The 14th International Topical Meeting on Nuclear Reactor Thermal Hydraulics, NURETH-14
Toronto, Ontario, Canada, September 25-30, 2011

NURETH14-266

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 using implicit nonlinear treatment of the governing equations and in parallel, which is a very 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 obtained from the discretized system of governing conservation equations in NPHASE-CMFD. The overall long-tem objective of this work is to extend the NPHASE-CMFD code into a fully-scalable solver of multiphase flow and heat transfer problems, applicable to both steady-state and stiff 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). This campaign appropriately begins with the linear algebraic equation solver, which is traditionally a bottleneck to scalability in PDE-based codes. The computational complexity of the solver is usually superlinear in problem size, whereas the rest of the code, the "physics" portion, usually has its complexity linear in the problem size.

Introduction

The required fidelity of modelling and resolution of computer simulations of complex physical phenomena must normally meet strict criteria in order to deliver the expected predictive capabilities for scientific discovery and engineering design by taking advantage of leading-edge computer platforms. However, the linear equation solver very often 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 grid functions and linear and nonlinear operators on them) and a rich selection of algorithmic building blocks for linear solvers, nonlinear solvers, and time-steppers 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 utilized for a wide variety of multiphase flows 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 time step 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 working set 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 study the linear solver scalability and the effect of using different splitting or decoupling strategies of the Jacobian matrix 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 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 the 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 δx^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 ω, $0<\omega < 1$, when the underlying continuous governing system is nonlinearly stiff.

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 couples at most 7 nonzero blocks per row and for a tetrahedral cell, at most 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, up to 50% 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 multiphase flow in three dimensions in Figure 1. A detailed discussion of the

discretization of NPHASE-CMFD is beyond the scope of this paper, but can be found elsewhere [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 favorite choice for linear equations involving a large number of variables in high dimensions. This is 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 nonsymmetrical matrices. However, with GMRES a restarting strategy [7] should always be adopted whenever the preconditioned problem has challenging condition number, to avoid a prohibitive linear increase of memory usage with the iteration counts in the algorithm.

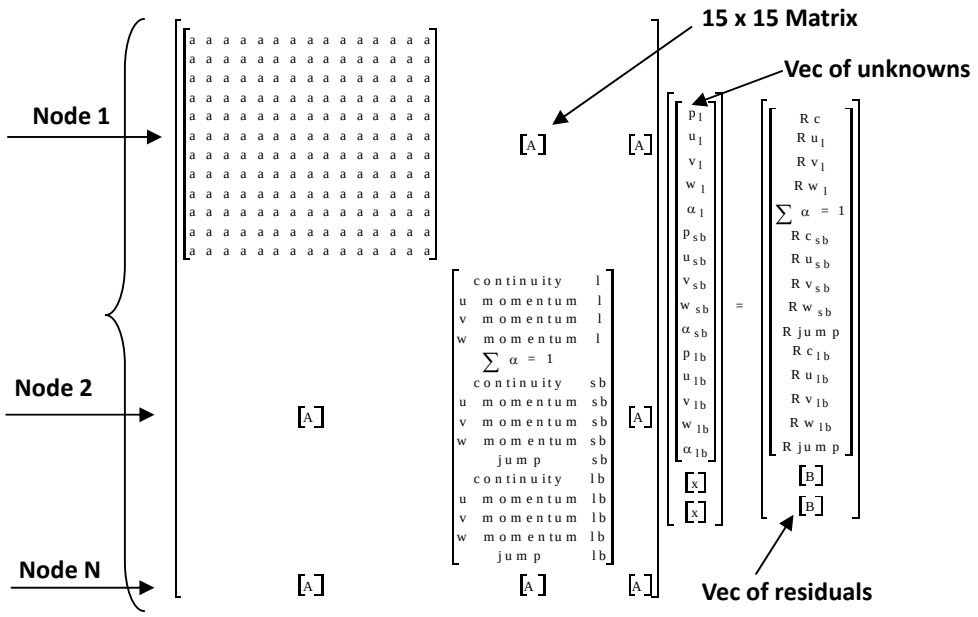

Figure 1. The linear system structure in NPHASE-CMFD for the 3-component fluid.

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. Numerical Libraries

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. For linear systems as difficult as those that arise in unstructured grid multiphase nuclear analyses, few of the standard preconditioners apply. Those that exploit geometric regularity, like geometric multigrid, are not relevant. Those that are not multilevel, such as flavors of ILU, SOR, and sparse approximate inverse, do not retain good convergence for very large scale meshes. Various field-blocked flavors of AMG are called upon for robustness herein. There is a compromise between the fully split scheme, which is asymptotically optimal in the limit of high diffusion within a field, and the partially split schemes, which allow greater fill-in by capturing more of the coupling in the Jacobian matrix within the preconditioner.

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. Particularly, the hybrid modified independent set (HMIS) algorithm with a strength threshold 0.25 in Hypre has been utilized to construct the algebraic coarse grid, which in turn requires distance-two interpolation operators [7]. In our multiphase fluid test, the “extended+i” interpolation is quite robust and scalable.

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 freedom 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 almost 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 almost linearly [10], namely, *O*(*c*N*) for some positive constant *c*. Therefore, our solver has an almost 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 ε 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 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 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 nor the inter-processor partition based data is sorted. On the other hand, 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 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 fluid component, 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, except for a few specific cases where another attribution was used. 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 1**Figure 2 shows the propagation of fission gas along the reactor coolant channel following a loss-of-flow accident

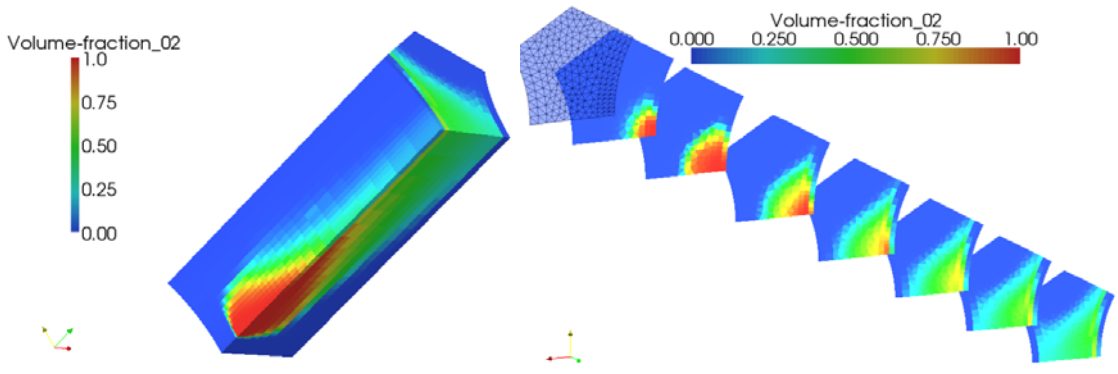

Figure 2. Simulation of fission gas jet propagation following a loss-of flow accident [8].

4.1. Test problems for solver verification

The first test problem consists of 5×10^3 mesh cells, simulating a single-phase fluid. Therefore, each cell corresponds to five degrees of freedom (5 physical fields, each field with 5 quantities). The second test problem consists of 5×10^3 mesh cells, simulating a two-phase flow. Each cell corresponds a degree of freedom 10 (2 physical fields, each field with 5 quantities). The third test problem consists of 5×10^3 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.

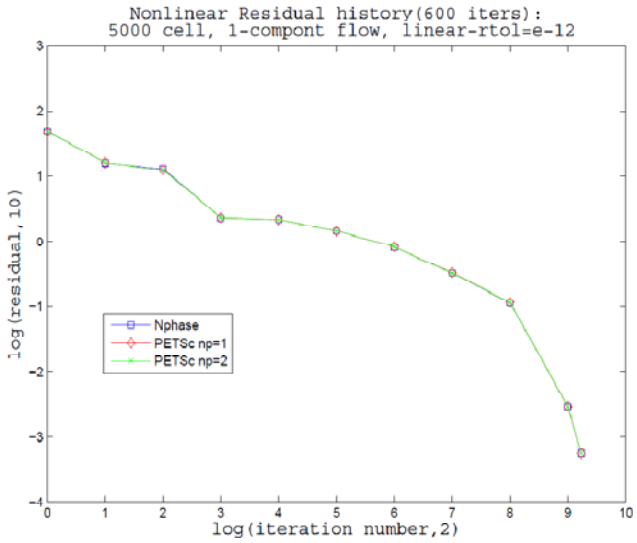

Figure 3. Nonlinear residual history over 600 iterations for 1 component, 5000 cell case.

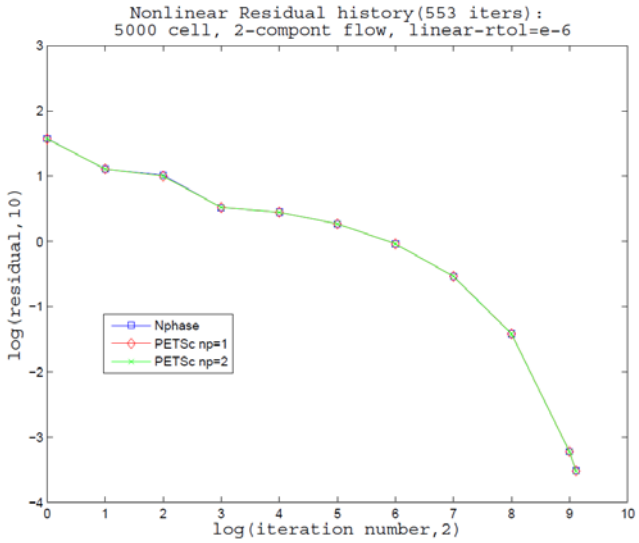

Figure 4. Nonlinear residual history over 553 iterations for 2 components, 5000 cell case.

Figure 3 and Figure 4 show the nonlinear residual history for the system of conservation laws for 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.

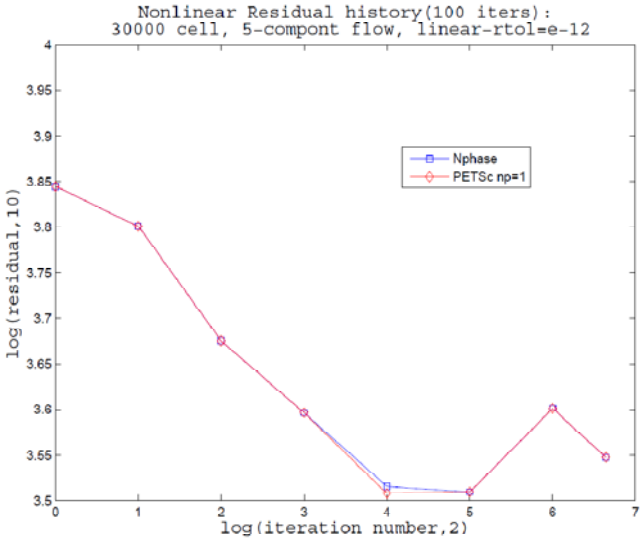

Figure 5. Nonlinear residual history over 100 iterations for 5-component, 30000-cell case.

In general, the NPHASE-CMFD code is used to solve complex flows problems for 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.

4.2. Test problems for solver performance (scaling analysis)

The purpose of this section is to study the weak scalability of the solver. For each test series, namely, 1-phase duct fluid, 1-phase fluid in reactor rod geometry, and 2-phase fluids in reactor rod geometry, 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-1," the decoupling strategy of using the index set {1}, {2,3,4}, {5},… {5n-4}, {5n-3,5n-2,5n-1}, {5n}"partial-splitting-2," 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 automated "under the hood" as to its ordering and MPI implications.

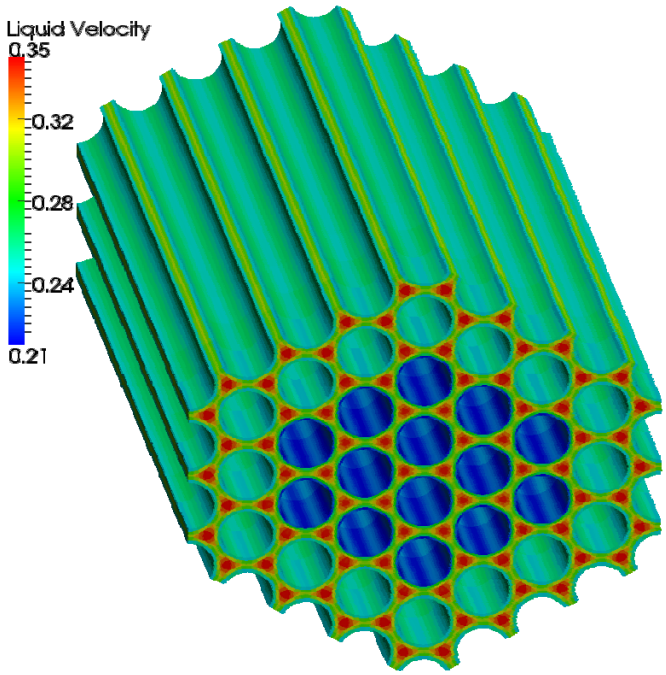

Figure 6.The geometry for the proposed physical test case.

The first benchmark has been aimed at testing solver robustness regarding different splitting strategies. This test problem is simulating 5-phase fluids with mesh cells. Each time, we run the linear solver for the first Newton iteration with different splitting strategies and with the linear solver tolerance , which is the unpreconditioned relative residual norm. The maximum GMRES iteration allowed was 100. This particular test was done on a Dell workstation with two quad-core Intel Xeon CPUs. The test results are shown in Table 1. Both full-splitting and partial-splitting-2 have converged in 39 iterations. However, partial-splitting-1 and phase-splitting failed to reach the linear solver tolerance within 100 iterations, with the residual stagnating after some iteration. We observed similar solver behavior in other test cases and even divergence on more difficult problems. It is evident that the partial-splitting-1 and phase-splitting strategy are having difficulties in removing low frequency errors on the algebraic coarse grid. The decoupling between velocity and pressure is essential to reduce the low frequency error for all the problems presented in this paper. Therefore, the results for other test cases in Table 1 pertain to the full splitting and partial-splitting-2 strategies.

The second test problem 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 remain constant for all cases.

Table 1. Robustness of different splitting strategies.

	Iteration Count	Wall Time
Full-splitting	39	8.90e+00
Partial-splitting-1	100	9.40e+01
Partial-splitting-2	39	1.03e+01
Phase-splitting	100	1.19e+02

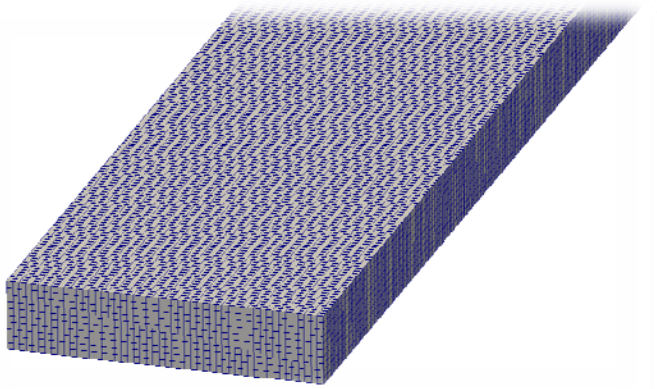

Figure 7. The geometry of the duct fluid mesh.

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 DOF per processor. Strictly speaking, varying physical duct length is not weak scaling in the mathematical sense, which should use a finer mesh on the same physical geometry. However, it is a natural one-dimensional form of weak scaling in the physical sense and does not lead to a change in the aspect ratio of the discretization from doubling to doubling.

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 kept the DOF per core fixed as 4.7E5 along the weak scaling path. The test results with full-splitting and partial-splitting-2 strategy are listed in Table 2 and Figure 8. First, these two splitting strategies have same iteration counts when reducing the linear residual to three orders of magnitude regarding all test cases. It is evident that partial-splitting-2 strategy is more expensive than full-splitting strategy for small size problems. However, the full-splitting strategy is less scalable than the partial-splitting-2 strategy. For the partial-splitting-2 strategy, the weak scalability is acceptable up to 18E6 mesh cells. But the performance starts to deteriorate at the mesh size 24E6.

Table 2. Single-phase duct fluid with full-splitting and partial-splitting-2 strategies.

# of Mesh Cells	NP	Full-splitting		Partial-splitting 2	
		Iter. Count	Wall Time	Iter. Count	Wall Time
1.5e6	16	7	1.93e+01	7	3.07e+01
3e6	32	8	2.47e+01	8	3.49e+01
6e6	64	8	2.74e+01	8	3.70e+01
9e6	96	8	3.32e+01	8	4.35e+01
12e6	128	8	3.66e+01	8	4.49e+01
18e6	192	8	7.52e+01	8	5.38e+01

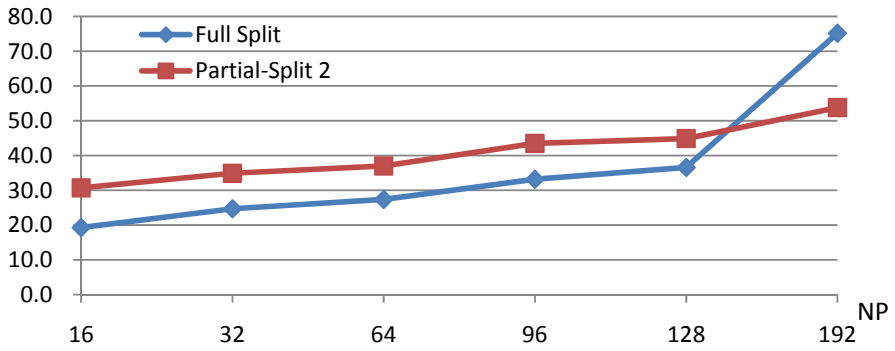

Figure 8. Weak scaling results with full and partial-splitting-2 strategies for a single-phase phase duct flow.

Similar performance has been observed for the tests using the fuel assembly of a proposed sodium-cooled fast reactor core. Figure 9 shows the cross-sectional view of fuel channels for a section of the assembly. Thirty complete fuel rods of 8mm diameter and 10.4 mm pitch are encompassed by the fluid domain along with several surrounding partial rods at the boundaries. The length of the domain was varied to control the aspect ratio of the elements. This assured a constant DOF per core and helped to preserve the aspect ratio of the elements.

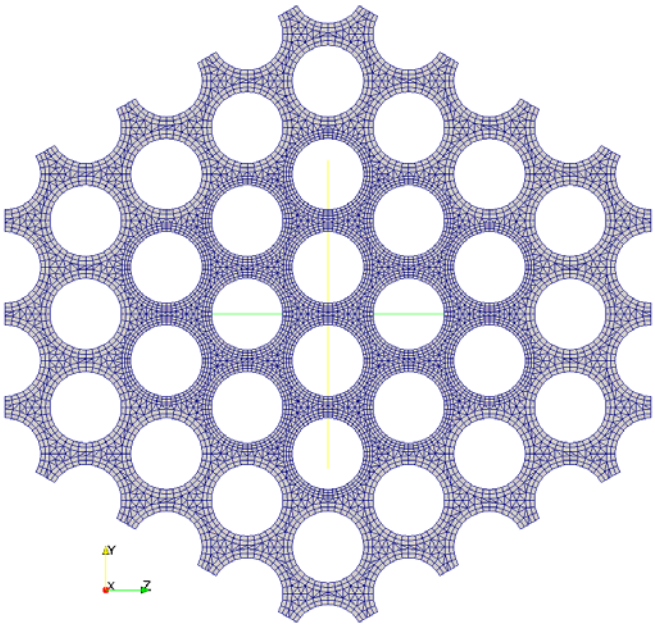

Figure 9. Cross-section of fuel rod geometry with computational grid overlaid.

Two series of tests have been performed, again one for a single-phase flow, the other for a two-phase flow. The results for single phase flow are shown in Table 3 and Figure 10, and the corresponding results for the two-phase flow case are shown in Table 4 and Figure 11. It was observed that, for a given run, a large fraction of the computational time has been spent on the function calls KSPSetup and KSPSolve. For the two-phase flow case, the timing on KSPSetup was 8 times larger, and on KSPSolve was 1.6 times larger, whereas the problem itself was 12 times bigger. In contrast, using the partial-splitting-2 strategy, the timing on KSPSetup was 4 times larger and on KSPSolve was 1.3 times larger. As a result, the partial-splitting-2 strategy turns out to be more scalable. Furthermore, there seems to be some nonscalable component in the setting up stage of this particular preconditioner which damages the overall weak scalability, and this will be the subject of a future study. On the other hand, the observation just described has confirmed our hypothesis that it would be beneficial to reuse the preconditioner during a nonlinear run to avoid the expense of setting up the preconditioner at each step of the time integration, in order to achieve an overall scalability of the nonlinear solver.

Table 3. Single-phase flow in the reactor rod geometry with full-splitting and partial-splitting-2 strategies.

# of Mesh Cells	NP	Full-splitting		Partial-splitting 2	
		Iter. Count	Wall Time	Iter. Count	Wall Time
1.5e6	16	6	1.81e+01	6	1.86e+01
3e6	32	6	2.03e+01	6	2.25e+01
6e6	64	6	2.67e+01	6	2.59e+01
9e6	96	6	3.16e+01	6	3.02e+01
12e6	128	6	3.59e+01	6	3.23e+01
18e6	192	6	5.25e+01	6	4.04e+01

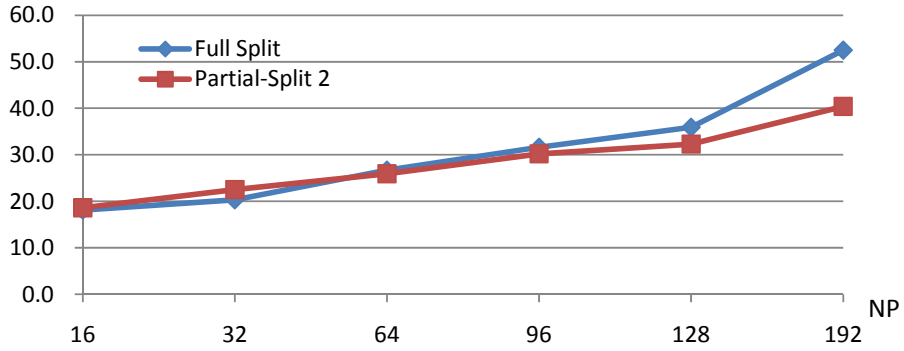

Figure 10.Weak scaling results with full and partial-splitting-2 strategies for a single-phase flow in the reactor core geometry.

In both single-phase and two-phase flow cases, the DOF per core was fixed at 1.175E5 along the weak scaling path, so that each core used same amount of memory. It should be noted that in the two-phase case, the Jacobian has a blocksize twice as large as that in the single-phase case. This limitation on the memory can be removed in the future work. We also note that the maximum GMRES iteration count of 20 has been reached in the two-phase test run. This is due to the fact that the physics of two-phase fluids is more challenging than that of single-phase fluids, and that we are using an iterate at the initial stage to test the linear solver. During a nonlinear test run, when the iterate gets closer to the steady state, the number of the GMRES iteration will be significantly reduced.

Table 4. Two-phase flow in the reactor rod geometry with full-splitting and partial-splitting-2strategies.

# of Mesh Cells	NP	Full-splitting		Partial-splitting 2	
		Iter. Count	Wall Time	Iter. Count	Wall Time
3.75e5	16	20	1.81e+01	20	2.65e+01
7.5e5	32	20	2.40e+01	20	2.80e+01
1.5e6	64	20	3.09e+01	20	3.41e+01
2.25e6	96	20	3.82e+01	20	3.88e+01
3e6	128	20	4.30e+01	20	4.32e+01
4.5e6	192	20	5.47e+01	20	5.02e+01

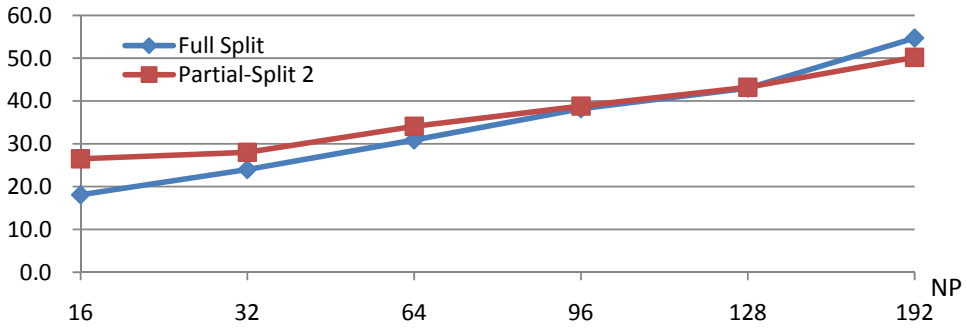

Figure 11.Weak scaling results with full and partial-splitting-2 strategies for two-phase flow in the reactor core geometry.

5. Conclusions

A strategy has been formulated and discussed for 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, with demonstrated applicability to NPHASE and likely much wider. 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 hundreds to 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. Acknowledgements

The authors gratefully acknowledge award DE-FG07-07ID14889 from the U.S. Department of Energy under the Nuclear Energy Research Initiative (NERI). We thank PETSc team and CCNI team for support. Also, special thanks are extended to Barry Smith, Edmond Chow, Dinesh Kaushik, Satish Balay, Jed Brown, Matthew Knepley, and Hong Zhang. We are also indebted to the paper reviewers for their valuable suggestions.

7. 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", 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 Users Manual", version 3.2.1, 2011.

[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.