INL/CON-08-15008
PREPRINT

MOOSE: A Parallel Computational Framework for Coupled Systems of Nonlinear Equations

2009 International Conference on Mathematics, Computational Methods & Reactor Physics

D. Gaston
G. Hansen
C. Newman

May 2009

International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)
Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009)

MOOSE: A parallel computational framework for coupled systems of nonlinear equations.

D. Gaston, G. Hansen and C. Newman
Multiphysics Methods Group
Idaho National Laboratory
Idaho Falls, ID 83415-3840
derek.gaston@inl.gov

ABSTRACT

Systems of coupled, nonlinear partial differential equations often arise in simulation of nuclear processes. `MOOSE`: Multiphysics Object Oriented Simulation Environment, a parallel computational framework targeted at solving these systems, is presented. As opposed to traditional data-flow oriented computational frameworks, `MOOSE` is founded on the mathematical principle of Jacobian-free Newton-Krylov (JFNK) solution methods. Utilizing the mathematical structure present in JFNK, physics are modularized into "`Kernels`" allowing for rapid production of new simulation tools. In addition, systems are solved fully coupled and fully implicit employing physics based preconditioning which allows for great flexibility even with large variance in time scales. A summary of the mathematics, an inspection of the structure of `MOOSE`, and several representative solutions from applications built on the framework are presented.

Key Words: framework, fully coupled, fully implicit, `libMesh`, multiphysics, nonlinear, finite-element.

1. Introduction

In reactor fuel performance simulations it is necessary to solve equations describing heat generation, thermal response of the fuel, chemical species diffusion, contact between the fuel and cladding, and nonlinear-mechanics. These equations are nonlinear with nonlinear material properties and strongly coupled. Similar challenges can be found in other areas of nuclear energy such as coolant modeling in light water reactors and simulation of heat flow through pebble bed reactors.

Classically, these types of problems are solved using operator split methods. Operator split methods solve each physics separately, passing data back and forth to achieve what is referred to as "loose coupling". Often, these methods grow out of a desire to reuse computational tools that have been created separately to solve each physics or to reduce the computational effort needed to solve the system of equations. Unfortunately, operator split methods can possess less than desirable properties [1], particularly when the coupled physics operate on disparate timescales or are very strongly coupled [2].

An alternative to solving nonlinear systems in a loosely coupled manner is to solve them simultaneously employing a tightly coupled solution procedure. This procedure involves solving for all solution variables at the same time, usually through generation of one large nonlinear algebraic system that is solved using Newton's method. Due to the inherent strongly coupled nature of many nuclear processes, MOOSE focuses on solving all systems in a fully coupled manner.

Solving nonlinear PDE's in a tightly coupled manner has its own set of challenges. One is the need to form a large linear system for use during Newton iterations. As the number of solution variables grows, the matrix holding the Jacobian entries also grows. The increasing size of the matrix and means greater memory consumption. Further, it takes time to fill in the full Jacobian and for problems with highly nonlinear material properties the true Jacobian can often be difficult to obtain. For these reasons, MOOSE utilizes Krylov methods for solving the resultant linear systems. Krylov iterative methods for solving linear systems require only a matrix-vector product and not the full matrix. Therefore, the full Jacobian is not necessary and the Jacobian-free Newton-Krylov (JFNK) scheme is utlized.

2. Jacobian-free Newton Krylov (JFNK)

The JFNK method used by MOOSE begins with writing a weak form of the specific set of PDEs to be solved and casting it into a residual function. Further, the `Boundary Conditions` must also be cast as a modified residual function. The discrete problem is

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}\,, \tag{1}$$

that is of length N, where N is the number of unknowns. The Jacobian of this system is an $N \times N$ sparse matrix,

$$\mathcal{J}(\mathbf{x}) = \frac{\partial \mathbf{F}(\mathbf{x})}{\partial \mathbf{x}}\,. \tag{2}$$

Given the Jacobian in this form, it is straightforward to express the Newton iteration,

$$\mathcal{J}(\mathbf{x}^{(k)})\,\delta\mathbf{x}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)}), \tag{3}$$

and

$$\mathbf{x}^{(k+1)} \longleftarrow\ \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}, \tag{4}$$

where the superscript k denotes the iteration count of the Newton iteration. Using Newton's method amounts to implementing a sequence of steps:

1. Form the Jacobian matrix.
2. Solve the sparse linear system (3) to obtain $\delta\mathbf{x}^{(k)}$.
3. Apply this update (4) to obtain the next iteration of the solution state vector, $\mathbf{x}^{(k+1)}$.

Even for moderately-large grids, the cost of forming the Jacobian is high and typically dominates the computation, making the above algorithm impractical for most situations. Fortunately, Krylov iterative solvers such as the generalized minimum residual (GMRES) algorithm [3], used in MOOSE to solve the Jacobian system, do not require the Jacobian matrix itself but simply the action of the Jacobian matrix on a vector. Approximating this matrix-vector product by differencing, which requires two nonlinear function evaluations, is the basis of the JFNK method. Specifically, to evaluate the matrix-vector product $\mathcal{J}(\mathbf{x}^{(k)})\delta\mathbf{x}^{(k)}$, a finite-difference approach,

$$\mathcal{J}(\mathbf{x}^{(k)})\delta\mathbf{x}^{(k)} \approx \frac{\mathbf{F}(\mathbf{x}^{(k)} + \varepsilon\delta\mathbf{x}^{(k)}) - \mathbf{F}(\mathbf{x}^{(k)})}{\varepsilon}, \tag{5}$$

is commonly used [4, 5]. Here, ε is chosen to avoid problems with machine precision.

Using this Jacobian-free approach, the dominant cost of the algorithm shifts from evaluating the Jacobian to the solution of the linear system. Indeed, the solution cost of GMRES for elliptic problems scales quadratically with the number of unknowns in the grid, unless effective preconditioning is used [6]. To combat this, physics–based preconditioning is applied [6, 7]. The right preconditioned system can be express as,

$$\mathcal{J}(\mathbf{x}^{(k)})\mathbf{M}^{-1}\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{x}^{(k)} + \varepsilon\mathbf{M}^{-1}\mathbf{v}) - \mathbf{F}(\mathbf{x}^{(k)})}{\varepsilon}, \tag{6}$$

where $\mathbf{M}$ symbolically represents the preconditioning matrix and $\mathbf{v} = \mathbf{M}\delta\mathbf{x}^{(k)}$. In GMRES $\mathbf{M}^{-1}$ only appears as a matrix–vector product with $\mathbf{v}$. Therefore, applying the preconditioner involves either approximately inverting $\mathbf{M}$ and applying it to $\mathbf{v}$ or solving the system,

$$\mathbf{M}\mathbf{q} = \mathbf{v} \tag{7}$$

As in the unpreconditioned system the explicit form of the Jacobian matrix is not required. Instead, one seeks to approximate $\mathbf{M}^{-1}\mathbf{v} \approx \mathcal{J}^{-1}\mathbf{v}$ through informed choices for the entries of $\mathbf{M}$ and selection of inversion operator. These choices form the basis of a physics–based preconditioner.

3. Structure

MOOSE capitalizes on the fact that JFNK only requires residual evaluations. This allows for a modular, pluggable architecture that greatly simplifies the addition of new physics and coupling them together. Further, nonlinear material properties and boundary conditions are all handled consistently.

As shown in Figure 1, MOOSE takes a layered approach to providing the core services in support of multiphysics simulation. The physics modules shown at the top of Figure 1 are called Kernels and can be added, removed, and coupled together easily. MOOSE then sits below this level, providing a set of core functionality necessary for residual and Jacobian evaluation. Underneath MOOSE is the libMesh finite-element framework developed by the CFDLab at the University of Texas in Austin [8]. libMesh provides a set of utilities for doing massively parallel finite-element based computations, including: mesh I/O, finite-element library, and interfaces to solver packages. Utilizing the interfaces in libMesh provides MOOSE, and subsequently the application built using MOOSE, a large amount of flexibility including the ability to swap out solver libraries and to utilize large scale parallel computing resources.

3.1. Kernels and Boundary Conditions

The heart of MOOSE is the Kernel. A Kernel is a piece of physics or a residual. To add new physics to an application built using MOOSE, a new Kernel must be supplied. Often, it's

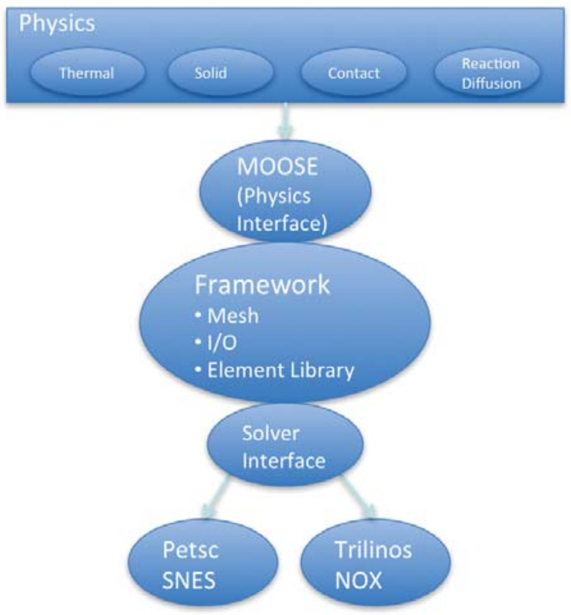


Figure 1. MOOSE Structure

convenient to think of a Kernel as a mathematical operator, such as a Laplacian or a convection term in a PDE. Kernels can be swapped or coupled together to achieve different application goals.

All Kernels are required to supply a residual. This residual usually involves summing products of finite-element shape functions (which MOOSE provides from libMesh). Alternatively, the residual could be calculated by calling a third party application or doing some calculations on the mesoscale and extrapolating to form the engineering scale residual.

Kernels can also optionally provide a Jacobian or preconditioning matrix, which can then be used for physics based preconditioning of the matrix free calculation. Often, the diagonal block is provided by a Kernel which is usually sufficient for effective preconditioning.

Boundary Conditions are treated similarly to Kernels, with residuals and Jacobians evaluated on boundaries. Boundary Conditions usually set a condition on either the value of a variable or it's gradient, but could also involve coupling two variables together or forming a Robin type boundary condition.

3.2. Materials

Material characterisation is extremely important in nuclear processes. Before asking the Kernels to compute their residuals, MOOSE first tells the Materials that are currently active to compute their values at all the necessary locations (coinciding with where Kernels are going to need material information). During this step a Material can call out to a third party library such as MATPRO to evaluate itself or it can perform it's own internal (possibly nonlinear) calculation.

Different `Materials` can be applied to different subsets of the domain. This plays an important role in the ability to do calculations on complex domains such as fuel rods with pellets and cladding. When calculating residuals in different sections of the domain the different active `Material` objects will be asked to compute their values.

3.3. Physics Based Preconditioner

As mentioned in Section 2 effective preconditioning of the Krylov iterative solver is necessary for optimal convergence. `MOOSE` includes an extremely flexible Physics Based Preconditioner (PBP) that can be utilized by any application built using `MOOSE`. The PBP allows for runtime configuration of all preconditioning options. Three main preconditioning schemes are usually employed by applications: operator split using only block diagonals, lower block triangular using one block Gauss-Seidel like iteration and variable block structure utilizing multiple block Gauss-Seidel iterations.

Traditionally operator splitting and linearization techniques are applied to form the preconditioning matrix $\mathbf{M}$ found in (6). In this scheme, only the diagonal blocks of $\mathbf{M}$ are approximated and each block row of (7) is solved for independently. Forming a fully decoupled $\mathbf{M}$ for a two variable (g,h) system amounts to,

$$\mathbf{F}(g,h) = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix}, \tag{8}$$

$$\mathcal{J} \approx \mathbf{M} = \begin{bmatrix} (\mathbf{F}_1)_g & 0 \\ 0 & (\mathbf{F}_2)_h \end{bmatrix}, \tag{9}$$

where $(\mathbf{F}_1)_g$ and $(\mathbf{F}_2)_h$ can be approximations and linearizations. Solving each of the block systems separately is compelling as it allows for the choice of solver which is tailored to the operators present in the block. In particular, if the diagonal block contains a large elliptic component then multigrid preconditioning can be very effective [9].

Extending this idea by supplying off-diagonal blocks such that $\mathbf{M}$ is lower block triangular; a set of augmented systems can be solved,

$$\mathbf{M} = \begin{bmatrix} (\mathbf{F}_1)_g & 0 \\ (\mathbf{F}_2)_g & (\mathbf{F}_2)_h \end{bmatrix}, \tag{10}$$

$$(\mathbf{F}_1)_g \mathbf{q}_1 = \mathbf{v}_1, \tag{11}$$

$$(\mathbf{F}_2)_h \mathbf{q}_2 = \mathbf{v}_2 - (\mathbf{F}_2)_g \mathbf{q}_1, \tag{12}$$

This formulation is well suited for preconditioning one-way coupled systems. Off-diagonal blocks in the lower block triangular region carry information about the coupling, allowing for a much more accurate preconditioner.

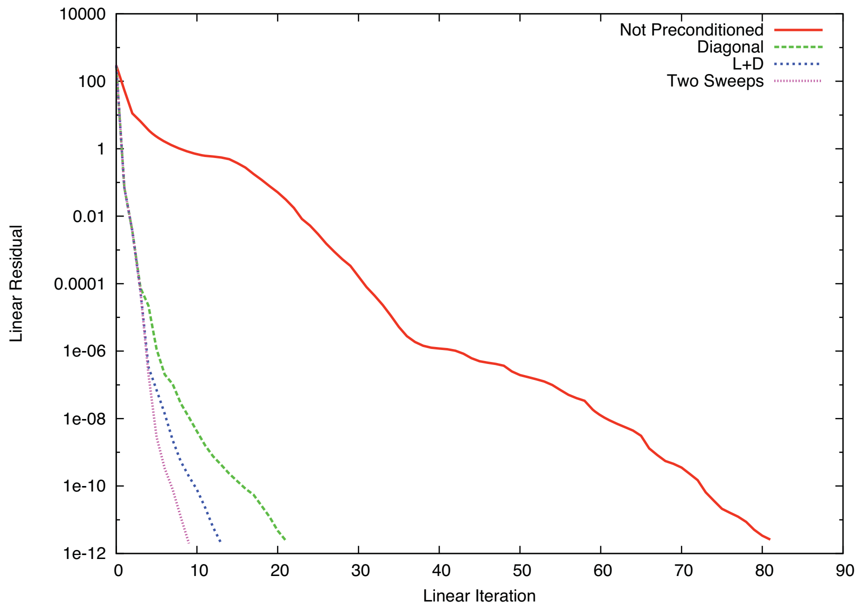


Figure 2. Convergence for Preconditioning Schemes. "Diagonal" utilizes only the diagonal blocks of M. "L+D" is lower block diagonal. "Two Sweeps" corresponds to two block Guass-Seidel iterations on (13)

A more complex option allows for a completely variable block structure. In this situation blocks lie both above and below the diagonal. In order to make use of the upper triangular information, multiple iterations of block Gauss-Seidel solves are performed [10]. Consider a linear thermo-mechanical system consisting of temperatre (T) and x,y,z displacements. In this case x,y and z are all coupled to each other as well as being one way coupled to temperature. An ideal preconditioning matrix would then look like:

$$\mathbf{M} = \begin{bmatrix} (\mathbf{F}_T)_t & 0 & 0 & 0 \\ (\mathbf{F}_x)_t & (\mathbf{F}_x)_x & (\mathbf{F}_x)_y & (\mathbf{F}_x)_z \\ (\mathbf{F}_y)_t & (\mathbf{F}_y)_x & (\mathbf{F}_y)_y & (\mathbf{F}_y)_z \\ (\mathbf{F}_z)_t & (\mathbf{F}_z)_x & (\mathbf{F}_z)_y & (\mathbf{F}_z)_z \end{bmatrix} \tag{13}$$

Solving this system would proceed as in the lower block diagonal example, first solving the temperature block row and using the result to solve the next row. In the first iteration the blocks above the diagonal will be ignored, while in subsequent iterations they will be used just as the lower diagonal blocks are. In this particular example the diagonal blocks are all elliptic dominated therefore a multigrid solver can be used to efficiently invert the systems. Figure 2 shows results obtained from the above three schemes, and the result of not preconditioning.

The PBP available in `MOOSE` allows full flexibility in specifying which blocks to compute and even what order to solve the block rows. The optimal order in which rows are solved is extremely important and not always clear. In the above thermo-mechanical system changing the solve order for the displacement rows can cause up to double the number of linear iterations. For these reasons, an extremely flexible PBP is desireable for preconditioning the linear systems arrising from a JFNK solution procedure being applied to a tightly coupled system of equations.

4. Advanced Capabilities

`MOOSE` provides applications with many advanced simulation capabilities including:

- Dimensionless physics: write the code once, and run problems in one, two or three dimensions.
- Massively parallel computation.
- Arbitrary order finite elements.
- Higher order time integration.
- Reading and writing many mesh / solution formats.
- *h* (cell subdivision), *p* (order elevation) and *r* (mesh redistribution) adaptation.
- Physics independent and physics dependent error estimation.

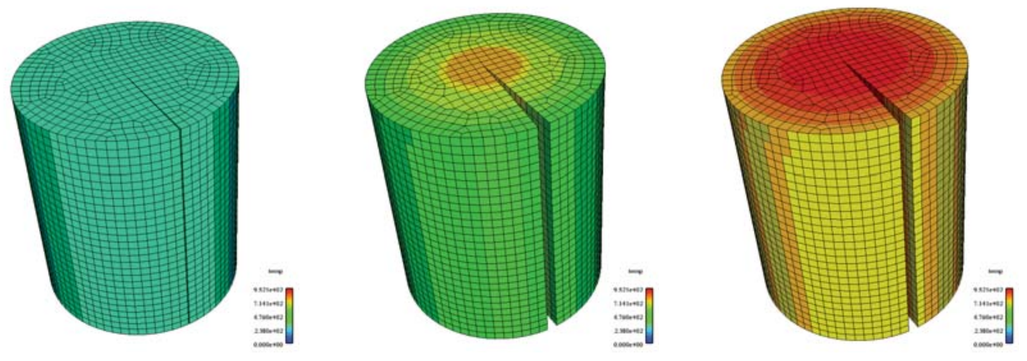
All of these capabilities work together seamlessly from the application developers perspective. The necessary input from the physics application developer is a residual in the form of a `Kernel`.

5. Applications

Several applications are currently in development based on the `MOOSE` framework. `BISON` is a leading edge reactor fuel performance modeling tool. It couples highly nonlinear mechanics, heat conduction and species diffusion equations to model fuel behavior in steady state, power-up and operational timeframes. After four months of development `BISON` can perform simulations in both 2 and 3D and on massively parallel computer architectures. This a direct result of it being based on `MOOSE`. Figure 3 shows simulation results calculated by `BISON` for a typical oxide fuel in a power-up scenario. Note the simulation is fully 3D and includes a predefined crack.

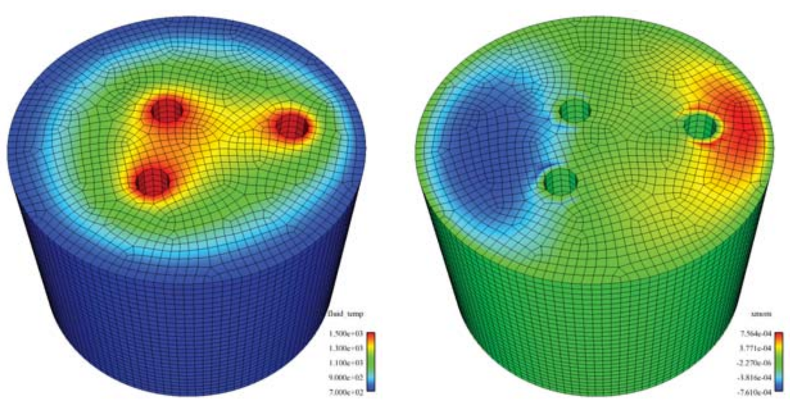
`PRONGHORN` is a pebble bed reactor simulation application. It can be run in 2D, 3D, and 2D cylindrical coordinates. It's current capabilities include heat conduction and transport using methods developed for porous media applications. After three months of development`PRONGHORN` is reproducing results from the SANA experiments utilizing heating rods in a bed of pebbles [11]. Figure 4 shows a typical 3D simulation calculated by `PRONGHORN`. In this simulation three different heating rods are spaced throughout the bed of pebbles with helium surrounding the pebbles.

Recently, a multi-group neutron diffusion simulation capability has been developed using `MOOSE`. The modular, pluggable architecture of `MOOSE` allowed this application to be built, results



(a) Initial **(b)** 1 min. **(c)** 2.5 min.

Figure 3. Transient Power-up With Crack



(a) Fluid Temperature **(b)** x Momentum

Figure 4. Pebble Bed With 3 Heating Elements

obtained and validated against the PBMR400 benchmark [12] with only one month of development time. This neutronics capability will be used by both `BISON` and `PRONGHORN`. Figure 5 shows a 3D, three group neutron diffusion result for a sodium cooled fast reactor. Figure 6 shows h-adaptivity being used to evolve the mesh to better capture a 2D neutronics solution.

6. Conclusion

JFNK, physics based preconditioning and a flexible pluggable code architecture are ushering in a new era of multiphysics computation. `MOOSE` combines these ideas into a modern, general, comprehensive framework enabling rapid prototyping as well as yielding production ready massively parallel codes in a fraction of the time previously thought necessary. Further, the advanced capabilities (*e.g.* error estimation and adaptivity) that `MOOSE` supplies enable high fidelity, efficient, predictive engineering simulation.

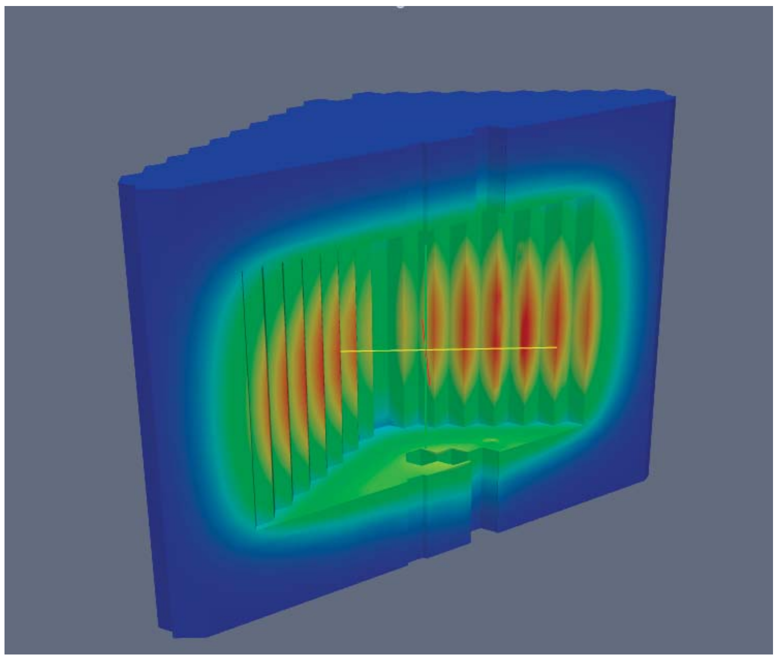
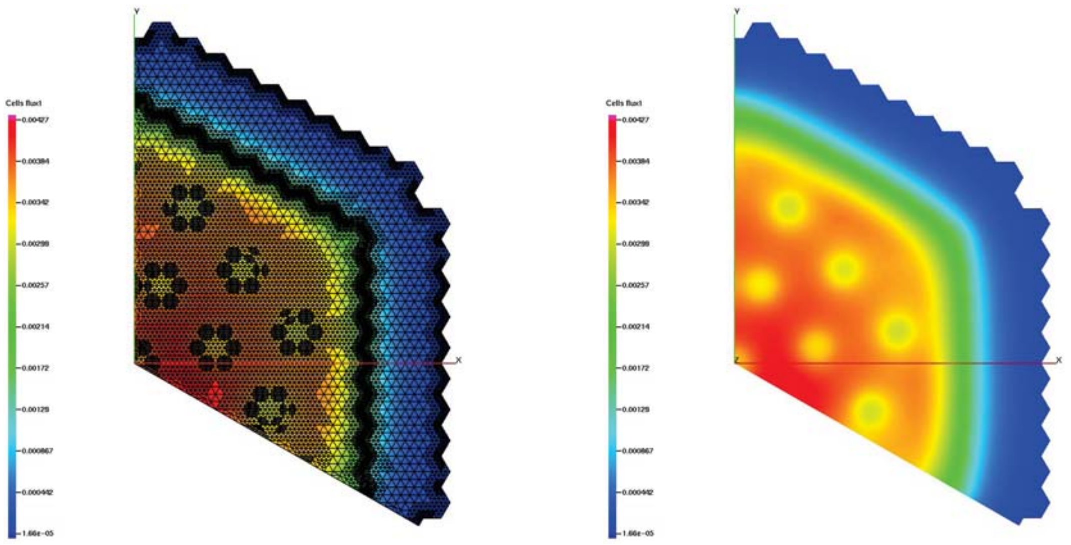


Figure 5. 3D Sodium Cooled Fast Reactor Neutron Diffion



(a) Adapted Mesh **(b)** Group 1 Flux

Figure 6. 2D Sodium Cooled Fast Reactor Neutron Diffusion With h-Adaptivity

Acknowledgments

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC07-05ID14517 (INL/JOU-07-13452). Accordingly, the U.S. Government

REFERENCES

[1] D. A. Knoll, L. Chacon, L. G. Margolin, and V. A. Mousseau, "On Balanced Approximations for Time Integration of Multiple Time Scales Systems," *J. Comput. Phys.*, **185**, pp.583–611 (2003).

[2] D. L. Ropp, J. N. Shadid, and C. C. Ober, "Studies of the accuracy of time integration methods for reaction-diffusion equations," *J. Comput. Phys.*, **194**, 2, pp.544–574 (2004).

[3] Y. Saad, *Iterative Methods for Sparse Linear Systems*, The PWS Series in Computer Science, PWS Publishing Company, Boston, MA (1995).

[4] D. A. Knoll and D. E. Keyes, "Jacobian-Free Newton-Krylov Methods: a Survey of Approaches and Applications," *J. Comput. Phys.*, **193**, 2, pp.357–397 (2004).

[5] M. Pernice and H. F. Walker, "NITSOL: a Newton iterative solver for nonlinear systems," *SIAM J. Sci. Comp.*, **19**, 1, pp.302–318 (1998).

[6] D. A. Knoll and W. J. Rider, "A Multigrid Preconditioned Newton-Krylov Method," *SIAM J. Sci. Comput.*, **21**, pp.691–710 (2000).

[7] L. Chacon, D. A. Knoll, and J. M. Finn, "An Implicit, Nonlinear Reduced Resistive MHD Solver," *J. Comput. Phys.*, **178**, pp.15–36 (2002).

[8] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, "libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations," *Eng Comput-Germany*, **22**, 3-4, pp.237–254 (Jan 2006).

[9] H. Park, R. Nourgaliev, V. Mousseau, and D. Knoll, "Physics-based Preconditioning of the rDG-JFNK method for All-Speed Fluid Flows with Heat Conduction and Viscosity," in "Tenth Copper Mountain Conference on Iterative Methods," (April 6–11 2008).

[10] V. A. Mousseau and D. A. Knoll, "New Physics-Based Preconditioning of Implicit Methods for Nonequilibrium Radiation Diffusion," *J. Comput. Phys.*, **190**, pp.42–51 (2003).

[11] H. Niessen and B. Stöcker, "Data sets of SANA experiment: 1994–1996," *JUEL-3409, Forschungszentrum Jülich* (1997).

[12] F. Reitsma, K. Ivanov, T. Downar, H. de Hass, S. Sen, G. Strydom, R. Mphahlele, B. Tyobeka, V. Seker, H. Gougar, and H. Lee, "PBMR Coupled Neutronics/Thermal Hydraulics Transient Benchmark The PBMR-400 Core Design," Tech. Rep. NEA/NSC/DOC(2007) Draft-V07, OECD/NEA/NSC (2007).