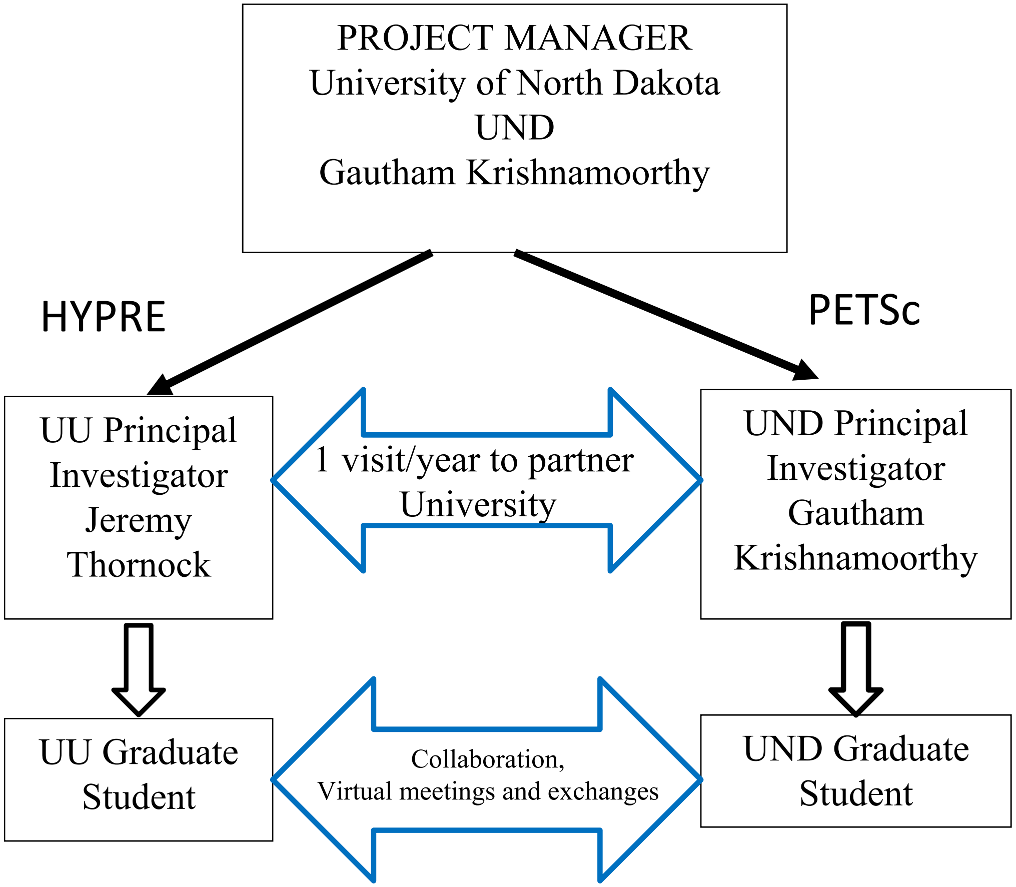
- Code verification, Documentation and Best Practices for established MFiX solutions







Team Description



PROJECT MANAGER
University of North Dakota
UND
Gautham Krishnamoorthy

HYPRE

PETSc

UU Principal Investigator
Jeremy Thornock

1 visit/year to partner University

UND Principal Investigator
Gautham Krishnamoorthy

UU Graduate Student

Collaboration,
Virtual meetings and exchanges

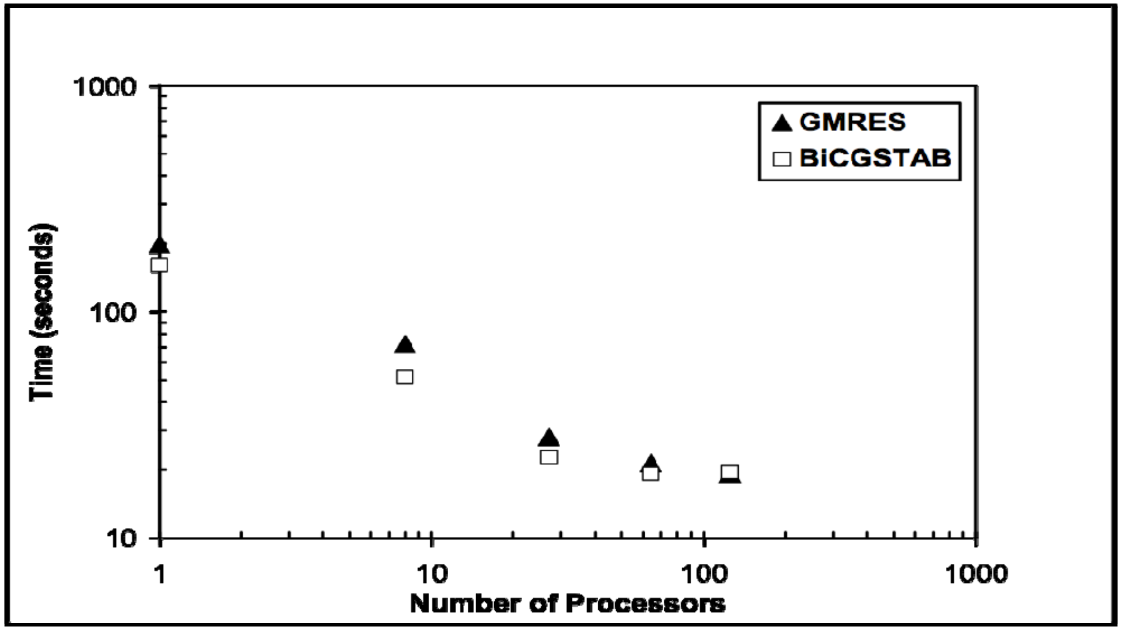
UND Graduate Student

UND

THE UNIVERSITY OF UTAH

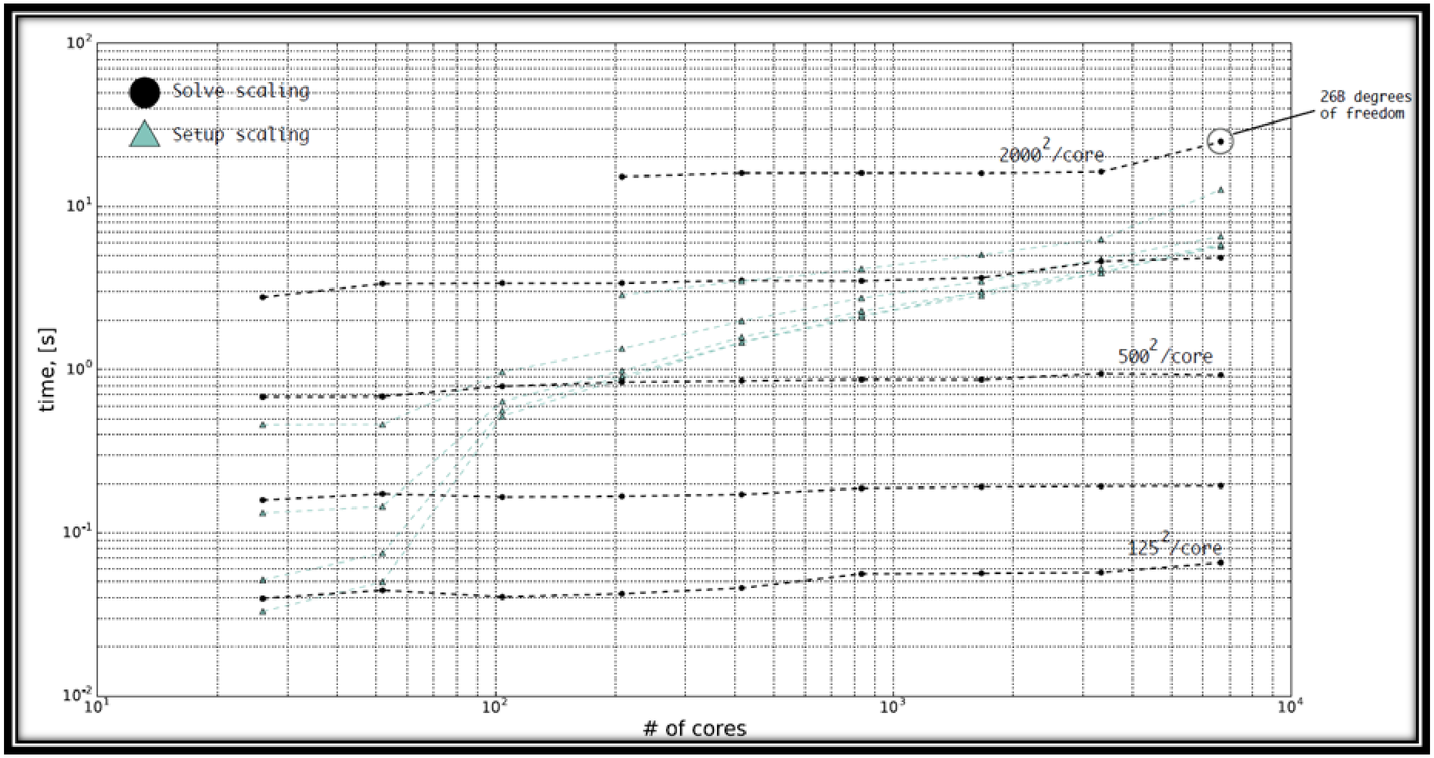
Background
PETSc

- PETSc (ANL) is a linear solver package for the solution of preconditioned, sparse linear systems (KSP)
- PETSc includes native support for Fortran codes (MFIX)
- U. Utah and UND have extensive experience using PETSc (non-symmetric matrices resulting from the discrete ordinates radiation model)



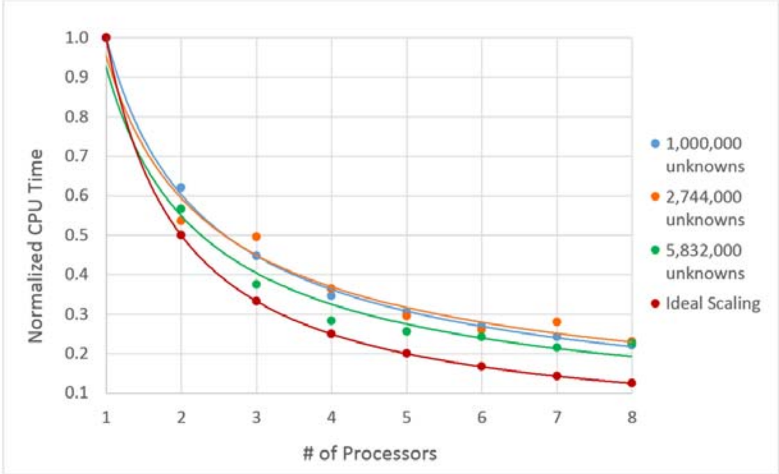
Background
Hypre

- Hypre (LLNL) is a linear solver package for the solution of preconditioned, sparse linear systems (including multigrid)
- Hypre includes native support for Fortran codes (MFIX)
- U.Utah and UND have extensive experience using Hypre for septa-diagonal symmetric matrix systems (Pressure-Poisson and P-1 radiation model)

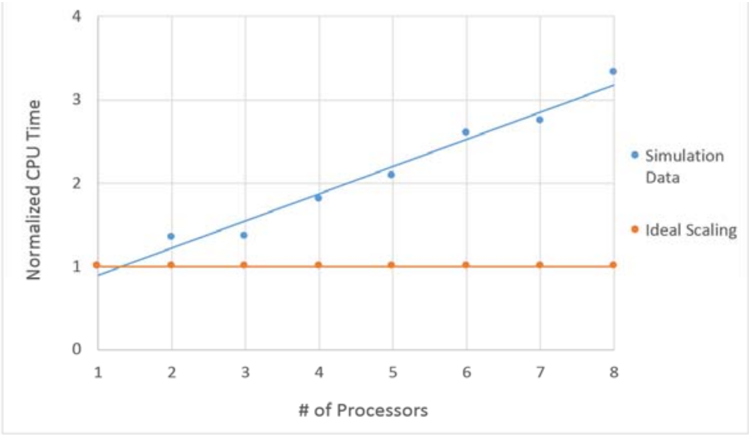


The Problem

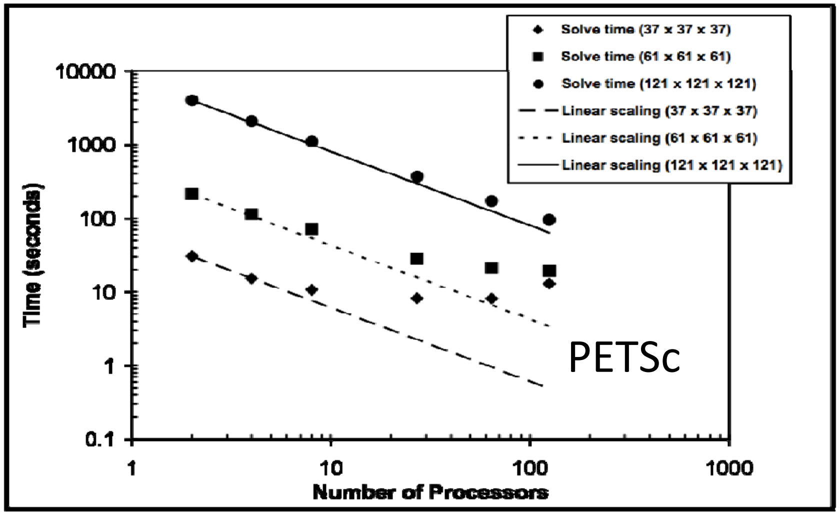
(Achieving good scaling of MFIX when invoking PETSc and HYPRE)

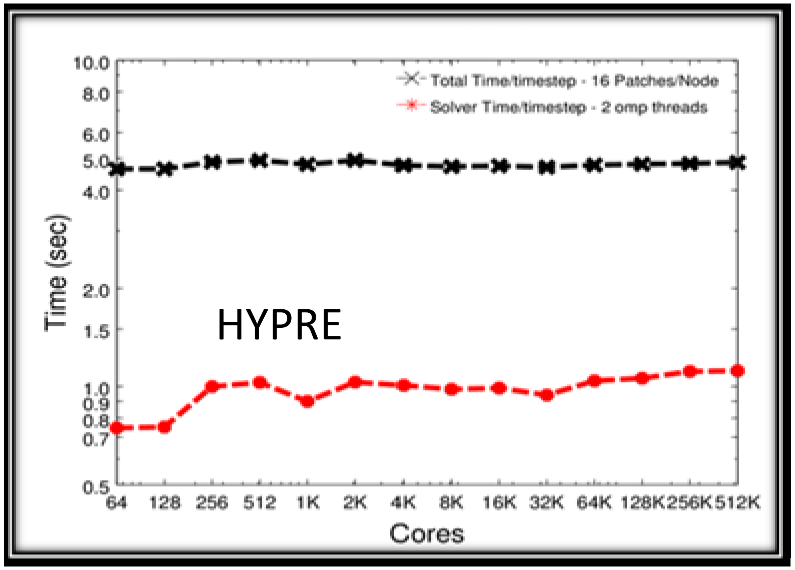


A plot of the normalized CPU time vs. the number of processors for three different matrix sizes (**fixed problem size efficiencies**)



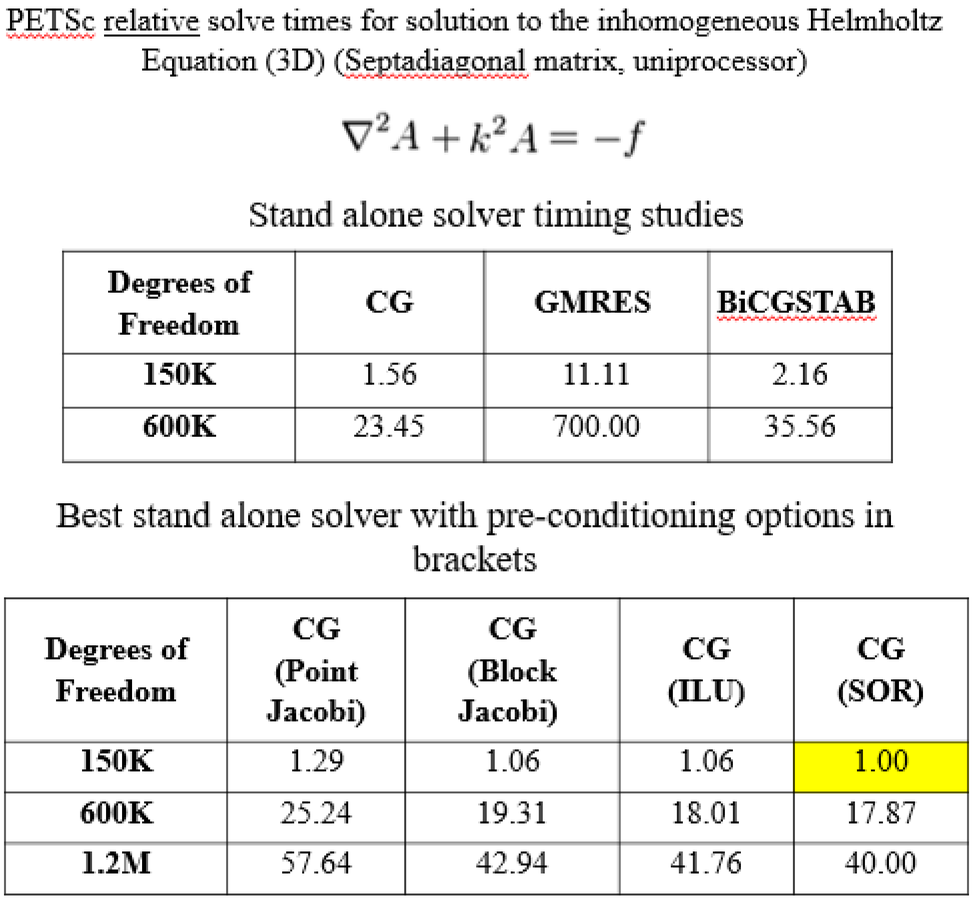
A plot of the normalized CPU time vs. the number of processors for a **scaled problem size**. The number of unknowns per processor was kept constant at 1 Million.





The Problem

(Identification of optimum solvers and pre-conditioners)



PETSc relative solve times for solution to the inhomogeneous Helmholtz Equation (3D) (Septadiagonal matrix, uniprocessor)

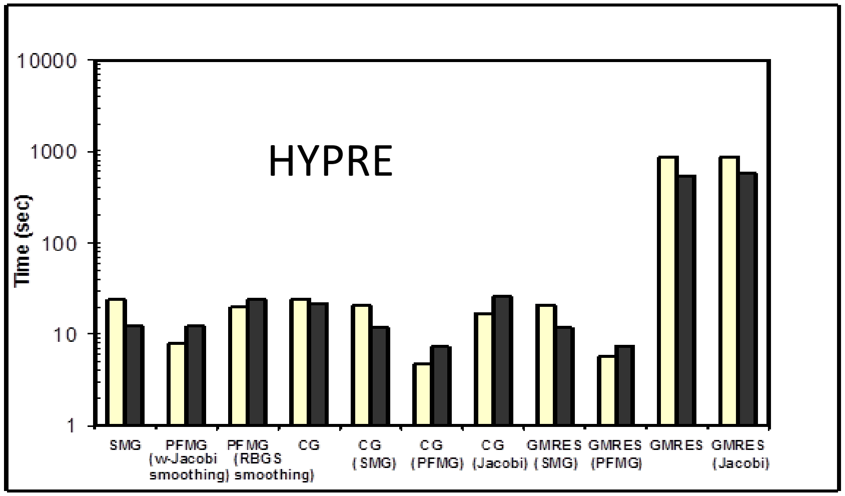
$$\nabla^2 A + k^2 A = -f$$

Stand alone solver timing studies

Degrees of Freedom	CG	GMRES	BiCGSTAB
150K	1.56	11.11	2.16
600K	23.45	700.00	35.56

Best stand alone solver with pre-conditioning options in brackets

Degrees of Freedom	CG (Point Jacobi)	CG (Block Jacobi)	CG (ILU)	CG (SOR)
150K	1.29	1.06	1.06	1.00
600K	25.24	19.31	18.01	17.87
1.2M	57.64	42.94	41.76	40.00



Background
Software Abstraction

The problem:

- MFiX already has linear solver options
- Interfacing with the linear solver packages is not universal (different stencil setup operations)
- Fortran (MFIX is written in F90) isn't an object-oriented programming language

Our approach:

- Programmers and users look for a user-friendly linear solver interface
- Operations to setup a general linear solve (Ax=b), is easily abstracted
  - Compute matrix and vector elements (local to global mapping in PETSc and HYPRE)
- Define a common interface and derive specific solver interface for existing MFiX solvers, HYPRE, and PETSC, etc.
- Documentation!

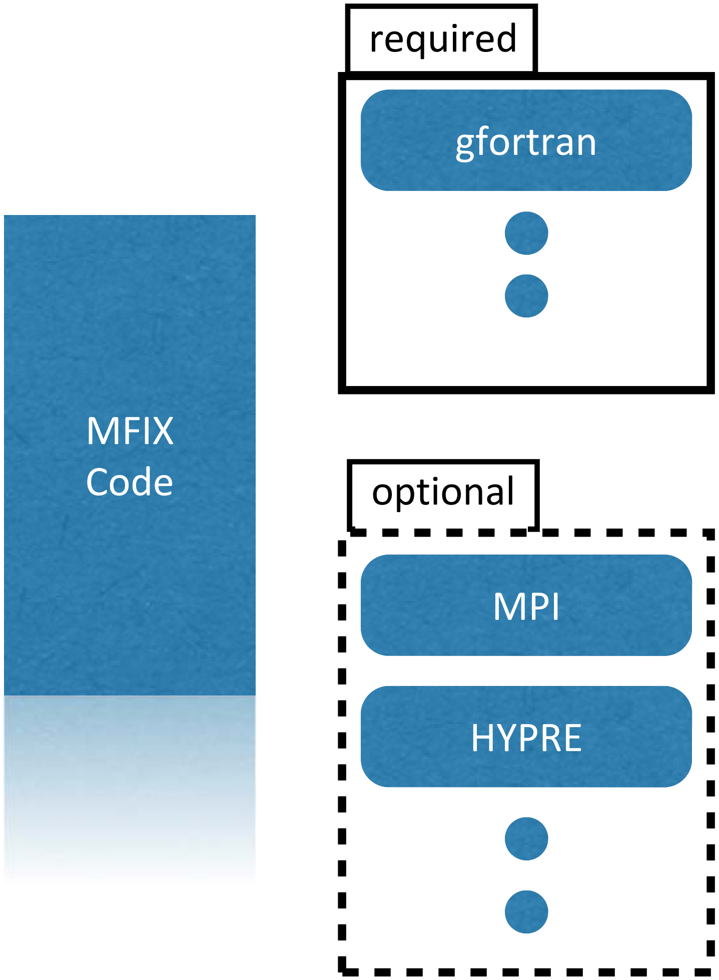
Software Tasks: Interfacing MFiX with PETSc and HYPRE

- Problem Setup*: Solver parameters (solver tolerances, maximum number of iterations, solver types, pre-conditioners etc...)
- Solver Setup*: Solver object creation (allocation of A, x, and b) and initialization methods.
- Communication Linear System: Handshake (or "mapping") function for passing the linear system coefficients (A) and right-hand-side values (b) in the current native MFIX data-structure to the solver-specific types.
- Solve System: Compute the solution (x) to the linear system
- Return/Copy Solution: Conversion of the solver type solution (x) to the current, native MFiX type
- Cleanup: De-allocation and destruction of solver objects

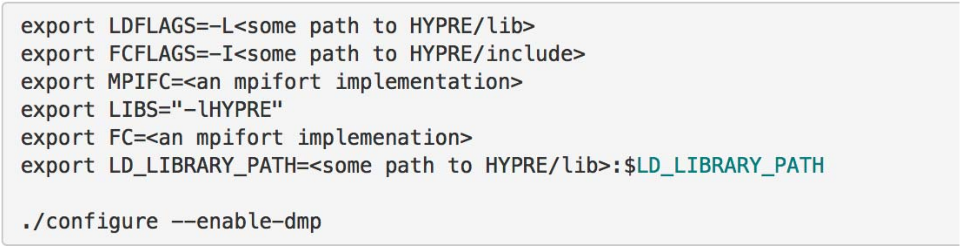
*one-time costs during simulation start-up during a transient calculation

Build Abstraction:

Add HYPRE as a non-disruptive, configure-time option



1) Set these env flags:



```
export LDFLAGS=-L<some path to HYPRE/lib>
export FCFLAGS=-I<some path to HYPRE/include>
export MPIFC=<an mpifort implementation>
export LIBS="-lHYPRE"
export FC=<an mpifort implemenation>
export LD_LIBRARY_PATH=<some path to HYPRE/lib>:$LD_LIBRARY_PATH

./configure --enable-dmp
```

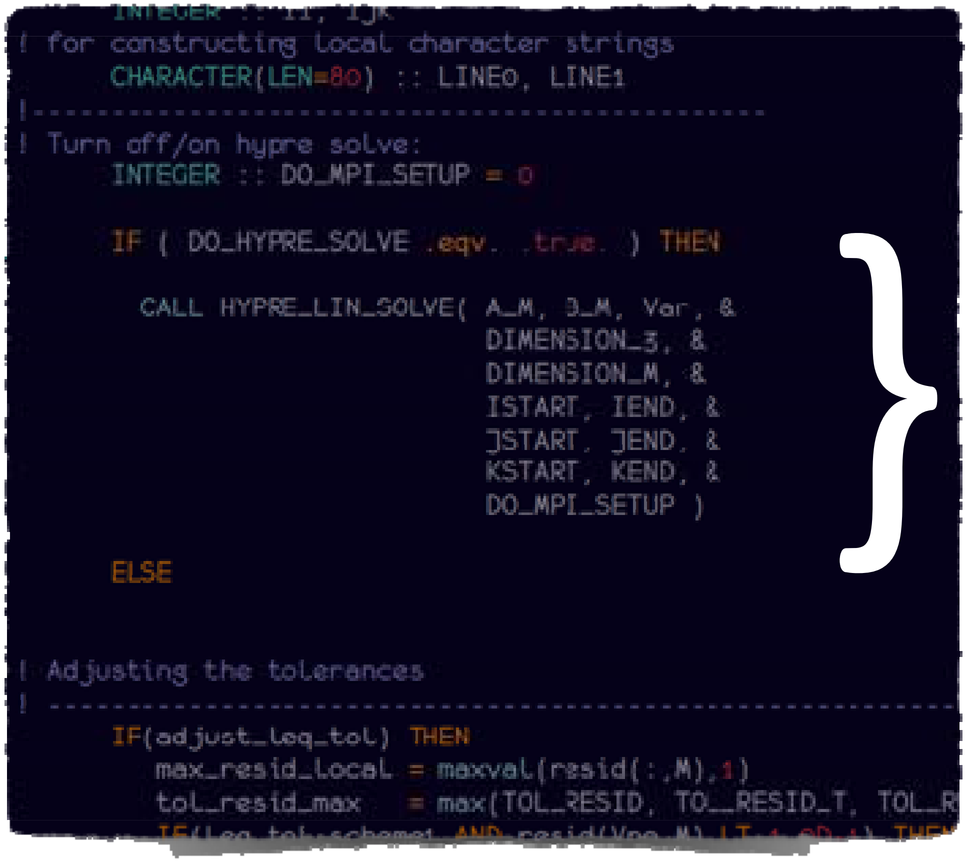
2) Edit Makefile† to add `hypre*.f`

3) `make`

4) Hypre is enabled in `solve_lin_eq.f`

Code Abstraction

`mfix/model/solve_lin_eq.f`

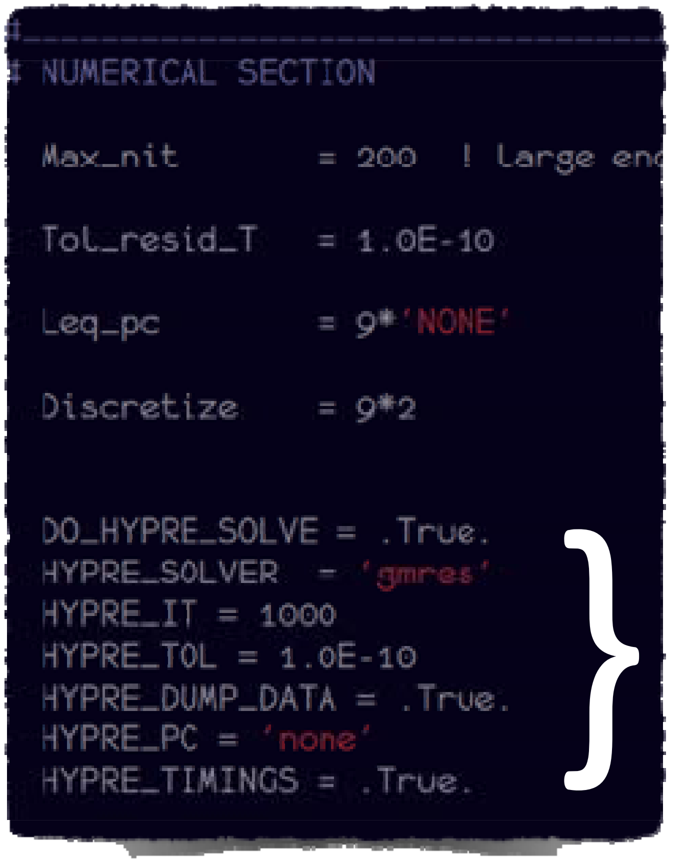


```
      INTEGER :: ii, ijk
! for constructing local character strings
      CHARACTER(LEN=80) :: LINE0, LINE1
!-----------------------------------------------
! Turn off/on hypre solve:
      INTEGER :: DO_MPI_SETUP = 0

      IF ( DO_HYPRE_SOLVE .eqv. .true. ) THEN

        CALL HYPRE_LIN_SOLVE( A_M, B_M, Var, &
                              DIMENSION_3, &
                              DIMENSION_M, &
                              ISTART, IEND, &
                              JSTART, JEND, &
                              KSTART, KEND, &
                              DO_MPI_SETUP )

      ELSE


! Adjusting the tolerances
! -----------------------------------------------------------------
      IF(adjust_leq_tol) THEN
         max_resid_local = maxval(resid(:,M),1)
         tol_resid_max   = max(TOL_RESID, TOL_RESID_T, TOL_R
         IF(leq_tol_scheme1.AND.resid(Var,M).LT.1.0D-1) THEN
```

- Add a "USE hypreUtilities"
- Option to call the HYPRE Solve added to `solve_lin_eq.f`
- Added two additional fortran modules (hypreUtilities, hypreParameters)
- Passing raw MFiX data directly into HYPRE_LIN_SOLVE which takes care of the rest (mapping, solve, etc)

Runtime Abstraction

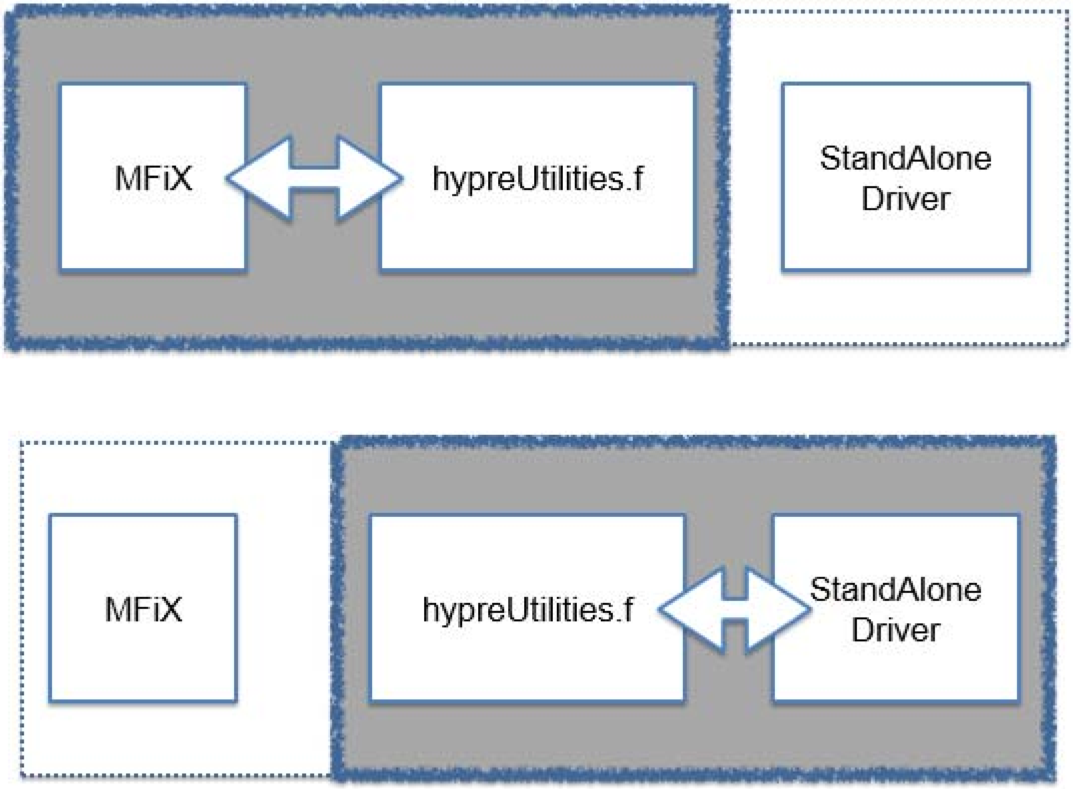
mfix.dat file:



```
#------------------------------------------
# NUMERICAL SECTION

  Max_nit          = 200   ! Large en

  Tol_resid_T      = 1.0E-10

  Leq_pc           = 9*'NONE'

  Discretize       = 9*2


  DO_HYPRE_SOLVE = .True.
  HYPRE_SOLVER  = 'gmres'
  HYPRE_IT = 1000
  HYPRE_TOL = 1.0E-10
  HYPRE_DUMP_DATA = .True.
  HYPRE_PC = 'none'
  HYPRE_TIMINGS = .True.
```

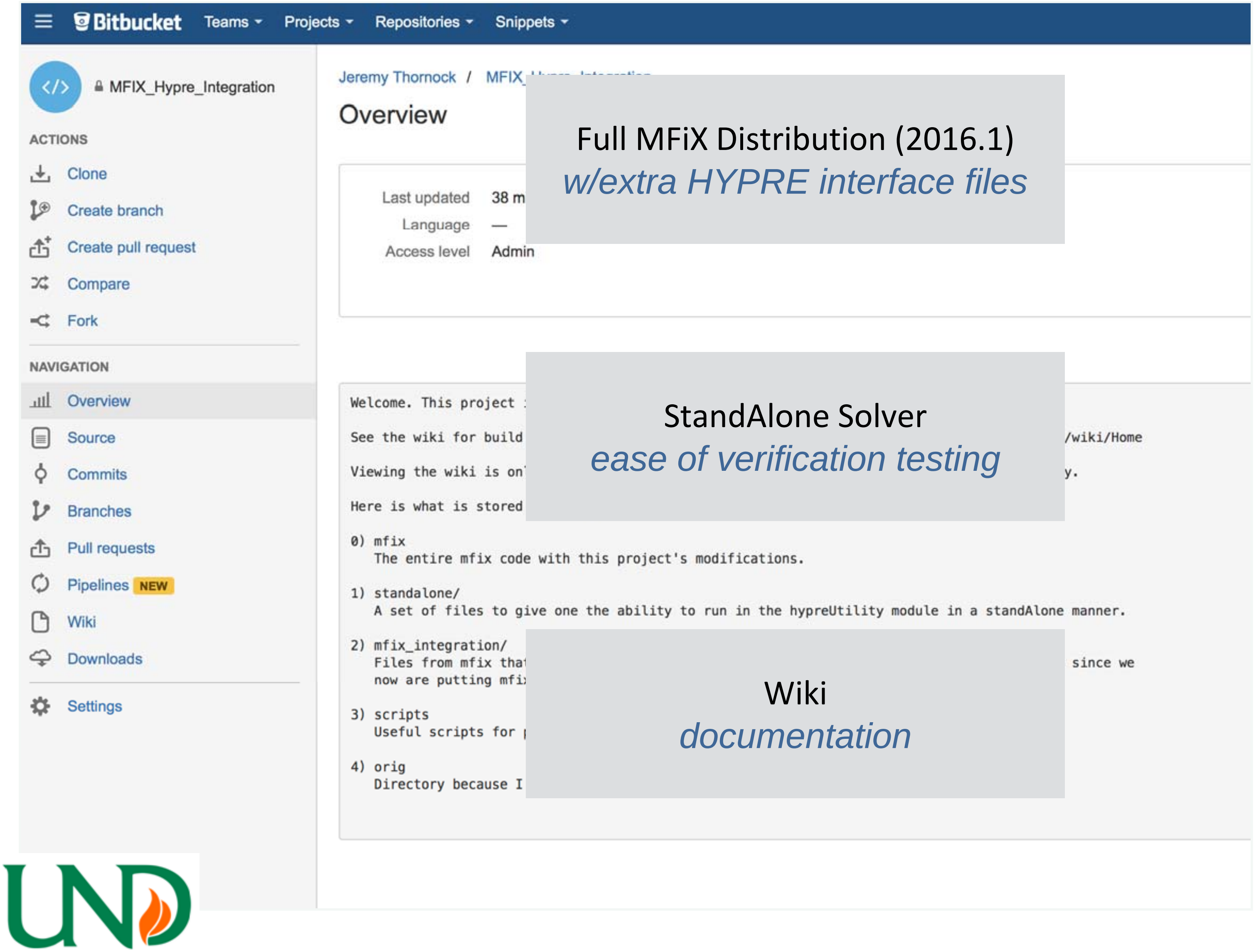
- HYPRE parameters controlled through the input file
- Inputs documented in the bitbucket project Wiki

Standalone Testbed Framework



MFiX/HYPRE Bitbucket Project

Private git repository with full MFiX (2016.1) dist. (see Thornock for access)



Last updated 38 m
Language —
Access level Admin

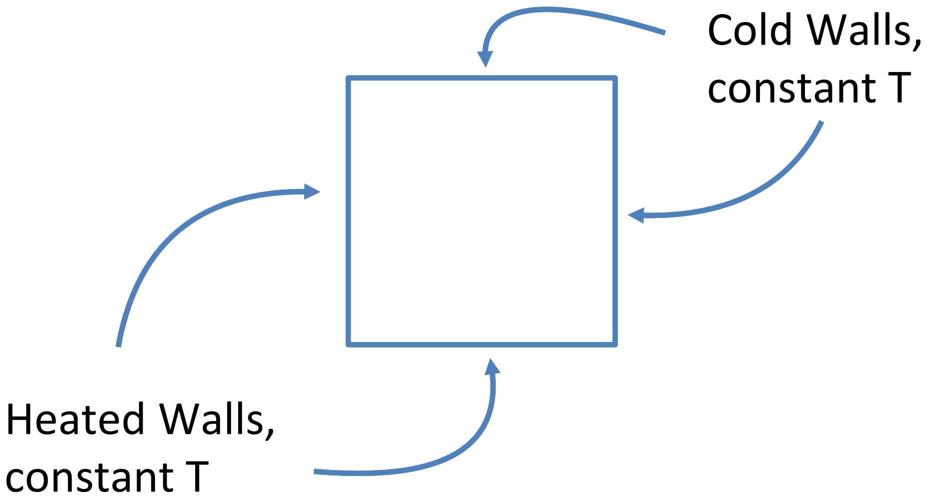
```
Welcome. This project
See the wiki for build                                                          /wiki/Home
Viewing the wiki is on                                                          y.
Here is what is stored
0) mfix
   The entire mfix code with this project's modifications.
1) standalone/
   A set of files to give one the ability to run in the hypreUtility module in a standAlone manner.
2) mfix_integration/
   Files from mfix that                                                          since we
   now are putting mfi
3) scripts
   Useful scripts for
4) orig
   Directory because I
```

Full MFiX Distribution (2016.1)
w/extra HYPRE interface files

StandAlone Solver
ease of verification testing

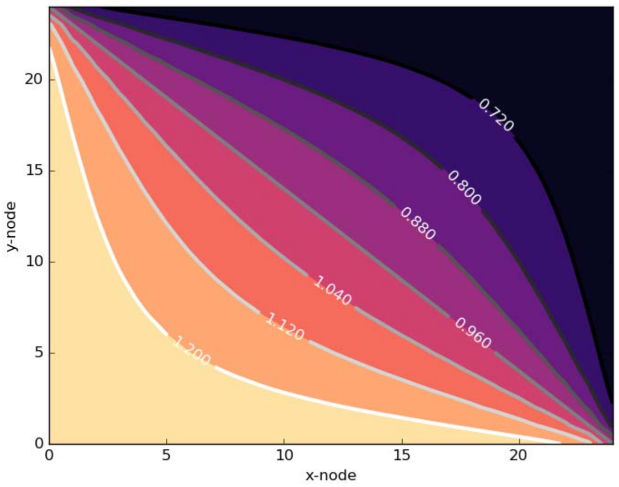
Wiki
documentation

Example: Standalone testing (verification) on heat eqn.



Cold Walls, constant T

Heated Walls, constant T



(solution example)

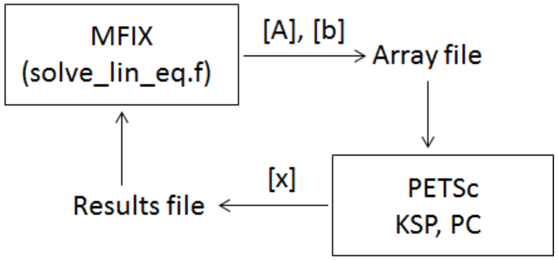
$$k\nabla\,.^{2}\,T = 0$$ Heat equation

$$Ax = b$$ Linearized

A : stencil coefficients

x : T (unknowns)

b : boundary conditions



MFIX (solve_lin_eq.f)

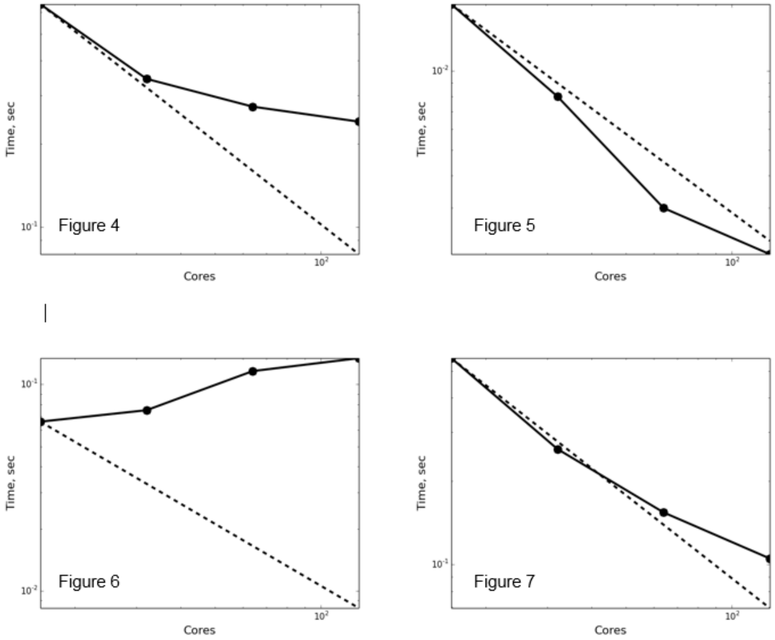
[A], [b]

Array file

PETSc KSP, PC

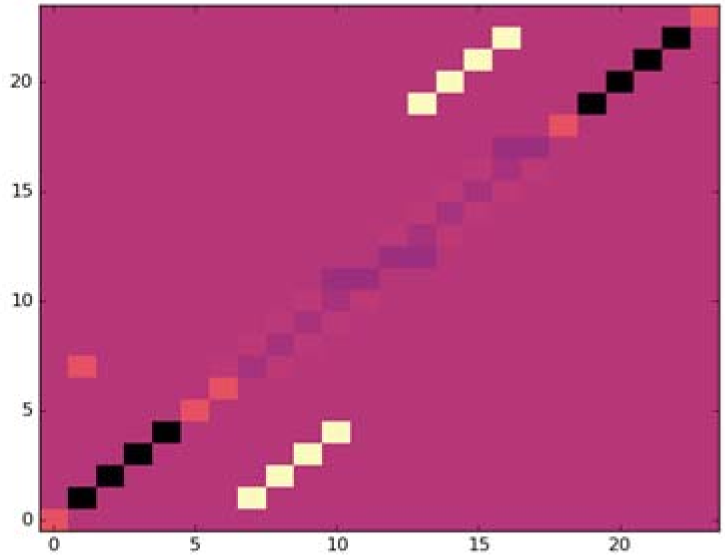
[x]

Results file

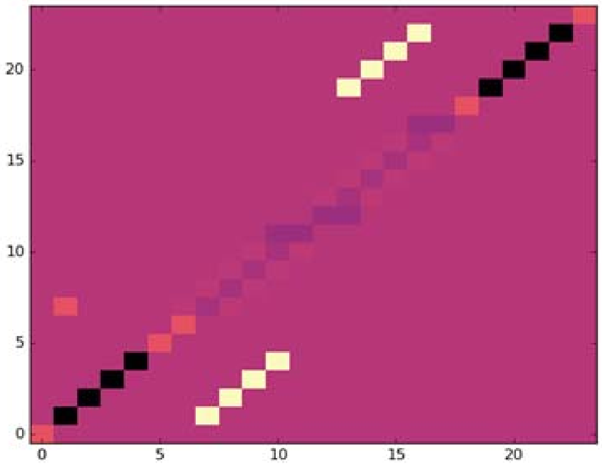


Figures 4 - 7: Plots of the (2) Total overall scaling (3) Matrix and vector object construction scaling (4) Solver object scaling and (5) BiCGStab scaling with an SMG preconditioner.

Asymmetry and the use of Conjugate Gradients

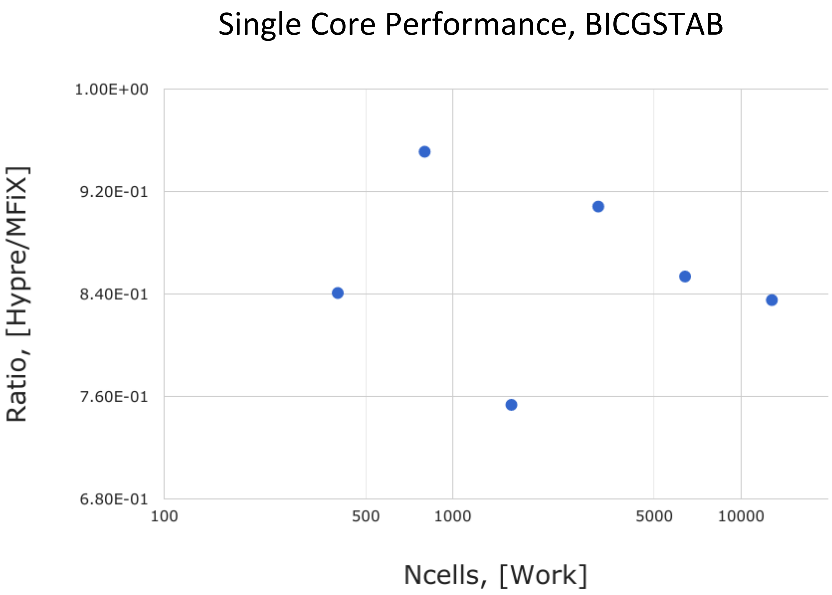


MFiX with HYPRE Single Core Performance

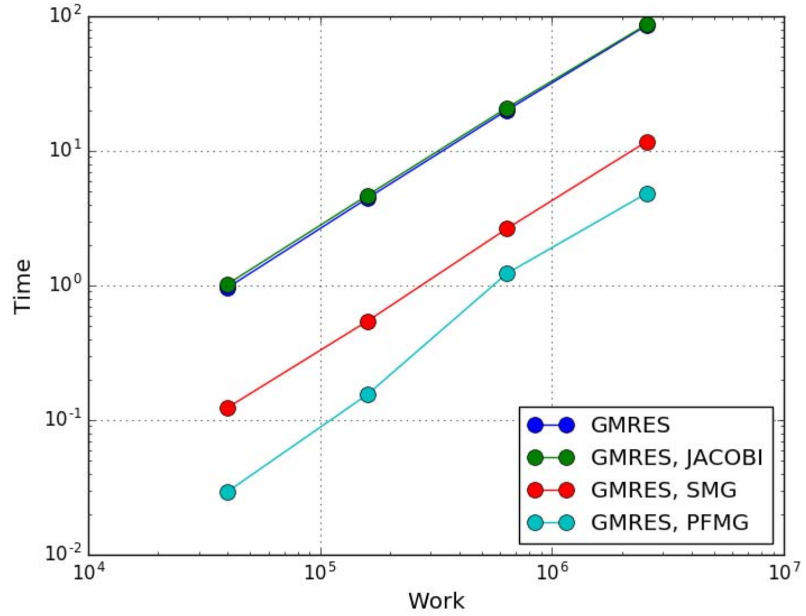


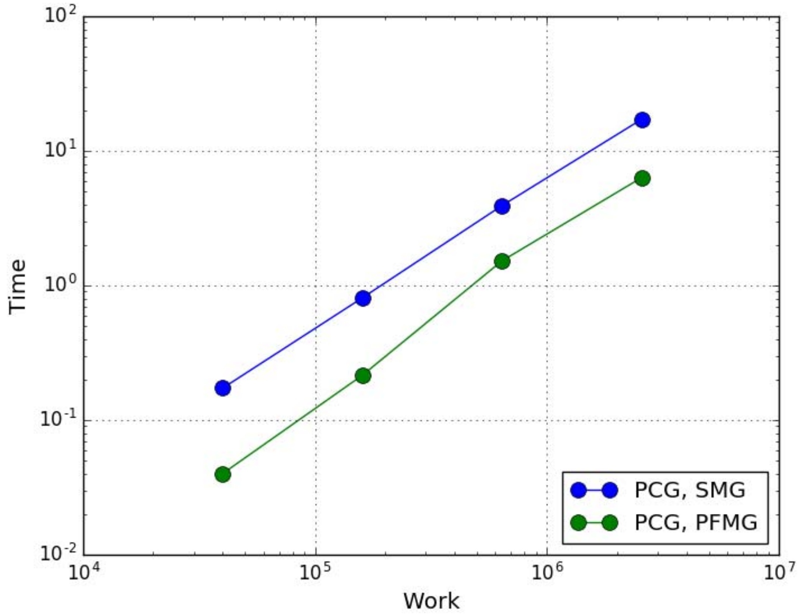
visualization of matrix coefficients

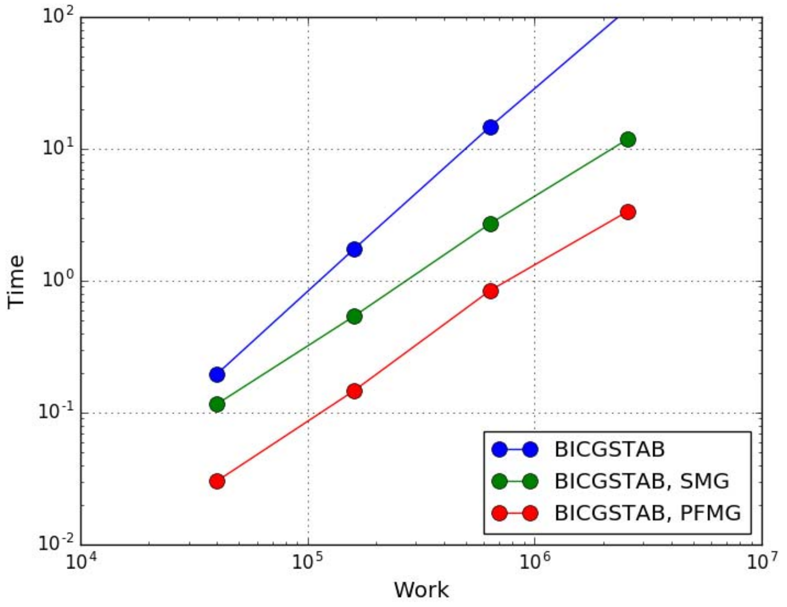
- The HYPRE-MFiX coupling was tested using TFM02 (heat eqn)
- Asymmetry shown in A matrix (left Figure)
- Asymmetry precludes CG solver (and some multigrid)
- GMRES and BICGSTAB tested for single-core efficiency
- No preconditioner
- Efficiency measured against MFiX solve



Standalone Solver Performance (single core) vs. Work Load



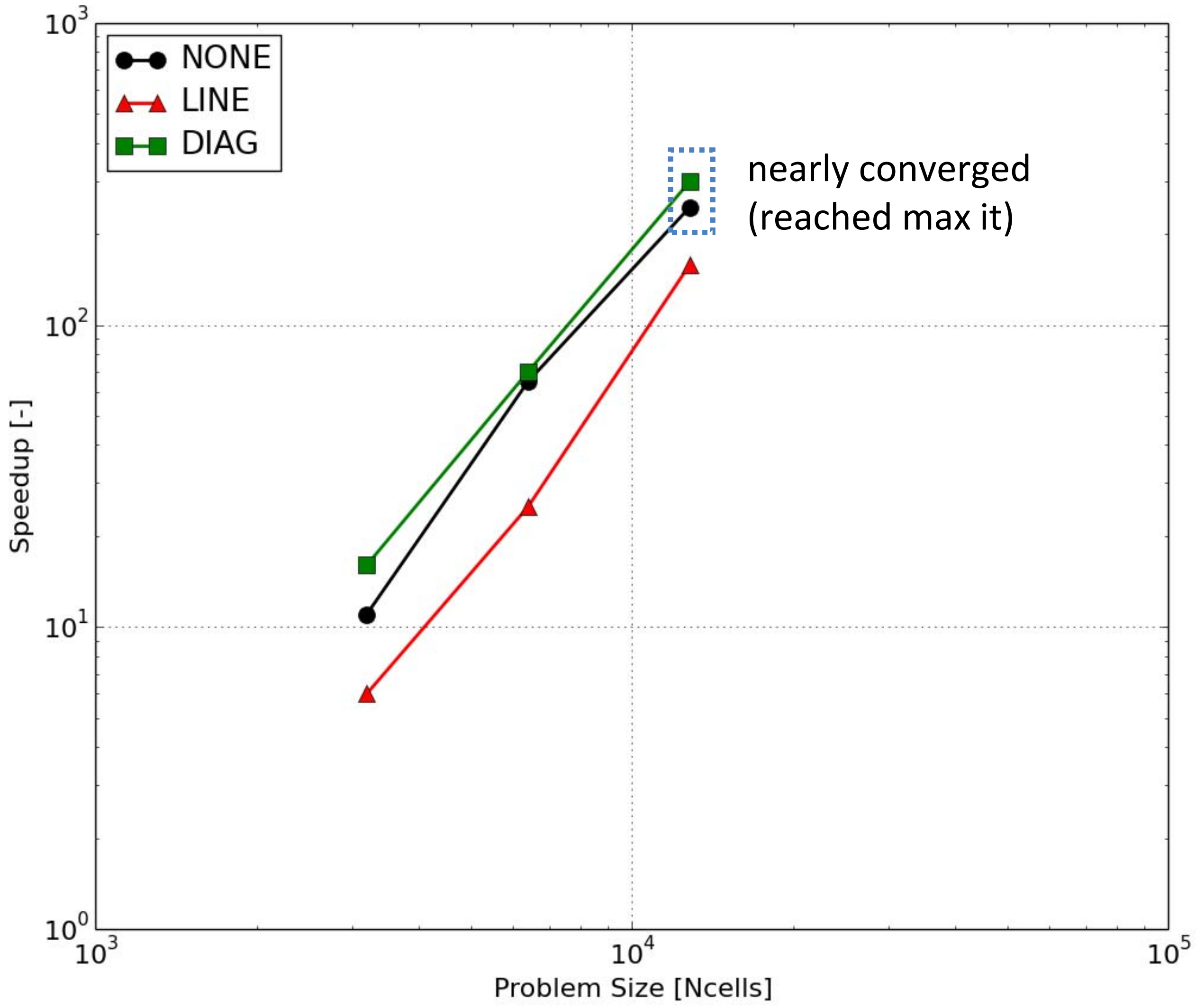




- System was symmetric (Laplace eqn)
- All solvers verified
- Preconditioners have a significant effect (as expected)
- Multigrid/CG solvers need symmetric systems

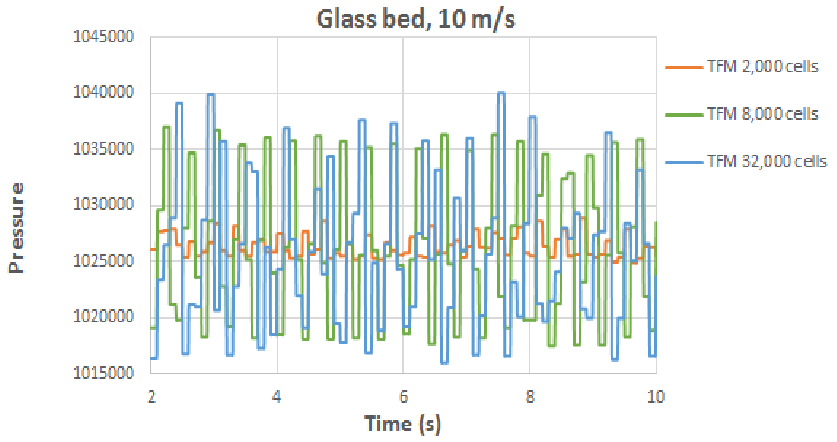
Multigrid-SMG Preconditioner Speedup (BiCGStab)

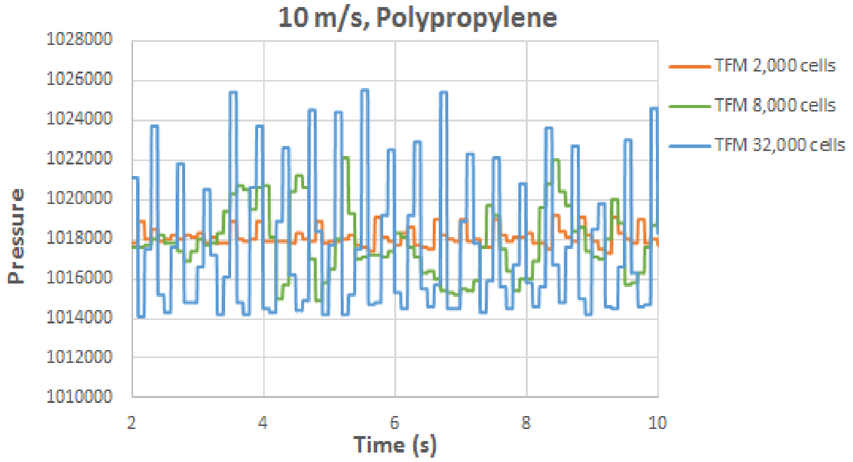
speedup = (mfix solve time)/(hypre solve time)

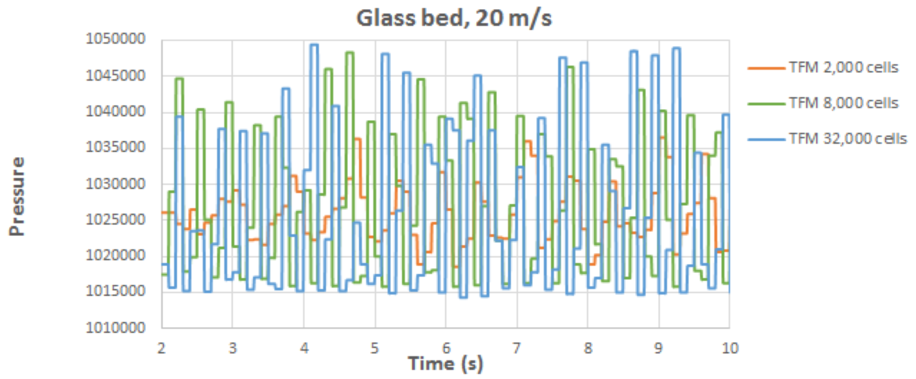


nearly converged
(reached max it)

Resolution and Computational Time are Application Specific







	2,000 Cells		8,000 Cells		32,000 Cells	
	Simulation Time	Computational Time	Simulation Time	Computational Time	Simulation Time	Computational Time
Glass 5 m/s	10.0 s	2 h	10.0 s	N/A	10.0 s	N/A
Glass 10 m/s	10.0 s	25 min	10.0 s	1.9 h	10.0 s	2.6 h
Glass 20 m/s	10.0 s	31 min	10.0 s	1.5 h	10.0 s	9.7 h
Polypropylene 5 m/s	10.0 s	33 min	10.0 s	1.0 h	10.0 s	5.0 h
Polypropylene 10 m/s	10.0 s	13 min	10.0 s	1.6 h	10.0 s	N/A
Polypropylene 20 m/s	10.0 s	8 min	10.0 s	50 min	10.0 s	N/A

Accomplishments and Next Steps…..

- Krishnamoorthy, Gautham. "A Computationally Efficient P 1 Radiation Model for Modern Combustion Systems Utilizing Pre-Conditioned Conjugate Gradient Methods." Applied Thermal Engineering (2017).
- Closer investigation of matrix asymmetry
- Attach specific solvers/equation (e.g., pressure solve to use PCG while u* uses GMRES)
- Extend the test problem suite (have already tested other TFM* problems with success)
- Handle secondary phases, not just the gas
- User Friendly: Interface - fairly clean (Fortran Module) but could be polished
- Resolve build system issues to ease configure/build process and documentation (User Friendly!)