**NURETH14-xxx**

# PARALLEL LINEAR SOLVERS FOR SIMULATIONS OF REACTOR THERMAL HYDRAULICS

**Y. Yan[1], S. P. Antal[2], B. Edge[2], D. E. Keyes[1], D. Shaver[2], I. A. Bolotnov[3] and M. Z. Podowski[2]**
[1]Columbia University, New York, NY, USA
[2]Rensselaer Polytechnic Institute, Troy, NY, USA
[3]North Carolina State University, Raleigh, NC, USA

## Abstract

The state-of-the-art multiphase fluid dynamics code, NPHASE-CMFD, performs multiphase flow simulations in complex domains nonlinearly implicitly and in parallel, which is a challenging environment for the linear solver. The present work illustrates how the Portable, Extensible Toolkit for Scientific Computation (PETSc) and scalable Algebraic Multigrid AMG preconditioner from Hypre can be utilized to construct robust and scalable linear solvers for the Newton correction equation resulting from the discretized governing conservation law systems in NPHASE-CMFD. The overall long-tem objective of the work is to cover NPHASE-CMFD into a fully-scalable solver of multiphase flow and heat transfer problems, applicable to both steady-state and time-dependent phenomena in complete fuel assemblies of nuclear reactors and, eventually, the entire reactor core (such as the Virtual Reactor concept envisioned by CASL).

## Introduction

As the fidelity of the modelling and the resolution of computer simulations of complex physical phenomena increase in order to deliver predictive capabilities for scientific discovery and engineering design, by taking advantage of leading-edge computer platforms, it is very often the linear solver that becomes the most stringent bottleneck to progress, as measured by the fraction of execution time. It is therefore important to design codes intended for a wide variety of simulation capabilities around a rich library of linear solvers, which can be tuned to the requirements of a particular application. Modern linear solvers allow many tradeoffs between computation, storage, and communication resources, so as to optimize the fit of the application to the resources of the machine. It is difficult to select the best settings of these tradeoffs, so for robustness, simulations often use suboptimal selections of algorithmic variants and suboptimal settings of such tuning parameters as inner and outer convergence tolerances, Krylov subspace sizes, preconditioner fill, multigrid levels, and so forth.

The Portable, Extensible Toolkit for Scientific Computation (PETSc) [1] is a software library with distributed data structures (for gridfunctions and linear and nonlinear operators on them) and a rich selection of algorithmic building blocks for linear solvers, nonlinear solvers, and timesteppers that provide building blocks for the implementation of large-scale application codes on parallel machines, using message passing. The toolkit is organized hierarchically in an object-oriented paradigm, providing users the freedom of employing the most appropriate level of abstraction. The toolkit can be extended by user-registry of new objects (data structures and the operators that go with them) and provides portable interfaces for a vast number of external direct and iterative solver packages, which it configures automatically. Among them and relevant to this paper are Hypre [2] and SuperLU [3]. Solvers can be swapped at launch by command-line scripting, and can be varied adaptively within an execution. This permits wide experimentation. To aid in performance tuning, PETSc offers many levels of profiling. Users can extend PETSc’s profiling capabilities to their own subroutines by properly registering the timers and counters. PETSc also comes with many monitoring capabilities that can be employed for understanding of algorithmic performance and with rudimentary graphical capabilities.

NPHASE-CMFD is a software library developed at the Center for Multiphase Research, RPI [4]. It employs the paradigm of divide-and-conquer through domain decomposition to host the extreme demands of multiscale multiphase fluid simulation on massively distributed memory architectures. The partitioning of the mesh is performed by Metis [5] to minimize the communication cost at subdomain boundaries. NPHASE-CMFD is a fully nonlinearly implicit time-stepping code that guarantees numerical stability by using Newton’s method on each timestep to reduce the nonlinear residual norm arbitrarily below truncation error. The custom algebraic multigrid solver used to solve the Newton correction equations utilizes its own data structure, which is independent of PETSc's algebraic data structures. The minimally invasive objective of the present study is to test PETSc’s capabilities on the linear operators generated by NPHASE-CMFD at each Newton correction stage. NPHASE-CMFD generates an explicit Jacobian matrix, block-by-block, partitioned over all processes. The PETSc distributed Jacobian object can be assembled in parallel without any inter-processes data movement. Then the Krylov accelerator GMRES is wrapped around Hypre’s algebraic multigrid (BoomerAMG) preconditioner in order to solve the linear system and return the partitioned Newton correction to NPHASE-CFMD. This is not an ultimate implementation, since it doubles the memory requirement of the largest workingset in the problem (the Jacobian), but an academic one.

Although algebraic multigrid, as a solver or preconditioner, can be quite robust for unstructured problems [6], it is very expensive to set up, since many successively coarser levels of the original matrix must be derived by blackbox algebraic means (without any need to query the original problem or mesh geometry). Hypre’s BoomerAMG has shown good scalability beyond 100,000 processors [7], and the computational expense of the set up can be amortized over many right-hand sides in many relevant contexts. In this study, we will experiment on the linear solver scalability and the effect of using different splitting or decoupling strategies of the Jacobian A, to build up AMG preconditioners for the Krylov accelerator, on the linear solver performance.

## 1. Problem Description

Performing Reynolds-averaged Navier-Stokes (RANS) simulations of turbulent flow inside the large computational domains at macro-scale requires the solution of a nonlinear system of partial differential equations (PDEs), representing the conservation laws for multi-phase fluid. For steady-state problems, discretizing the PDE on a finite volume mesh results in a set of nonlinear algebraic equations, $F(x) = 0$, which can be solved using a variant of Newton's method. At iterate $x^n$, in order to obtain the next iterate $x^{n+1} = x^n + \omega\delta x^n$, Newton requires the correction $\delta x^n$ from solving a large sparse system of linear equations of the form, $A\delta x^n = b^n$, where $A = [a_{ij}]$ is the Jacobian of $F(x)$ evaluated at $x^n$ and $b^n$ is the negative of the residual of the nonlinear system, namely, $b^n = -F(x^n)$. This iteration is typically damped through an under-relaxation factor $\omega$, $0<\omega < 1$, when the underlying continuous governing system is nonlinearly stiff and the current iterate is far from steady state.

The objective of the present work is to find a robust and scalable parallel linear solver for the Newton systems that arise in the application of the NPHASE-CMFD code to nuclear reactor configurations [8, 9]. A physical test problem of reference is concerned with 3-D simulations of two-phase flow and heat transfer in the coolant channels of a Gen. IV Sodium Fast Reactor (SFR) during a channel blockage accident. The geometry is discretized with the Finite Volume Method on unstructured hexahedral and tetrahedral meshes, as required by the irregularity of the computational domain. Cells in the mesh have only interfacial connectivity, namely for a hexahedral cell, the stencil width is 7 and for a tetrahedral cell, the stencil width is 5. Moreover, the resulting linear system lacks symmetry and positive definiteness due to the unstructured mesh. Also, the Jacobian *A* is always refreshed over the course of the each nonlinear (outer Newton) iteration. A ripe strategy for future investigation would be to reuse the multilevel components of a preconditioner for one Jacobian on several subsequent Newton steps. This strategy is related to, but one stage better than the "modified Newton method" often used in multicomponent problems in combustion [10], since it does not compromise on a fresh Jacobian, but only on the preconditioner to the Jacobian, which is thus amortized. A weak preconditioner does not retard the asymptotically quadratic nonlinear convergence of a true Newton method.

Currently, upwards of 90% of the computation time of an NPHASE-CMFD simulation can be spent on solving the linear system mentioned above, making this issue the bottleneck of the overall simulation process. Therefore, the performance of the linear solver is of great importance, and a built-in scalable robust linear solver is expected to significantly accelerate the calculations. The structure of Jacobian in NPHASE-CMFD is inherited from the ordering of the unknowns and the structure of the FVM stencil. It is illustrated for a 3-component fluid in three dimensions in **Figure 1**. Detailed discussion of the discretization of NPHASE-CMFD is beyond the scope of this paper; see [4].

Although direct methods such as LU decomposition are robust on solving the nonsingular though generally ill-conditioned linear systems generated by the discretization of PDEs, iterative methods are most often the favored choice for linear equations involving a large number of variables in high dimensions due to their great advantages of memory and per-iteration implementation scalability. Krylov subspace methods make up an important family of iterative solvers. This class includes the GMRES [11] method and its variants, which is robust for nonsymmetric matrices. However, with GMRES a restarting strategy [7] should always be adopted when even the preconditioned problem has challenging condition number, to avoid a prohibitive linear increase of memory usage with the iteration counts in the algorithm.

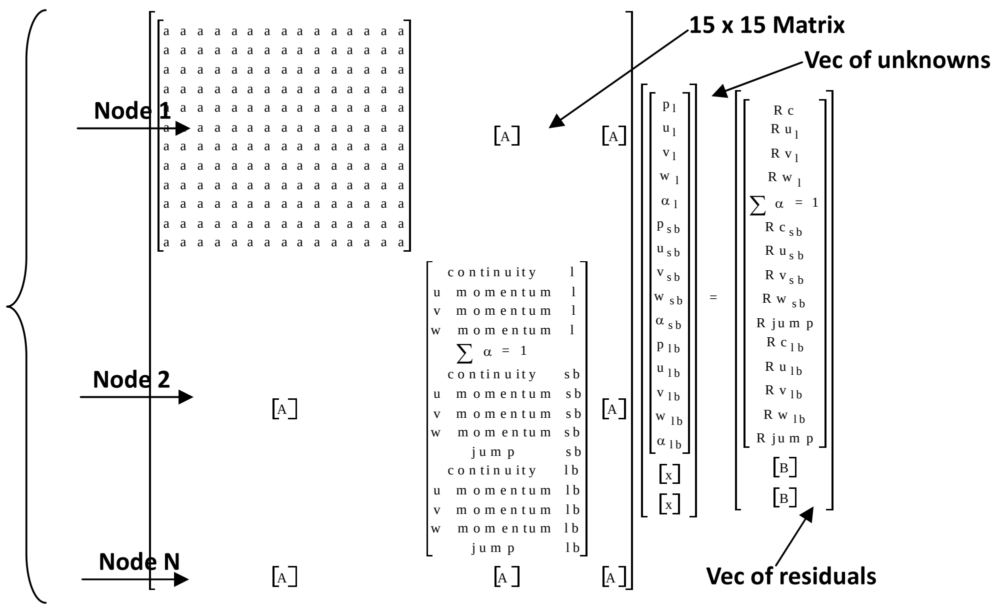

Figure 1. The linear system structure in NPHASE-CMFD for the 3-component fluid.

Numerical L In practice, for an ill-conditioned linear system combining a Krylov subspace method with a preconditioner is essential to reduce the condition number of the resulting coefficient matrix, alleviate the effect of round-off errors, and enhance the convergence rate. Yet, a stand-alone scalable solver is not sufficient for a satisfying implementation. Therefore, we also discuss some observations from the implementation aspect about the data interface between the NPHASE and PETSc.

## 2. ibraries

In the PETSc library, several Krylov subspace solvers and sophisticated preconditioners are available, including the generalized minimal residual method (GMRES) [10] and Flexible GMRES (FGMRES) [12]. Krylov solvers may be thought of as algebraic Galerkin methods. They choose the best solution within an expanding trial space, as enforced by (in the case of GMRES) minimizing residuals against a test space in some norm. FGMRES is a variant of the GMRES method with right preconditioning that enables the use of a different preconditioner at each step of the Arnoldi process. We also note that FGMRES can bring users the ease of setting up the true residual norm as stopping criteria for preconditioned systems. Therefore, in this application we choose FGMRES whenever we wish to monitor the true (unpreconditioned) residual norm as a stopping criterion for the linear solver.

Regarding the choice of the preconditioner, we use the PETSc preconditioning directive PCFieldSplit, which has been created by combining separate preconditioners for individual fields or components. Within each field created by PCFieldSplit, users have the freedom to choose among a variety of preconditioners. For this application, we have been using algebraic multigrid preconditioners from Hypre for each field and all the preconditioners are coupled in a multiplicative way.

To be more specific, when we build up the preconditioner, we apply PCFieldSplit to separate each fluid component, and then apply a Krylov solver object to perform the action of inverse of each subfield. This method can be seen as a multiplicative composition of preconditioners built up by submatrices. The PCFieldSplit preconditioner can separate coupled degrees of freedoms within each mesh cell, namely, each variable in the conservation laws. For instance, in a single-phase fluid, we have five degrees of freedom associated with each cell. For 3 mesh cells, using lexicographic ordering, the unknowns have the zero-origin index set: {{0,...4},{5,...9},{10,14}}. The preconditioner can separate each degree of freedom by introducing 5 independent index sets, namely {0,5,10}, {1,6,11}, {2,7,12}, {3,8,13}, {4,9,14}. Clearly, this preconditioner intrinsically has an emphasis on the block structure and thus it requires the matrix to be partitioned on block level. So the default component wise partitioning of the matrix in PETSc is not applicable. Therefore, we have to explicitly partition the matrix on block level over all processes.

In terms of computational complexity, our solver has been shown to be linear with problem size, *N*, of the dimension of the Jacobian *A* or the full number of degrees of freedom (DOFs) on all processes. Or equivalently, for fixed DOF per core, our solver uses almost constant time for whatever problem sizes. Indeed, for each pure GMRES iteration the complexity is *O(r*N)* on a sparse matrix of size *N* with nonzero ratio per row of *r/N*. Therefore, for each preconditioned GMRES iteration on a sparse matrix of size *N*, the complexity will be *O(r*N*+sparse preconditioning complexity*)*. And the complexity of the sparse preconditioner, which is the complexity of the BoomerAMG in this context, increases with the problem size linearly [10], namely, *O(c*N)* for some positive constant *c*. Therefore our solver has a linear complexity per iteration.

The convergence criterion of an iterative method is generally related to the linear system residual $r^m = b - Ax^m$, where $m$ denotes the $m^{th}$ iteration. In this application, we choose the convergence criterion as, $\|r^m\|_1 < \varepsilon\|b\|_1$, where $\varepsilon$ is the relative tolerance and we set $\varepsilon = 10^{-3}$ when the current iterate is far away from steady state solution. In some instances, we may set $\varepsilon = 10^{-6}$ for specific comparison purposes. For production runs, a standard $L^1$ checking mechanism is not utilized, since it needs to build up the residual vector explicitly in FGMRES and is unnecessarily expensive.

## 3. Scalable data interface from NPHASE to PETSc

The choice of using the preconditioner PCFieldSplit in PETSc, as described in Section 2, has led us to the choice of partitioning the Matrix *A* with respect to the block structure, which, in turn, determines the communication pattern among different processes in the stage of assembling the parallel matrix object in PETSc.

In the NPHASE code, for assembling the Jacobian of the Newton's System, there are three data structures on each local process: the node-based, the face-based, and the inter-processor partition-based data structures.

The node-based data structure is a contiguous array holding all matrix coefficients from the Finite Volume Method (FVM) calculation for the self-interaction of local nodes (FV cell centers), and it contributes to diagonal elements of the diagonal partition of the PETSc matrix object MATMPIAIJ on the local process. The face-based data structures are two contiguous arrays, one of which is holding all matrix coefficients from one direction in the FVM calculation for the face connectivity of local nodes and the other one of which is holding coefficients from the other direction. They contribute to lower and upper diagonal elements of the diagonal partition of the PETSc matrix object MATMPIAIJ on the local process, respectively. The inter-processor partition-based data structure is a contiguous array holding all the matrix coefficients from the FVM calculation for the face connectivity of local nodes and remote nodes that are located on other processes. They contribute to off-diagonal elements of the PETSc matrix object MATMPIAIJ on the local process.

Since the mesh is unstructured, neither the face-based data nor the inter-processor partition based data is sorted. For a specific local node (cell center), one can obtain the count of local faces, and thus set up the PETSc matrix pre-allocation exact. However, it is difficult to fill in the matrix values of a full block row for a particular local node, since neither of the face based data and the inter-processor partition based data is sorted and the matrix coefficient is stored in an order corresponding to the ordering of the face-based data structure and the inter-processor partition based data structure, in other words, the matrix coefficients are stored (non-contiguously) almost corresponding to the compressed row storage (CRS) format. To be more specific, one may get access to the data of a face sharing node 0, far behind all the data of the node 1.

Based on this consideration, we loop through each of the three data structure arrays and set up the matrix data for each block entry of the block matrix. This is a mode of contiguous read to set up non-contiguous blocks inside MATMPIAIJ. An obvious pitfall here is that PETSc MatSetValuesBlocked routine is called as many times as the number of nonzero blocks in MATMPIAIJ. The code implemented in this way could be inefficient based on subroutine call overhead on some systems. However, in practice, we have not observed any performance suffering from the data interface.

## 4. Test problems

In the present application, the algebraic problem size is completely specified by the mesh size and the number of fluid phases that are coupled in the simulation. For each fluid phase $j$ at each mesh cell $k$ in the multiphase flow, we need 5 primitive variables to characterize the flow, namely: the 3D fluid velocity $u_k^j, v_k^j, w_k^j$, pressure $p_k^j$, and volume fraction $a_k^j$. In other words, for each phase, each mesh cell corresponds to a degree of freedom (DOF) of 5.

Most numerical experiments have been performed on the Opteron Blade Cluster located at the Computational Center for Nanotechnology Innovations (CCNI), RPI. This cluster consists of 462 IBM LS21 blade servers with two dual-core 2.6 GHz Opteron processors, gigabit Ethernet and Infiniband interconnects, and system memory in eight, twelve and sixteen gigabyte configurations.

The computational speed improvements to the linear solvers and thus the full simulation will be used for various multiphase problems such as a loss-of-flow accident scenario in a Gen. IV reactor [8, 9]. As an illustration, Figure 2 shows the propagation of the fission gas along the reactor coolant channel.

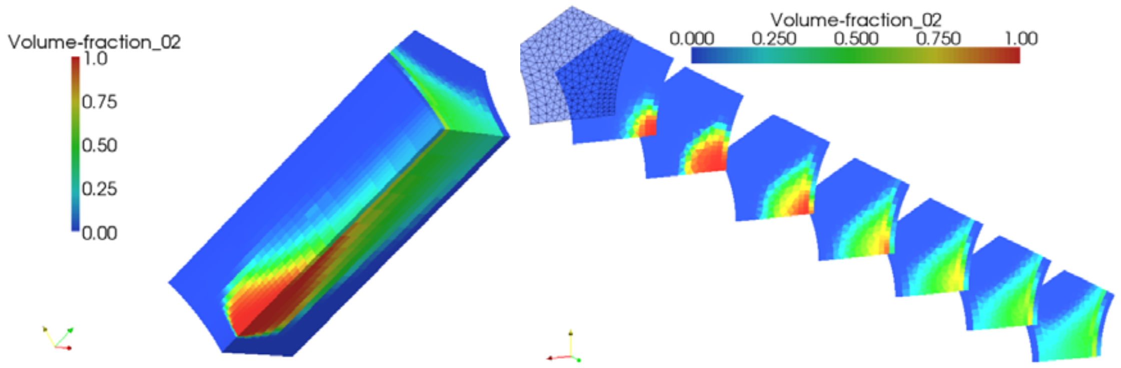

Figure 2. Simulation of fission gas jet propagation following a loss-of flow accident [8].

## 4.1 Test problems for solver verification.

The first test problem consists of $5 \times 10^3$ mesh cells, simulating a single-phase fluid. So, each cell corresponds to five degrees of freedom (5 physical fields, each field with 5 quantities). The second test problem consists of $5 \times 10^3$ mesh cells, simulating a two-phase flow. So each cell corresponds a degree of freedom 10 (5 physical fields, each field with 5 quantities). The third test problem consists of $5 \times 10^3$ mesh cells, simulating a five-phase fluid flow. So each cell corresponds a degree of freedom 25 (5 physical fields, each field with 5 quantities). The nonlinear residual history versus iteration counts is shown in Figure 3 through Figure 5.

In Figure 3 and Figure 4, the nonlinear residual history is shown for the system of conservation laws for the single phase and two-phase flows, respectively, after 600 iterations, with the inner Newton system stopping criteria given as $\varepsilon = 10^{-3}$. It can be seen that NPHASE-CMFD and PETSc have very similar nonlinear convergence paths. This is further confirmed in Figure 5, which describes the nonlinear residual for a five-field fluid. From these plots, we can conclude that the PETSc solver and NPHASE solver follow similar convergence paths.

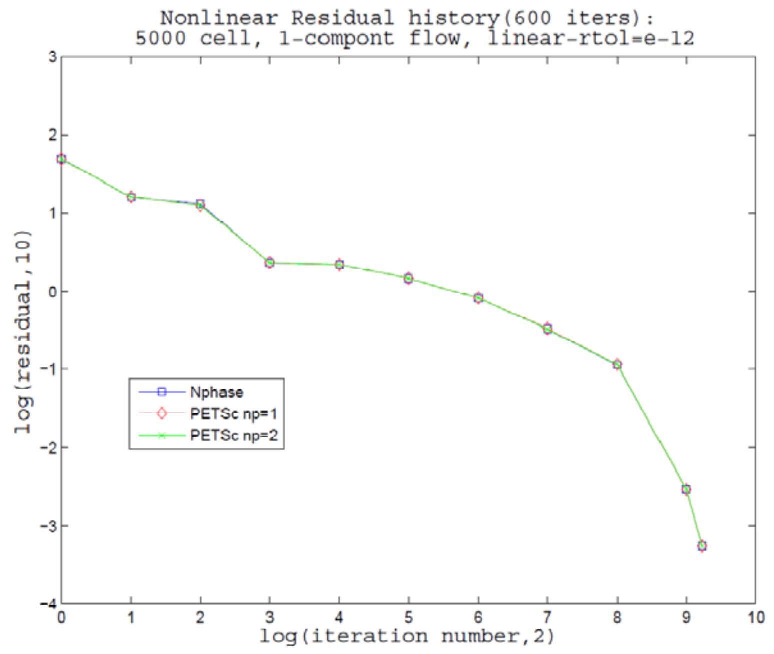

Figure 3. Nonlinear residual history over 600 iterations for 1 component, 5000 cell case.

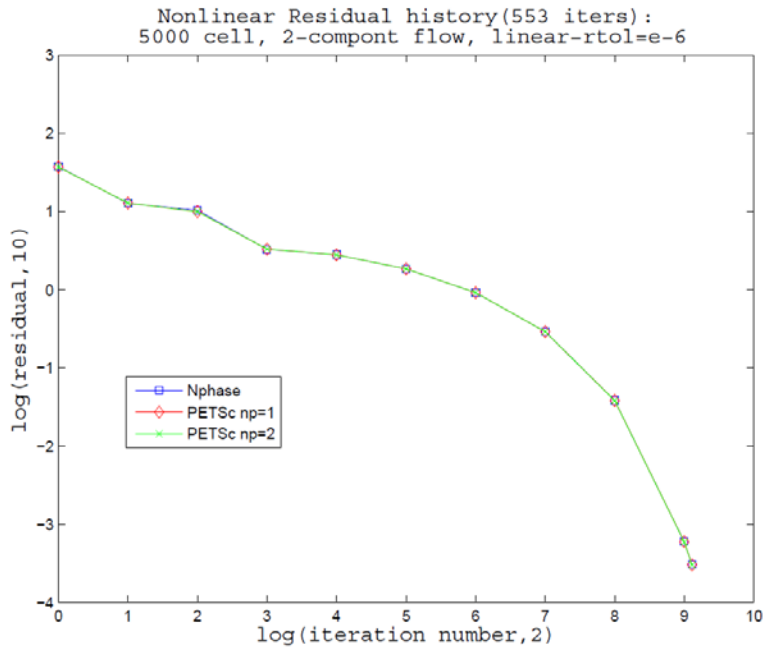

Figure 4. Nonlinear residual history over 553 iterations for 2 components, 5000 cell case.

In general, the NPHASE-CMFD code is used to solve for complex flows of multiphase fluids. A physical test case used to compare the results of the PETSc solver with the NPHASE-CMFD solver is similar to that shown in Figure 2, but this time it will involve two-phase simulations for a large three-dimensional section of reactor fuel assembly. An overview of the geometry and the initial predictions using the original NPHASE-CMFD solver are shown in Figure 6.

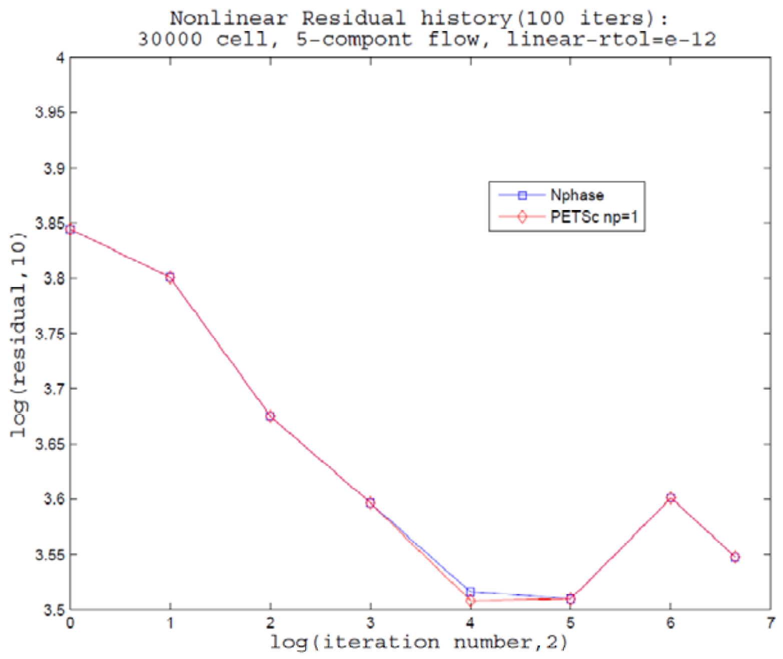

Figure 5. Nonlinear residual history over 100 iterations for 5-component, 30000-cell case.

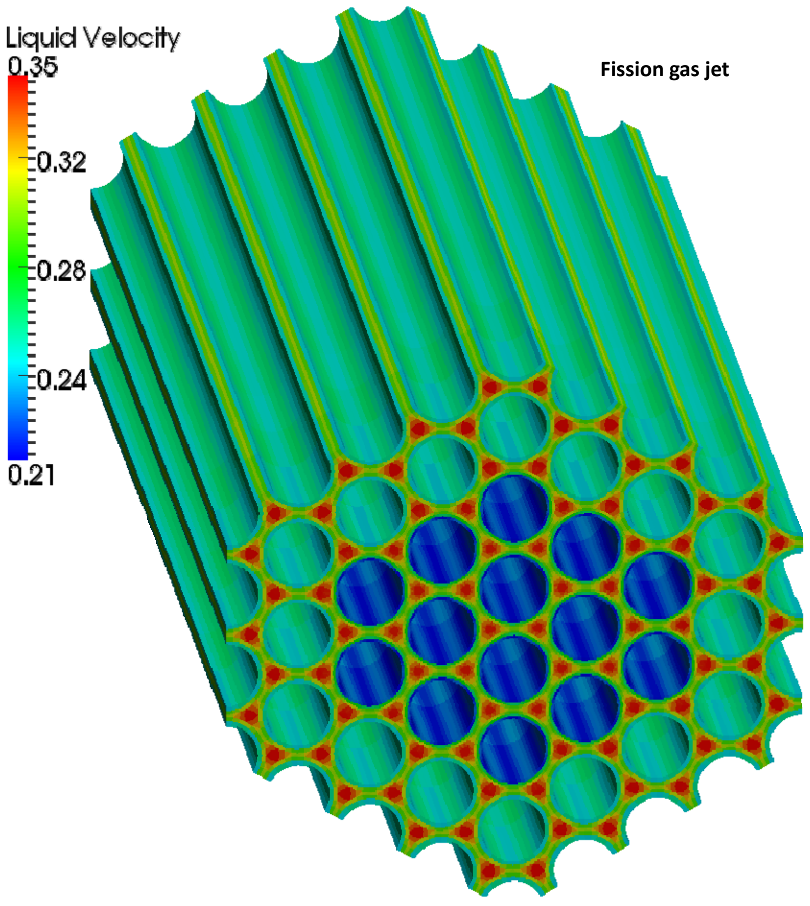

Figure 6. The geometry for the proposed physical test case.

## 4.2. Test problems for solver performance (scaling analysis)

### 4.2.1 Weak scaling.

The purpose of this section is to study the weak scalability of the solver. For each test series, namely, 1-phase duct fluid, 2-phase duct fluids, 3-phase duct fluids and 4-phase duct fluids, we perform the weak scaling analysis using different coupling strategies. More specifically, for n-phase fluids, for each cell, the parallel block matrix will have the index set {1,2,3,4,5… 5n-4,5n-3,5n-2,5n-1,5n}. We call the decoupling strategy of using the index set {1}, {2}, {3},… {5n-1}, {5n} “full-splitting,” the decoupling strategy of using the index set {1,2,3,4}, {5},… {5n-4,5n-3,5n-2,5n-1}, {5n} “partial-splitting,” and the decoupling strategy of using the index set {1,2,3,4,5},… {5n-4,5n-3,5n-2,5n-1,5n} “phase splitting.” After choosing a specific splitting strategy, we can then use PETSc preconditioning directive PCFieldSplit to impose the AMG preconditioner on the corresponding field. The whole process of assembling preconditioner matrices corresponding to each index set and performing the parallel Matrix Vector Multiplication is fully automatic.

A series of test problems has been constructed to examine the solver scalability. The geometry of the problem is a simple duct with a cross section height of 1 meter and width of 5 meters. See Figure 7. The height is discretized into 10 elements and the width is discretized into 50 elements. These discretizations remained constant for all case. The length of the duct is adjusted to maintain cubic hexahedral elements while allowing the total number of elements to vary. It should be pointed out that the duct geometry is particularly useful for testing purposes since it provides a convenient platform to alter mesh element counts while not affecting other mesh properties that can impact numerical convergence, such as element aspect ratios. One end face of the duct is designated as an inflow boundary and the opposing face is designated as a pressure outflow boundary. The remaining 4 outer faces of the duct are designated as no-slip walls. Seven different meshes were built with the following number of elements (in millions): 1.5, 3, 6, 9, 12, 18, and 24. For scaling studies, these meshes were decomposed to run on 16, 32, 64, 96, 128, 192, and 256 processors, respectively. This combination of mesh and processors results in approximately 4.7E5 elements per processor.

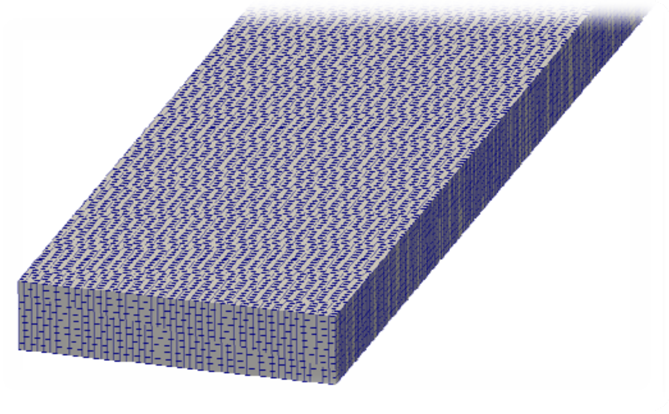

Figure 7. The geometry of the duct fluid mesh.

The first benchmark in the weak-scaling analysis was set up for a single-phase duct fluid. Starting with a problem size of 1.5e6 mesh cells using 16 processors. We keep the DOF per core fixed as 4.7E5 along the weak scaling path. The test run with full splitting strategy is listed in Table 1 and Figure 8. In fact, full splitting is the most expensive one in term of memory usage and complexity, but should converge faster. We will include the test results regarding the convergence rate and other splitting strategies for 1-phase fluid in our final report as well as the results for other multiphase fluids.

Table 1. Weak scaling results with full splitting.

| Mesh Size | NP | Wall Time |
|---|---|---|
| 1.5e6 | 16 | 1.35e+02 |
| 3e6 | 32 | 1.48e+02 |
| 6e6 | 64 | 1.99e+02 |
| 9e6 | 96 | 2.56e+02 |
| 12e6 | 128 | 2.94e+02 |

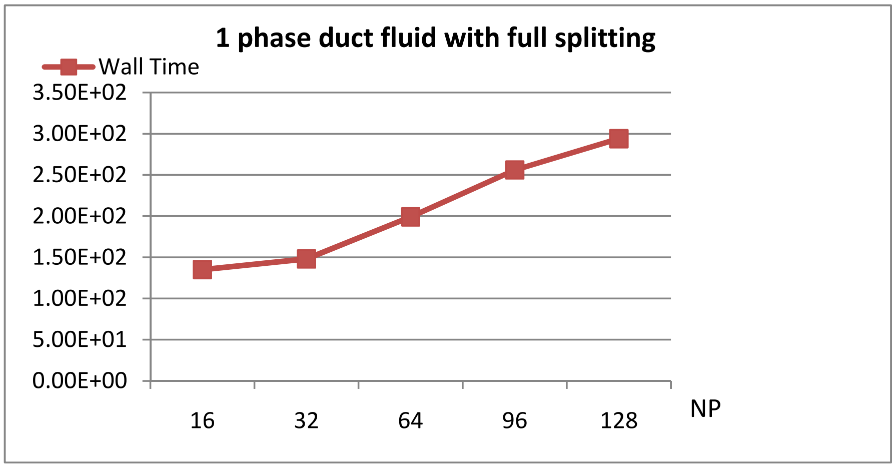

Figure 8.Weak scaling results with full splitting.

## 5. Conclusions

A strategy has been discussed and demonstrated of building a robust and scalable parallel linear solver for the Newton correction system resulting from the discretization of the conservation laws system of multiphase fluids. The sensitivity of the solver performance with respect to the preconditioner built on different splitting strategy has been studied, together with the weak scaling of promising methods. Although this is still work in progress, the results obtained so far have already helped us to identify several major issues which will be critical for the development of a complete fast and robust multiprocessor solver. Future work will include scalability measurements on machines with thousands of computing nodes (such as IBM Blue Gene, Cray XT5 and Jaguar) as well as a study of reusing the Jacobian or preconditioners to the Jacobian on the nonlinear solver level.

Further testing and reactor applications of the methodology outlined in this paper will include the NPHASE-CMFD simulations, performed using the ORNL supercomputers of single and two-phase flow in reactor fuel assemblies for gradually increasing computational domains, which will focus on the development of the Virtual Reactor for the CASL project.

## 6. References

[1] S. Balay, K. Buschelman, W. D. Gropp, D. K. Kaushik, M. G. Knepley, L. Curfman McInnes, B. F. Smith, and H. Zhang, "PETSc", 2011. http://www.mcs.anl.gov/petsc.

[2] R. Falgout, A. Baker, V. Henson, U. Yang, T. Kolev, B Lee, J. Painter, C. Tong, and P. Vassilevski, "Hypre", Journal of Citation, 2011. https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html.

[3] X. S. Li, "SuperLU, Sparse Direct Solver", 2011. http://crd.lbl.gov/˜xiaoye/SuperLU.

[4] Interphase Dynamics LLC, "NPHASE-CMFD Program Manual", version 3.1.3, 2008.

[5] G. Karypis, "Metis", 2011, http://www.cs.umn.edu/~metis.

[6] V. Henson and U. M. Yang, "BoomerAMG: A parallel algebraic multigrid solver and preconditioner", Appl. Numer. Math., Vol. 41, pp. 155-177, 2002.

[7] A.H. Baker, R.D. Falgout, T.V. Kolev, and U.M. Yang, "Scaling hypre's Multigrid Solvers to 100,000 Cores", LLNL-JRNL-479591, 2011.

[8] I. A. Bolotnov, S. P. Antal, K. E. Jansen and M. Z. Podowski, "Multidimensional Analysis of Fission Gas Discharge following Fuel Element Failure in Sodium Fast Reactor", The 13th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-13), Kanazawa City, Ishikawa Prefecture, Japan, September 27-October 2, 2009.

[9] I. A. Bolotnov, F. Behafarid, D. R. Shaver, S. P. Antal, K. E. Jansen, R. Samulyak, H. Wei, and M. Z. Podowski, "Multiscale Computer Simulation of Fission Gas Discharge During Loss-of-Flow Accident in Sodium Fast Reactor", Proc. Computational Fluid Dynamics for Nuclear Reactor Safety (CFD4NRS-3), Washington, D.C., USA, September 14-16, 2010.

[10] D. E. Keyes and M. D. Smooke, "A parallelized elliptic solver for reacting flows", in "Parallel computations and their impact on mechanics: Proceedings of the Symposium, ASME Winter Annual Meeting, Boston, pp. 375-402, 1987.

[11] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems", SIAM J. Sci. Stat. Comput., Vol. 7, pp. 856-869, 1986.

[12] Y. Saad, "A Flexible Inner-Outer Preconditioned GMRES Algorithm", SIAM J. Sci. Comput., Vol. 14, pp. 461-469, 1993.