Interfacing MFiX with PETSc and HYPRE Linear Solver libraries

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

Co-PI: Jeremy Thornock (U.Utah)
Surya Yamujala (Grad Student, U.Utah)

April 9th, 2019

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

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

(Multigrid solvers and pre-conditioners)

(Multigrid solvers and pre-conditioners)

• Code verification against established MFiX solutions and code to code comparisons

Improved speed and scaling performance
Project Successes

• Lauren Elizabeth Clarke (Combined BS/MS Student, UND)
  • Mickey Leland Energy Fellow ‘17
  • 2 peer-reviewed papers (1 published, 1 under review)
  • Currently pursuing PhD at MIT

• KayLee Smith (Combined BS/MS Student, UND)
  • Mickey Leland Energy Fellow ‘18
  • Placed 1st (Engineering) in UND’s Graduate Student Expo

• Surya Yamujala (MS Student, UUtah)
  • Quality and Reliability Assurance Engineer at IMFlash

• The MFiX-PETSc, MFiX-HYPRE interfaces have turned out to be faster and more robust in several problems including:
  • Domains involving local cell refinements and abrupt cell jumps
  • Complex geometries and large cell count (multi-tube BFB, multi-spout CLC reactors)
  • Large relaxation-factors
  • Large property variations (laminar sCO2 flows)
  • Multi-fluid (different particle sizes) systems
  • Increased collaboration between UND ↔ UUtah
Our First Step
(Identification of optimum solvers and pre-conditioners)

\[ \nabla^2 A + k^2 A = -f \]

Stand alone solver timing studies

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>CG</th>
<th>GMRES</th>
<th>BiCGSTAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>150K</td>
<td>1.56</td>
<td>11.11</td>
<td>2.16</td>
</tr>
<tr>
<td>600K</td>
<td>23.45</td>
<td>700.00</td>
<td>35.56</td>
</tr>
</tbody>
</table>

Best stand alone solver with pre-conditioning options in brackets

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>CG (Point Jacobi)</th>
<th>CG (Block Jacobi)</th>
<th>CG (ILU)</th>
<th>CG (SOR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>150K</td>
<td>1.29</td>
<td>1.06</td>
<td>1.06</td>
<td>1.00</td>
</tr>
<tr>
<td>600K</td>
<td>25.24</td>
<td>19.31</td>
<td>18.01</td>
<td>17.87</td>
</tr>
<tr>
<td>1.2M</td>
<td>57.64</td>
<td>42.94</td>
<td>41.76</td>
<td>40.00</td>
</tr>
</tbody>
</table>
A Road Block!

Asymmetry and the use of Conjugate Gradients

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.
Coarse Mesh

(a)

Fine Mesh

(b)

- MFix (Line)
- MFix (Diag)
- MFix-PETSc (SOR - left)
- MFix-PETSc (SOR - right)
- MFix-PETSc (BJACOBI - left)
- MFix-PETSc (BJACOBI - right)
Polypropylene
(Non-Uniform Grid, 2D, H/D = 2)

Glass
(Non-Uniform Grid, 2D, H/D = 2)
Glass, 2D, Non-Uniform Mesh

Polypropylene, 2D, Non-Uniform Mesh

Glass, 3D, Non-Uniform Mesh

Polypropylene, 3D, Non-Uniform Mesh

<table>
<thead>
<tr>
<th>Preconditioner</th>
<th>2D, Non-Uniform</th>
<th>3D, Non-Uniform</th>
<th>2D, Non-Uniform</th>
<th>3D, Non-Uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFiX – line relaxation</td>
<td>1.3</td>
<td>3.8</td>
<td>4.8</td>
<td>4.7</td>
</tr>
<tr>
<td>MFiX-PETSc – BJACOBI (left)</td>
<td>1.3</td>
<td>3.9</td>
<td>5.0</td>
<td>5.2</td>
</tr>
<tr>
<td>MFiX-PETSc – BJACOBI (right)</td>
<td>1.5</td>
<td>1.6</td>
<td>3.0</td>
<td>4.6</td>
</tr>
</tbody>
</table>
Scalability

- Machine latency is a fixed cost - when computational work gets close or runs below this cost, parallel scalability (strong) breaks down.

- Algorithmic decisions/design/methods contribute to scalability - some methods/algorithms (weak) scale better than others.

- Multigrid solvers scale like $O(n)$ whereas CG scales like $O(n^{1.5})-O(n^2)$.

- Multigrid scalability has shown problem dependency.

<table>
<thead>
<tr>
<th>Method</th>
<th>Storage</th>
<th>Flops</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE(banded)</td>
<td>$n^5$</td>
<td>$n^7$</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>$n^3$</td>
<td>$n^5 \log n$</td>
</tr>
<tr>
<td>Optimal SOR</td>
<td>$n^3$</td>
<td>$n^4 \log n$</td>
</tr>
<tr>
<td>CG</td>
<td>$n^3$</td>
<td>$n^{3.5} \log n$</td>
</tr>
<tr>
<td>Full MG</td>
<td>$n^3$</td>
<td>$n^3$</td>
</tr>
</tbody>
</table>

To solve the Poisson problem in 3D ($N = n^3$)

Source: Kab Seok Kang, GIST HPC Summer school 2015
Modified “inner solve” - MFIX coupled with HYPRE

- Equation Construction (pressure, momentum, ...)
- MFiX::solve_lin_eq calls hyperUtilities.f
- Translate MFiX storage to HYPRE objects - remapping of index space (Matrix Setup)
- Setup of HYPRE solver and preconditioner objects (Solver Setup)
- Solve Ax = B (Solve)
- Translate HYPRE objects back to MFiX storage - inverse index space mapping

= costly in floating point and communication
HYPRE Solver and Matrix setup costs must be overcome with enough work

Comparison with MFiX on a single core

Benchmark TFM02
Single Core
Efficient regime

setup cost dominates
scaling like $O(n)$
Where is the HYPRE overhead?

Matrix Setup

Solver Setup

Actual Solve

Actual Solve - strong and weak scaling
Setup Cost Mitigation

- Can setup costs be mitigated by:
  - reusing the solver/matrix objects every iteration?
  - over the lifetime of the simulation?

![Comparison of overall timings of HYPRE and MFIX solver](chart.png)

- Reuse of matrix objects shows same scaling behavior with a reduction in time.
- Reuse of solver objects was met with memory leaks and other issues - still needs investigation.
- Solver object problems could be due to underlying communication requirements in HYPRE for Struct interface.
- For the Hypre-Struct interface, no solver reinitialize option was available (as opposed to other interfaces, e.g. IJ-Interface).
Fluidized Bed Problem

Comparison of hypre and mfix result - CFB

- mfix result - 32 cores
- hypre result - 32 cores

Void fraction

0.670
0.665
0.660
0.655
0.650
0.645
0.640
0.635
0.630

timestep [s]
0
10
20
30
40
50

Solid volume fraction after some time

HYPRE

MFIX

- Initial apple-to-apples comparison (same solver tolerance, max iterations)
- MFiX native solver diverged while HYPRE converged
- PFMG preconditioner consistently offers better performance (well known)
Can HYPRE inner solver robustness be exploited?

Benchmark TFM03 Multiple Cores

Pressure Convergence with high outer relaxation factors

- HYPRE appears to support higher relaxation factors (it converges) for the outer solve
- Stalling or slow/noisy convergence seen with MFiX native solver with higher relaxation factors
Challenge Problem: Flow over a tube bank

Same Workload per Core

- **Workload/Core \(O(10^4)\)**
- Promising at lower core counts
- Should repeat with more workload/core

Summary
Challenge Problem: Flow over a tube bank

- 5X more work/core
- \(\sim\)3.5-2.5X speedup
- Needs more work, further study