Interfacing MFIX with PETSC and HYPRE Linear Solver Libraries

Jeremy Thornock, U. of Utah  
Surya Yamujala, U. of Utah  
Gautham Krishnamoorthy, U. of North Dakota  
Lauren Clarke, U. of North Dakota

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

Setup $A, x, b$

Construct Eqns

Solve $Ax = b$
Problem Statement

MFIX Algorithm

Construct Eqns

Setup A, x, b

Solve Ax=b

Incorporate Hypre/PETSc as an inner solver option

Ash cluster, Center for High Performance Computing, U. of Utah
Linear Solver Packages: Hypre and PETSc

• Both PETSc (ANL) and Hypre (LLNL) are fairly mature with active development
• Large user bases
• Many examples of good scaling up to large numbers of cores for sparse linear systems
• Both are C/C++ based codes with Fortran interfaces
• Roadmaps for heterogenous architecture (GPU, OpenMP, ...)

![Graph showing performance of Hypre](image-url)
Major Objectives

- **Non-disruptive** implementation with tech transfer
- **Verified implementations** of the third-party linear solvers
- Demonstrate **parallel scaling** on local resources (up to 7K cores)
- Demonstrate **algorithmic scaling** (robustness)
Team

Project Manager
G. Krishnamoorthy

J. Thornock, U. of Utah PI
Surya Yamujala, Student

G. Krishnamoorthy, UND PI
Lauren Clarke, Student

• Experience with HYPRE in an in-house LES code.
• Symmetric Pressure Poisson

• Experience with PETSC and HYPRE for solving the RTE.
• RTE is non-symmetric

MFIX TEAM (Jordan Musser, Jeff Dietiker)
Jason Hissam
Computational Implementation

Development Principles

- Non-disruptive interface to the linear solver options
- Useable

- PETSc and Hypre are easy to build!

- Linking 3p packages handled with modification of environment variables during configure

- Leverage the existing modules for input file parameters (Bools, Ints, Floats, etc)

- New Fortran modules hold the 3p solver interface

- Logic in solve_lin_eq.f direct the algorithm to the selected solver (eqn dependent)
Code Example

solve_lin_eq.f

See: [https://bitbucket.org/jthornock/mfix_hypre_integration](https://bitbucket.org/jthornock/mfix_hypre_integration) for code and wiki documentation. Email J. Thornock for access.
Code Correctness

- Several cases have been tested to demonstrate correctness by comparing against known data or comparing to solution with the native MFiX solver.
- Tests have been performed on single and multiple cores

*Lid driven cavity problem with momentum and pressure solves*
Accuracy

- Comparison of the pressure coefficient (Cp) results using left-side Block Jacobi preconditioning against experimental data.

- Overall, higher-order discretization schemes compare better with experimental data compared to the lower order scheme (FOUP).

- There was little to no difference in results for all preconditioning methods that were tested for the coarse mesh and the intermediate mesh.
3P Overhead - Hypre

- 2-d heat transfer problem
- Work measured as total number of cells
- Ideal lines are just multiples of two from the left or right-most points
Solver Performance: Setup

- **Equation Construction**
- **solve_lin_eq.f**
  Utilize hyperUtilities.f

- **Matrix Setup**
  - Translate MFIX storage to HYPRE objects - remapping of index space
  - Setup of HYPRE solver and preconditioner objects

- **Solve Ax = B (Solve)**

- **Memory Cleanup**
  - Translate HYPRE objects back to MFIX storage - inverse index space mapping and memory cleanup

= expense concerns
- Blue shaded regions exhibit non-ideal behavior
- Solver setup costs are particularly problematic
- Problem persists for multicores
Strong Scaling

Total times

Exploratory Questions/Observation

- Setup the solver less frequently? Effect on overall convergence?
- Do all solvers have a large setup cost?
- More work/Core!
Comparison with MFiX

• Attempts at an apples-to-apple comparison for several test problems
• Varies problem to problem
• Work/core must be large enough for Hypre to be competitive
• Lots of knobs - only just a few have been explored up until now
Comparison with MFiX

- Left and Right preconditioning agrees well with the data.

- Finer mesh resolutions highlight the difference between the solvers

- Efficiency of the linear solver is also highlighted.
Solution Efficiency

- Using flow-over-a-cylinder problem - exploring the stiffness of the linear system using various convection schemes.
- Changing intermediate parameters (wide space)
- Efficiencies are gained by tuning several adjustable solver parameters (tolerance, tolerances, multigrid parameters, etc)
3D Fluidized Bed – Polypropylene

Case 4: Fluidized Bed with Polypropylene Particles (3-Dimensional)

<table>
<thead>
<tr>
<th>Case</th>
<th>Dimensions + Mesh</th>
<th>Time Step</th>
<th>Tolerance</th>
<th>$U_{\text{in}}$</th>
<th>Solver</th>
<th>Scheme</th>
<th>P.C.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>20x100x2 cm$^3$ 40x250x10</td>
<td>DT: 10$^{-3}$ Max: 10$^{-3}$ Min: 10$^{-6}$</td>
<td>Outer: 10$^{-1}$ Solver: 10$^{-3}$</td>
<td>5 m/s</td>
<td>BCGS</td>
<td>van Leer</td>
<td>1. MFiX Line 2. PETSc B(\sqrt{\text{ACOBI}}) (left) 3. PETSc B(\sqrt{\text{ACOBI}}) (right)</td>
</tr>
<tr>
<td>4.2</td>
<td>20x100x2 cm$^3$ 40x250x10</td>
<td>DT: 10$^{-3}$ Max: 10$^{-3}$ Min: 10$^{-6}$</td>
<td>Outer: 10$^{-1}$ Solver: 10$^{-3}$</td>
<td>20 m/s</td>
<td>BCGS</td>
<td>van Leer</td>
<td>1. MFiX Line 2. PETSc B(\sqrt{\text{ACOBI}}) (left) 3. PETSc B(\sqrt{\text{ACOBI}}) (right)</td>
</tr>
<tr>
<td>4.3</td>
<td>20x100x2 cm$^3$ 40x250x10</td>
<td>DT: 10$^{-3}$ Max: 10$^{-4}$ Min: 10$^{-6}$</td>
<td>Outer: 10$^{-1}$ Solver: 10$^{-1}$</td>
<td>5 m/s</td>
<td>BCGS</td>
<td>van Leer</td>
<td>1. MFiX Line 2. PETSc B(\sqrt{\text{ACOBI}}) (left) 3. PETSc B(\sqrt{\text{ACOBI}}) (right)</td>
</tr>
</tbody>
</table>

Case 5: Fluidized Bed with Polypropylene Particles (2-Dimensional)

<table>
<thead>
<tr>
<th>Case</th>
<th>Dimensions + Mesh</th>
<th>Time Step</th>
<th>Tolerance</th>
<th>$U_{\text{in}}$</th>
<th>Solver</th>
<th>Scheme</th>
<th>P.C.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>20x100 cm$^2$ 56x250</td>
<td>DT: 10$^{-3}$ Max: 10$^{-3}$ Min: 10$^{-6}$</td>
<td>Outer: 10$^{-1}$ Solver: 10$^{-1}$</td>
<td>5 m/s</td>
<td>BCGS</td>
<td>van Leer</td>
<td>1. MFiX Line 2. PETSc B(\sqrt{\text{ACOBI}}) (left) 3. PETSc B(\sqrt{\text{ACOBI}}) (right)</td>
</tr>
<tr>
<td>5.2</td>
<td>20x100 cm$^2$ 56x250</td>
<td>DT: 10$^{-3}$ Max: 10$^{-3}$ Min: 10$^{-6}$</td>
<td>Outer: 10$^{-1}$ Solver: 10$^{-3}$</td>
<td>5 m/s</td>
<td>BCGS</td>
<td>van Leer</td>
<td>1. MFiX Line 2. PETSc B(\sqrt{\text{ACOBI}}) (left) 3. PETSc B(\sqrt{\text{ACOBI}}) (right)</td>
</tr>
<tr>
<td>5.3</td>
<td>20x100 cm$^2$ 56x250</td>
<td>DT: 10$^{-3}$ Max: 10$^{-3}$ Min: 10$^{-6}$</td>
<td>Outer: 10$^{-3}$ Solver: 10$^{-3}$</td>
<td>5 m/s</td>
<td>BCGS</td>
<td>van Leer</td>
<td>1. MFiX Line 2. PETSc B(\sqrt{\text{ACOBI}}) (left) 3. PETSc B(\sqrt{\text{ACOBI}}) (right)</td>
</tr>
</tbody>
</table>
Fluidized Bed

CPU Time Ratio
PETSc/MFiX

Inner Iterations
Example: FluidBed1

Increased resolution of original input file by a factor of four.

1st attempt with native MFiX solve: diverged. :-(

2nd attempt with gmres/smg: converged. :-)
Summary

• Several test problems have been explored and tested for correctness (passed), both multicore and single core.

• Performance has been fairly well characterized on a series of single phase problems, highlighting the overhead costs of Hyper.

• Comparisons with MFiX are favorable, but may depend on the scenario and require solver parameter tuning, of which there are a few in Hypre, especially the multigrid parameters.

• Some cases better algorithmic scaling when compared to the use of the native MFiX solver.
Future Work

- Include more **complexity** in the problems we are exploring (spouted beds for CLC, etc.)
- **Scale-up** the problems to larger core counts
- Look for ways to **amortize** setup costs
- Other efficiency gains? **OpenMP**?
- Discussion with **MFiX team**: What problems would they like to see?