SAND2017-8245C
Proceedings of the ASME 2017 Fluids Engineering Division Summer Meeting
FEDSM2017
July 31-August 3, 2017, Waikoloa, Hawaii, USA

FEDSM2017-69555

NEXT-GENERATION MULTIPHASE FLOW SOLVER FOR FLUIDIZED BED APPLICATIONS

V M Krushnarao Kotteda
Mechanical Engineering
University of Texas at El Paso
El Paso, Texas–79968
vkotteda@utep.edu

Ashesh Chattopadhyay
Computational Science
University of Texas at El Paso
El Paso, Texas–79968

Vinod Kumar
Mechanical Engineering
University of Texas at El Paso
El Paso, Texas–79968

William Spotz
Sandia National Laboratories
Albuquerque, New Mexico–87185

ABSTRACT

A framework is developed to integrate MFiX (Multiphase Flow with Interphase eXchanges) with advanced linear solvers in Trilinos. MFiX is a widely used open source general purpose multiphase solver developed by National Energy Technology Laboratories and written in Fortran. Trilinos is an objected-oriented open source software development platform from Sandia National Laboratories for solving large scale multiphysics problems. The framework handles the different data structures in Fortran and C++ and exchanges the information from MFiX to Trilinos and vice versa. The integrated solver, called MFiX-Trilinos hereafter, provides next-generation computational capabilities including scalable linear solvers for distributed memory massively parallel computers. In this paper, the solution from the standard linear solvers in MFiX-Trilinos is validated against the same from MFiX for 2D and 3D fluidized bed problems. The standard iterative solvers considered in this work are Bi-Conjugate Gradient Stabilized (BiCGStab) and Generalized minimal residual methods (GMRES) as the matrix is non-symmetric in nature. The stopping criterion set for the iterative solvers is same. It is observed that the solution from the integrated solver and MFiX is in good agreement.

NOMENCLATURE

- $\vec{g}$ Acceleration due to gravity
- k Conductivity
- J Collisional dissipation
- ρ Density
- Π Exchange term
- Θ Granular temperature
- I_g Interphase momentum exchange
- P Pressure
- τ Stress tensor
- $\vec{U}$ Velocity vector
- ε Volume fraction
- t Time

subscript

- g gas phase
- s solid phase

1 INTRODUCTION

Fluidized beds are generally used in petroleum, pharmaceutical, chemical, mineral and fossil fuel plants [1–14]. Gasification of a feedstock in fluidized bed increases the efficiency of

the power plant and reduces the greenhouse gases. In a fluidized bed, the gasification of solid particles takes place via air from a distributor plate. Initially, the pressure drop across the bed is zero and the particles are at rest. The particles move as the pressure drop across the bed increases with increase in fluid velocity. At the upward superficial velocity, the bed weight equals to the pressure force due to the flow velocity and the bed is said to be fluidized. Further increase in the velocity pushes the excess flow from the bed in the form of bubbles or slugs. The flow in fluidized beds is multi-physics in nature. Therefore, understanding of such complex flow is still limited [15]. The existing computer applications lack the capabilities for designing, optimizing, and controlling of industrial-scale fluidized bed reactors. Sophisticated models can improve the capabilities of such applications. To improve the fluidization simulation capabilities of MFiX (Multiphase Flow with Interphase eXchanges), it is integrated with state-of-the art linear solvers in Trilinos.

MFiX, a general-purpose computer code, is developed at the National Energy Technology Laboratory (NETL) for studying dynamics of fluids, thermal effects and chemical reactions in fluidized beds. It includes successive over relaxation method, Bi-Conjugate gradients stabilized (BiCGStab) method, generalized minimal residual method and conjugate gradient method for solving the linear system of equations [16]. BiCGStab is most commonly used [17] method to solve the system of equations as it is less expensive and can handle non-symmetric system of equations. The preconditioners accessible to MFiX are: none, diagonal and line/plane relaxation. In the present study, BiCGStab/GMRES method is used to solve system of equations.

Two memory models available in MFiX are: shared and distributed memory. The linear solver routines in MFiX consumes more than 70% of the overall runtime [18] and are parallelized with OpenMP directives. The remaining time is spread across other routines. In the distributed memory version, the domain is decomposed across "I", "J" and "K" indices in a user specified order [18]. Fortran 90 calls to message passing interface (MPI) library update the solution in overlapped region, find matrix vector multiplication, and gather/scatter the data for receive/send operations. In the present study distributed memory version of MFiX is considered for integration with Trilinos.

Modernization techniques leverage libraries that provide state-of-the-art algorithms to improve its capabilities. For example, such solvers and preconditioning packages are widely available in libraries such as Trilinos [19]. Trilinos [19, 20], an exascale open-source software library, includes robust algorithms and enabling technologies for solving large scale flow problems besides others. The library includes more than 50 packages including discretization, load balancing, preconditioning, linear, nonlinear, eigen value, uncertainty quantification and transient solver packages which uses more than 100 third party libraries such as Blas, Lapack, Metis, ParMetis, SuperLU and PETSc. Trilinos contains libraries that exploits hybrid architectures composed of a general-purpose multi-core processor and a graphics processing unit (GPU) via static and dynamic polymorphism [21]. Static polymorphism results in generic algorithms. Static polymorphism results in generic algorithms, whereas the dynamic one decouples linear solvers from preconditioning, orthogonalization and stopping criteria. The modern linear solver packages in Trilinos offer portability, scalability and ease of integration with other solvers/libraries, for example, MFiX. However, the integration of those libraries with a legacy code such as MFiX poses a unique challenge to handle semantically different data structures [22, 23, 23–25].

In the present work, MFiX [16] is integrated with modern linear solvers in Trilinos [19] via an interface. The interface exchanges the information among the softwares/libraries written in two different programming languages such as Fortran and C++. It consists of MFIX, Fortran, C and Cpp wrappers to exchange the information required to solve the linear system. The MFIX wrapper understand the structure of the linear equations and passes the information to the Fortran Wrapper. The Fortran wrapper transfers that information to the C wrapper. This C wrapper transforms the memory semantics and passes the information to the Cpp wrapper that implements the object oriented programming interface. This framework handles the communication between distributed memory parallel MFiX and Trilinos. It enables us to access the advanced linear solvers in Trilinos and exploit the hybrid computing environments. The framework used for the integration is described in Section 2.1. The integrated solver is tested on fluidized beds in two and three dimensions. MFiX uses native iterative method to solve the system of equations, whereas the integrated solver uses same method in next-generation solver packages in Trilinos. Two iterative methods considered to solve the system of equations are BiCGStab and GMRES. The stopping criteria for the method in MFiX and Trilinos is same. The equations governing the fluid are described in Section 3. An algorithm in finite volume method, modified semi-implicit pressure linked equations (SIMPLE) [26], discretize the governing equations. The averaged gas and solid phase quantities are used to evaluate the drag force. The results with the integrated solver are presented in Section 4.

2 MFiX-TRILINOS OVERVIEW

MFiX is generally used to simulate fluidization phenomenon with thermal effects in fluidized beds. In MFiX, the equations governing the flow are solved via the modified SIMPLE algorithm. Two modifications to the algorithm have improved the stability and speed of calculations [16]. The modified solid volume fraction equation improves the convergence. The automatic time-step adjustment improves the execution time. Thus, it was observed that the modified algorithm is 3-30 times faster than the actual algorithm [16]. In the present study, MFiX with the two modifications is integrated with Trilinos.

Three different suites of models available in MFiX to simulate fluidized bed reactors are: Two-fluid (TFM), Discrete element (DEM) and Particle in cell (PIC) models. The first and the last models are Eulerian-Eulerian models whereas DEM is an Eulerian-Lagrangian model. In TFM model, the fluid and solid phases are assumed to be continuum and are modeled with separate sets of conservation equations with proper interaction terms. In this model, the particle pressure experience convergence problems when particles are not completely fluidized. However, DEM model traces every solid particle and results in high fidelity solution. In PIC, the fluid is assumed to be continuum while using "parcels" to represent groups of particles with similar characteristics. This model offers reduced computational cost over DEM. It is noticed that the framework to simulate the solid phase is different in different models. However, fluid phase exists in Eulerian framework in all the three models. Therefore, in the present study, the linear system of equations for fluid phase are solved with MFiX and MFiX-Trilinos. The equations for the solid phase are solved with native solver in MFiX. Therefore, MFiX-Trilinos can be used to simulate the flow with any of the three models in MFiX. In the present study, we considered TFM model to simulate flow in fluidized beds. However, the framework can be used to solve the system of equations in fluid or solid phase or both phases.

Trilinos includes number of next-generation scalable and performant packages [19]. The intention of developing these packages was to provide advanced features such as generic template programming, smart pointers and lambdas expressions in C++11 to application writers for scientific computing. The first step in solving the system of equations with Trilinos is to create a map. A pseudocode of the Cpp wrapper is shown in Fig. 1.

In Cpp wrapper, `Tpetra Map` class is used to create a map. This map has various query functions to compute various attributes. For example, `MaxMyGID()` returns the maximum global index value on the calling processor. The Map is used to create a graph and the graph is used to create an object for a sparse or dense matrix. In the present study, MFiX provides the sparse matrix. Thus the graph is created for the sparse matrix (`A`). The graph keeps the matrix structure constant and allows only to change the values of elements in A. In 2D problems, there are maximum five non-zero elements in a row. On the other hand, the matrix for 3D flow problems contains maximum seven non-zero elements in a row. A function in `Tpetra CrsMatrix` class, `replaceGlobalValues()`, is used to populate the matrix, `A`, via the row pointers (GlobalRow in Fig.1), the column pointers (Indices) and non-zero values (Values). The LHS and RHS vectors are created via Tpetra MultiVector class. The RHS vector can be filled up using the `SumIntoGlobalValues()` function in Tpetra MultiVector class. The linear problem is created using the sparse matrix `A` and `LHS` and `RHS` vectors. The preconditioning is formed with `MueLu`, an algebraic multi-grid preconditioner/solver package in Trilinos. The parameters to form the

```
1  Tpetra_Map map(numGlobalElements,numMyElements,
       indexBase,comm);
2  Tpetra_graph mgraph(map,cols,Tpetra::StaticProfile);
3  for(i=0;i<numMyElements;++i){
4     gblRow=map->getGlobalElement(i);
5     mgraph->insertGlobalIndices(gblRow,NumEntries,
          Indices);
6  }
7
8  Tpetra_crsMatrix A(mgraph);
9  for (i=0;i<numMyElements;++i) {
10    Values[]=Am[][];
11    Indices[]=pos[][];
12    A->replaceGlobalValues(gblRow,NumEntries,Values,
          Indices);
13    }
14 A->fillComplete();
15
16 Tpetra_multivector LHS(map,1);
17 Tpetra_multivector RHS(map,1);
18 for (i=0;i<numMyElements;++i) {
19    gblRow=map->getGlobalElement(i);
20    RHS->sumIntoGlobalValue(gblRow,0,bm[i]);
21 }
22
23 Tpetra_LinearProblem problem(A,LHS,RHS);
24 problem->setRightPrec(mueLuPreconditioner);
25 belos_solver_manager_type solver(problem,belosList);
26 solver->solve();
```

FIGURE 1. A sample Cpp wrapper with Tpetra, Belos packages in Trilinos to solve linear system of equations.

preconditioner can be set through `SetParameters()` function or a xml file. In the present study, parameters (mueLuPreconditioner) are set through xml file. The packages, for example, `MueLu` and `Belos`, validates the input parameter list in the xml file and halt the execution if a parameter is misspelled, unknown, or has an incorrect value type. An iterative method in `Belos` [27], one of the next-generation solver package, is used as a solver. The solver attributes are set to their default values when `Belos` object is constructed. These defaults values are usually changed by calling `SetParameters()` functions. For example, stopping criteria such as the maximum number of iterations and a tolerance can be set to the required values. Therefore, the default solver is changed to either BiCGStab or GMRES. `Solve()` function solves the system of equations according to the `Belos` parameters list and stores the solution in `LHS`.

2.1 A FRAMEWORK TO INTEGRATE MFiX WITH TRILINOS

MFiX is written in Fortran and cannot directly be integrated with a library, Trilinos, written in object oriented framework as the memory layout in C++ and Fortran are different. Hence, a framework is developed to integrate the softwares/libraries writ-

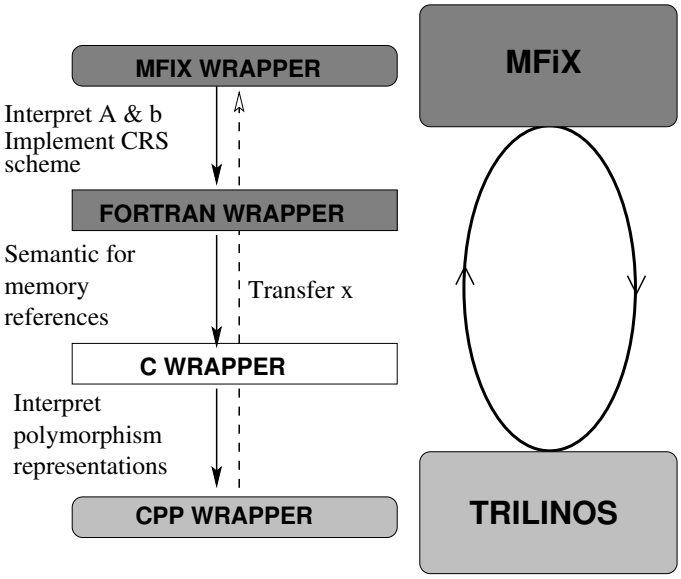

FIGURE 2. A schematic of the framework to integrate MFiX with Trilinos.

ten in different languages and exchange the information irrespective of a computer architecture. Hence a well scrutinized and structured communication protocol across the language barriers must provide semantic support to the software/library on either sides of the framework. The framework includes MFIX, Fortran, C and Cpp wrapper. It is shown in Fig.2. The MFIX wrapper is a Fortran code that understand the structure of the linear system of equations from MFiX. Arrays required for storing the sparse linear system of equations are created in MFIX wrapper and passed to the Fortran Wrapper. The Fortran wrapper transfer the arrays to the C wrapper. The C wrapper acts as a mediator and changes the memory semantics of the arrays in Fortran to C++. The Cpp wrapper uses the arrays to form the system of equations and those equations are solved with an iterative method in Belos. This can be seen in Fig.1. More details about the framework can be found in [28].

3 GOVERNING EQUATIONS

The averaging approach is used to obtain the governing equations that describe inter-penetrating continua.

Continuity equation for fluid phase g:

$$\frac{\partial}{\partial t}(\varepsilon_g \rho_g) + \nabla.(\varepsilon_g \rho_g \vec{U}_g) = 0 \tag{1}$$

Continuity equation for solid phase s:

$$\frac{\partial}{\partial t}(\varepsilon_s \rho_s) + \nabla.(\varepsilon_s \rho_s \vec{U}_s) = 0 \tag{2}$$

The governing equations, Eqns (1) - (4), are conservation of mass and momentum for the fluid and solid phases. The first and second term in Eqns (1) and (2) represents the rate of change of mass and the net mass flux, respectively.

Momentum equations for fluid phase g:

$$\frac{\partial}{\partial t}(\varepsilon_g \rho_g \vec{U}_g) + \nabla.(\varepsilon_g \rho_g \vec{U}_g \vec{U}_g) = -\varepsilon_g \nabla P_g + \nabla.\tau_g - I_{gs} + \varepsilon_g \rho_g \vec{g} \tag{3}$$

Momentum equations for solid phase s:

$$\frac{\partial}{\partial t}(\varepsilon_s \rho_s \vec{U}_s) + \nabla.(\varepsilon_s \rho_s \vec{U}_s \vec{U}_s) = -\varepsilon_s \nabla P_s + \nabla.\tau_s + I_{gs} + \varepsilon_s \rho_s \vec{g}. \tag{4}$$

The first term and second term on the left hand side of Eqns(3) and (4) accounts for rate of change of momentum and the advection term, respectively. The first two terms on the right side of the equations represents the force terms, while the last two terms represents the momentum transfer between the fluid and solid phases and body force, respectively. More details about the equations and Eqns (1) - (4) in Cartesian and cylindrical coordinate system can be found in [16, 29].

The equation to model the granular energy for solid phase s:

$$\frac{3}{2}\rho_s \left[\frac{\partial \varepsilon_s \Theta_s}{\partial t} + \nabla.(\varepsilon_s \vec{U}_s \Theta_s) \right] = \nabla.(k_s \nabla \Theta_s) + \tau_s.\nabla \vec{U}_g + \Pi_g - \varepsilon_s \rho_s J_s \tag{5}$$

The first term and second term on the left hand side of Eqn(5) represents the transient and the advection term, respectively. The first two terms on the right side of the equations represents the force terms, while the last two terms represents the energy transfer terms between the fluid and solid phases. More details can be found in [16, 29, 30].

4 RESULTS

MFiX is integrated with Trilinos via the framework described in Section 2.1. The computations are carried out with distributed memory parallel MFiX and MFiX-Trilinos on Stampede, Intel Xeon CPU E5-2680 2.7GHz, 32GB RAM, 64KB L1 cache, 256K L2 cache, 20480K L3 cache. The linear system of equations for pressure from MFiX are non-symmetric in nature. Therefore, BiCGStab and GMRES methods are considered to solve the system of equations. The method to solve the system of equations in MFiX and MFiX-Trilinos is same. The stopping criteria for the Krylov-subspace iterative solver is also same. In the present study, the solution from BiCGStab as well as GMRES method in MFiX and Trilinos is compared.

4.1 COMPUTATIONS IN TWO DIMENSIONS

First, we considered a fluidized bed with central jet in two dimensions. The fluidized bed is $10cm$ wide $100cm$ tall. Initially,

the half of the bed is filled with sand particles of size $0.04cm$ and density of $2g/cm^3$. The void fraction at minimum fluidization of the bed is 0.42. The sand particles are fluidized with air of density of $0.0012g/cm^3$ and viscosity of $0.00018Poise$. The flow is assumed to be laminar as well as incompressible. The computational domain is discretized with 100 equi-spaced cells in the axial direction and 10 equi-spaced cells in the normal direction. Constant mass flow and pressure boundary condition is specified on the bottom and top boundary, respectively. The central jet velocity is $124.6cm/s$ while the velocity of fluid from the distributor plate ($1.43 < x < 10$) is $25.9cm/s$. The computations are carried out to simulate the flow with MFiX and MFiX-Trilinos. In MFiX, the system of equations for pressure field are solved with native BiCGStab method. However, the same system of equations in MFiX-Trilinos are solved with the same solver in `Belos`. The equations in MFiX are passed to Trilinos via MFIX, Fortran, C and Cpp wrappers. MFIX wrapper transfers the matrix and the vectors constructed using the "I" and "J" indices in MFiX to Cpp wrapper via Fortran and C wrappers. In Cpp wrapper, the objects for the matrix (`Am` in Fig.1) and the vector (`bm` in Fig.1) are created with `Tpetra`. This can be seen on line 8, 16 and 17 in Fig.1. The objects are populated with `replace/SumInto GlobalValues` functions in Tpetra. This can be seen on line 12 and 20 in the figure. The system of equations are formed via these objects and solved with the preconditioned iterative solver. This can be seen on line 23 and 26 in the figure, respectively. The convergence criteria set for linear solver in MFiX, for example, number of iterations and normalized residue, are also passed to the Cpp wrapper. BiCGStab as well as GMRES method is used to solve the system of equations. The flow with MFiX-Trilinos and MFiX for BiCGStab is compared in Figure 3. The void fraction from MFiX-Trilinos and MFiX is qualitatively same. This can also be seen in the figure. We also compared the solution from GMRES solver in MFiX and Trilinos and observed that they are qualitatively same. Further, we compared the flow with MFiX and MFiX-Trilinos quantitatively. The comparison is shown in Fig.4. The volume fraction at $y = 25cm$ from MFiX-Trilinos and MFiX is same at $t = 0.3$. A minor difference between the volume fraction from MFiX and MFiX-Trilinos can be seen in the figure. However, the difference is less than 1% and negligible. However, the convergence study to quantify the numerical errors will be done in the future.

4.2 COMPUTATIONS IN THREE DIMENSIONS

The fluidized bed with central jet in two-dimensions is extended in three dimensions. The width of the bed is 10cm. The number of cells uniformly distributed in the widthwise direction is 10. The velocity of the jet is $61.7cm/s$ while the velocity of fluid from the distributor plate ($1.43 < x < 10, 0 < z < 10$) is $25.9cm/s$. The initial condition is similar to the one specified for the 2D computations. The equations governing the fluid in the

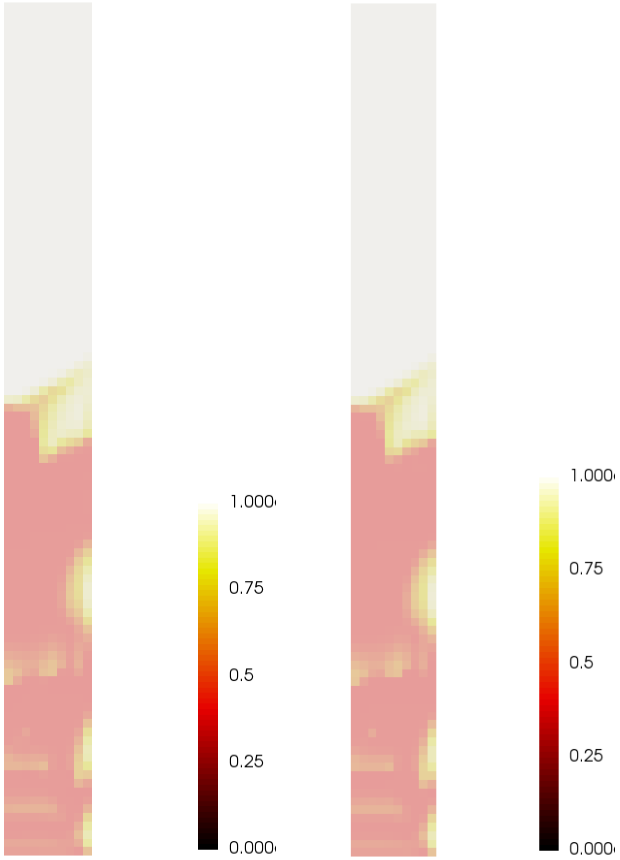

FIGURE 3. Flow (volume fraction) in a fluidized bed $t = 0.85$ in two dimensions with MFiX (left) and MFiX-Trilinos (right).

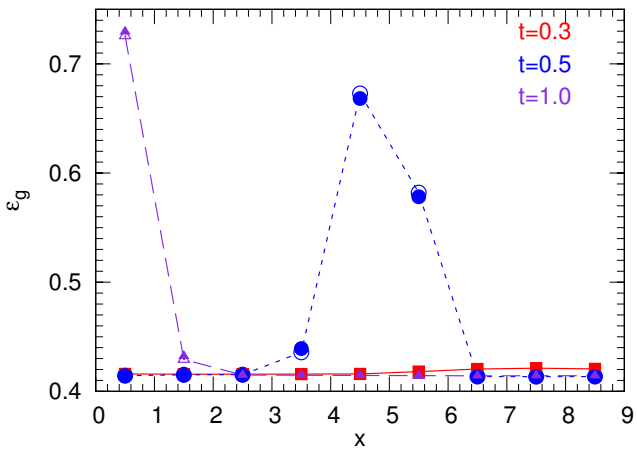

FIGURE 4. Volume fraction in the fluidized bed along $y = 25cm$ at various time instants. Open symbols denote results from MFiX while the closed symbols represent the results from MFiX-Trilinos.

Cartesian coordinate system are solved with MFiX and MFiX-Trilinos. In MFiX Trilinos, MFIX wrapper transfers the matrix and the vectors constructed in MFiX via the "I", "J" and "K" indices to the Cpp wrapper. In cpp wrapper, the objects for a septa-diagonal matrix (`Am` in Fig.1) and vectors (`bm` in Fig.1) are created. The non-zero entries in the sparse matrix and vectors are inserted via `replace/SumInto GlobalValues` functions in `Tpetra CrsMatrix` and `vector`, respectively. An iterative method is used to solve the system of equations. MFiX uses native BiCGStab method to solve the system of equations while MFiX-Trilinos uses the same method in Trilinos. The iso-surfaces of volume fraction with MFiX and MFiX-Trilinos at time $t = 0.75$ is shown in Fig. 5. The flow from MFiX-Trilinos is same as that from MFiX. This can also be seen in the figure. Further, we compared the flow with MFiX and MFiX-Trilinos quantitatively. The comparison is shown in Fig.6. The volume fraction with MFiX-Trilinos and MFiX is same for $t = 0.3$ and 1.0. A minor difference between the volume fraction from MFiX and MFiX-Trilinos for $t = 0.5$ can be seen in the figure. The difference is less than 1% and negligible. The solver time in MFiX and MFiX-Trilinos is compared and observed that the solvers in Trilinos are approximately 1.2 times faster than those in MFiX.

The fluidized bed with central jet in two-dimension is extended in the three dimensions. The bed is cylindrical vessel of 10cm diameter 100cm tall. The velocity of the jet is $61.7cm/s$ while the velocity of fluid from the distributor plate ($1.43 < r < 10, 0 < \theta < 180$) is $25.9cm/s$. The computational domain is consists of 100 and 10 equi-spaced cells in the axial and radial and tangential directions, respectively. The equations in cylindrical co-ordinate system are solved. Constant mass flow is specified on the bottom boundary while constant pressure outflow condition is specified on the top boundary. The initial condition is similar to the one specified for the computations in two dimensions. BiCGStab is used to solve the linear system of equations. Figure 7 shows the flow (void fraction) at time $t = 0.75$ with MFiX (left) and MFiX-Trilinos (right). The flow from the MFiX-Trilinos and MFiX is same. This can also be seen in the figure.

5 CONCLUSIONS

MFiX is integrated with modern linear solvers in Trilinos via MFIX, Fortran, C and Cpp wrappers. The MFIX wrapper understands the structure of the linear equations and passes the information to the Fortran Wrapper. The Fortran wrapper transfers that information to the C wrapper. This C wrapper transforms the memory semantics and passes the information to the Cpp wrapper that implements the object oriented programming interface. This framework handles the communication between distributed memory parallel MFiX and Trilinos. And, enables us to access the advanced linear solvers in Trilinos and exploit the hybrid computing environments.

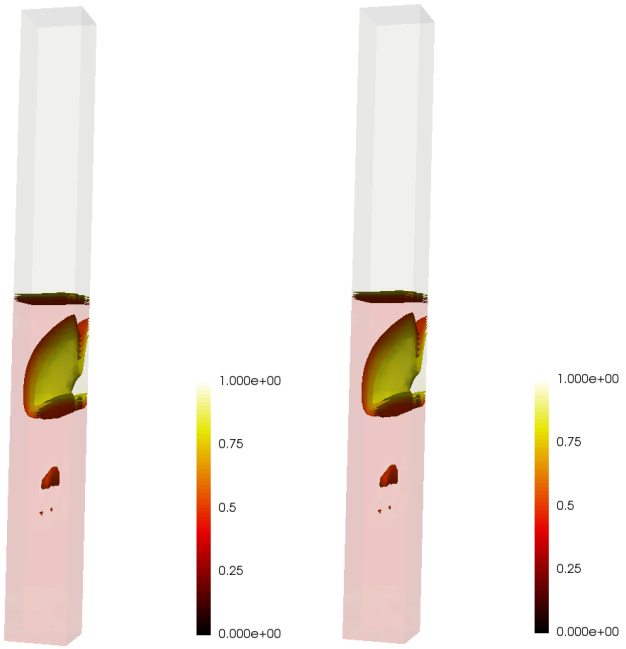

FIGURE 5. Flow (volume faction) in a fluidized bed at $t = 0.75$ in three dimensions with MFiX (left) and MFiX-Trilinos (right). The equations governing the flow in Cartesian co-ordinate system are solved with MFiX and MFiX-Trilinos.

The integrated solver is tested on fluidized bed problems in two and three dimensions. MFiX uses native iterative method to solve the system of equations, whereas the integrated solver uses the same method in next-generation solver packages in Trilinos. Two iterative methods considered to solve the system of equations are BiCGStab and GMRES. The stopping criteria for the method in MFiX and Trilinos is same. The solution from the integrated solver and MFiX are compared and they are in good agreement.

6 Acknowledgements

This work supported by the U.S. Department of Energy (DOE) National Energy Technology Laboratory (NETL) under Grant Number $DE-FE\_0026220$, XSEDE computational resources (Bridges, Stampede, Lonestar) and Department of Mechanical Engineering and Computational Science Program at University of Texas at El Paso.

REFERENCES

[1] Howard, J., 1983. "Fluidized beds. combustion and applications". *Applied Science Publishers, Ripple Road, Barking, Essex, England, 1983.*

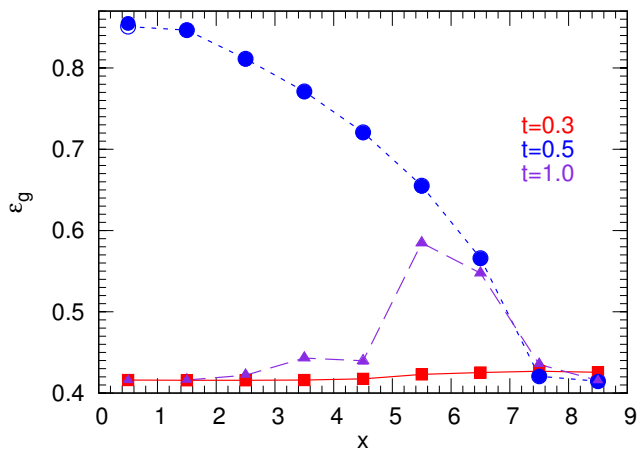

FIGURE 6. Volume fraction in the fluidized bed along $y = 25cm$ and $z = 5cm$ at various time instants. Open symbols denote results from MFiX while the closed symbols represent the results from MFiX-Trilinos.

FIGURE 7. Flow (volume faction) in a fluidized bed at $t = 0.75$ in three dimensions with MFiX (left) and MFiX-Trilinos (right). The governing equations in cylindrical co-ordinate system are solved with MFiX and MFiX-Trilinos.

[2] Davidson, J. F., Clift, R., and Harrison, D., 1985. *Fluidization*. Academic Press, Inc., Orlando, FL.

[3] Tsuji, Y., Kawaguchi, T., and Tanaka, T., 1993. "Discrete particle simulation of two-dimensional fluidized bed". *Powder technology,* **77**(1), pp. 79–87.

[4] Anthony, E., 1995. "Fluidized bed combustion of alternative solid fuels; status, successes and problems of the technology". *Progress in Energy and Combustion Science,* **21**(3), pp. 239–268.

[5] Jacqmin, D., 1999. "Calculation of two-phase navier–stokes flows using phase-field modeling". *Journal of Computational Physics,* **155**(1), pp. 96–127.

[6] Jalali, A. A., and Hadavand, A., 2007. "Bed temperature control of a circulating fluidized bed combustion system using h algorithm". In Control, Automation and Systems, 2007. ICCAS'07. International Conference on, IEEE, pp. 2658–2662.

[7] Sun, J., Battaglia, F., and Subramaniam, S., 2007. "Hybrid two-fluid dem simulation of gas-solid fluidized beds". *Journal of Fluids Engineering,* **129**(11), pp. 1394–1403.

[8] Van der Hoef, M., van Sint Annaland, M., Deen, N., and Kuipers, J., 2008. "Numerical simulation of dense gas-solid fluidized beds: A multiscale modeling strategy". *Annu. Rev. Fluid Mech.,* **40**, pp. 47–70.

[9] Hosseini, S. H., Zhong, W., Esfahany, M. N., Pourjafar, L., and Azizi, S., 2010. "Cfd simulation of the bubbling and slugging gas-solid fluidized beds". *Journal of Fluids Engineering,* **132**(4), p. 041301.

[10] Hu, X., Wang, T., Song, J., and Dong, C., 2012. "Research on wear mechanism of dense phase area in fluidized bed". In Power and Energy Engineering Conference (APPEEC), 2012 Asia-Pacific, IEEE, pp. 1–4.

[11] Grace, J. R., Knowlton, T., and Avidan, A., 2012. *Circulating fluidized beds*. Springer Science & Business Media.

[12] Yates, J., 2013. *Fundamentals of Fluidized-Bed Chemical Processes: Butterworths Monographs in Chemical Engineering*. Butterworth-Heinemann.

[13] Kunii, D., and Levenspiel, O., 1991. *Fluidization engineering*. Butterworth-Heinemann, Boston, MA.

[14] Yamaguchi, H., Niu, X.-D., Nagaoka, S., and De Vuyst, F., 2011. "Solid-liquid two-phase flow measurement using an electromagnetically induced signal measurement method". *Journal of Fluids Engineering,* **133**(4), p. 041302.

[15] Crowe, C., 1993. "Research needs in fluids engineering: Basic research needs in fluid-solid multiphase flows". *Journal of fluids engineering,* **115**(3), pp. 341–342.

[16] Syamlal, M., Rogers, W., and OBrien, T. J., 1993. "MFIX documentation: Theory guide". *National Energy Technology Laboratory, Department of Energy, Technical Note DOE/METC-95/1013 and NTIS/DE95000031*.

[17] Barrett, R., Berry, M. W., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Van der Vorst, H., 1994. *Templates for the solution of linear systems: building blocks for iterative methods*, Vol. 43. Siam.

[18] Pannala, S., D'Azevedo, E., and Syamlal, M., 2003. "Hy-

brid (openmp and mpi) parallelization of MFIX: a multiphase cfd code for modeling fluidized beds". In Proceedings of the 2003 ACM symposium on Applied computing, ACM, pp. 199–206.

[19] Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., et al., 2005. "An overview of the trilinos project". *ACM Transactions on Mathematical Software (TOMS),* **31**(3), pp. 397–423.

[20] Lin, P., Bettencourt, M., Domino, S., Fisher, T., Hoemmen, M., Hu, J., Phipps, E., Prokopenko, A., Rajamanickam, S., Siefert, C., et al., 2014. "Towards extreme-scale simulations with next-generation trilinos: a low mach fluid application case study". In Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International, IEEE, pp. 1485–1494.

[21] Baker, C. G., Hetmaniuk, U. L., Lehoucq, R. B., and Thornquist, H. K., 2009. "Anasazi software for the numerical solution of large-scale eigenvalue problems". *ACM Transactions on Mathematical Software (TOMS),* **36**(3), p. 13.

[22] Chisnall, D., 2013. "The challenge of cross-language interoperability". *Communications of the ACM,* **56**(12), pp. 50–56.

[23] Gel, A., Pannala, S., Syamlal, M., O'Brien, T., and Gel, E. S., 2007. "Comparison of frameworks for a next-generation multiphase flow solver, MFIX: a group decision-making exercise". *Concurrency and Computation: Practice and Experience,* **19**(5), pp. 609–624.

[24] Gel, A., and Hu, J., 2015. "Modernizing a legacy open source cfd code by leveraging scalable parallel preconditioners and linear equation solvers". In 27th International Conference on Parallel Computational Fluid Dynamics, McGill University, Montreal, QC Canada.

[25] Bentley, J. L., and McIlroy, M. D., 1993. "Engineering a sort function". *Software: Practice and Experience,* **23**(11), pp. 1249–1265.

[26] Versteeg, H. K., and Malalasekera, W., 2007. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education.

[27] Heroux, M., Bartlett, R., Hoekstra, V. H. R., Hu, J., Kolda, T., Lehoucq, R., Long, K., Pawlowski, R., Phipps, E., Salinger, A., et al., 2003. An overview of trilinos. Tech. rep., Citeseer.

[28] Kotteda, V. K., Chattopadhyay, A., Kumar, V., and Spotz, W., 2016. "A framework to integrate mfix with trilinos for high fidelity fluidized bed computations". In High Performance Extreme Computing Conference (HPEC), 2016 IEEE, IEEE, pp. 1–6.

[29] Syamlal, M., et al., 1998. "MFIX documentation: Numerical technique". *National Energy Technology Laboratory, Department of Energy, Technical Note No. DOE/MC31346-5824*.

[30] Li, T., Dietiker, J.-F., and Shahnam, M., 2012. "MFIX simulation of netl/psri challenge problem of circulating fluidized bed". *Chemical engineering science,* **84**, pp. 746–760.