Parallel multiphysics algorithms and software for computational nuclear engineering

D Gaston, G Hansen, S Kadioglu, D A Knoll, C Newman, H Park, C Permann and W Taitano

Multiphysics Methods Group
Idaho National Laboratory, Idaho Falls, ID 83415-3840
E-mail: dana.knoll@inl.gov

Abstract. There is a growing trend in nuclear reactor simulation to consider multiphysics problems. This can be seen in reactor analysis where analysts are interested in coupled flow, heat transfer and neutronics, and in fuel performance simulation where analysts are interested in thermomechanics with contact coupled to species transport and chemistry. These more ambitious simulations usually require some level of parallel computing. Many of the coupling efforts to date utilize simple code coupling or first-order operator splitting, often referred to as loose coupling. While these approaches can produce answers, they usually leave questions of accuracy and stability unanswered. Additionally, the different physics often reside on separate grids which are coupled via simple interpolation, again leaving open questions of stability and accuracy. Utilizing state of the art mathematics and software development techniques we are deploying next generation tools for nuclear engineering applications. The Jacobian-free Newton-Krylov (JFNK) method combined with physics-based preconditioning provide the underlying mathematical structure for our tools. JFNK is understood to be a modern multiphysics algorithm, but we are also utilizing its unique properties as a scale bridging algorithm. To facilitate rapid development of multiphysics applications we have developed the Multiphysics Object-Oriented Simulation Environment (MOOSE). Examples from two MOOSE-based applications: PRONGHORN, our multiphysics gas cooled reactor simulation tool and BISON, our multiphysics, multiscale fuel performance simulation tool will be presented.

1. Introduction

The use of multiphysics simulation is on the increase in computational nuclear engineering. The increase in this activity is typically attributed to increasing computer power, but in truth, advanced numerical methods are playing an equal role. The term multiphysics simulation is used to describe simulations which include some level of disparate physical phenomena. Typical examples of multiphysics problems in nuclear engineering include coupling fluid flow, heat transfer and neutron kinetics for reactor dynamics, coupling fluid flow and structural dynamics to consider fluid-structure interactions for fuel rod fretting, and coupling nonlinear thermomechanics with contact, fission product behavior, and species transport to study fuel performance. In addition to multiphysics coupling, most of these problems also have multiscale issues to resolve.

In this endeavor the computational nuclear engineer is faced with choices of time integration and spatial discretization of the coupled system. Frequently, first-order operator splitting, or even explicit coupling of different codes, is used to perform the time integration. While this approach
can produce results, it can also produce significant time integration error [1] and stability issues [2].

In this paper we discuss a modern multiphysics algorithm, the Jacobian-Free Newton-Krylov method (JFNK) combined with physics based preconditioning [3]. This approach has many of the desirable features of operator splitting, but can significantly reduce the time integration error and stability issues. We describe the JFNK methodology and its use in both a multiphysics and multiscale setting. Next we will describe and evolving software framework (MOOSE[4]), which utilizes this algorithmic framework and enables rapid development of multiphysics engineering analysis tools. Finally, we describe two evolving multiphysics engineering analysis tools; one for high temperature gas cooled reactors (PRONGHORN[5]) and one for fuel performance simulation (BISON[6]).

2. Multiphysics methods
As a nuclear engineering model problem used for the purposes of algorithm discussion, consider thermal conduction coupled to a one group diffusion model of neutron kinetics, ignoring delayed neutrons,

\[
\rho C_p \frac{\partial T}{\partial t} - \nabla \cdot (K(T) \nabla T) = w \Sigma_f(T) \phi, \tag{1}
\]

\[
\frac{1}{v} \frac{\partial \phi}{\partial t} - \nabla \cdot (D(T) \nabla \phi) + \Sigma_a(T) \phi = \nu \Sigma_f(T) \phi. \tag{2}
\]

In this highly simplified homogeneous picture of a reactor, nuclear fission generates heat, and this heat is removed from the system by thermal conduction. Here \( \rho \) is material density, \( C_p \) is material heat capacity, \( T \) is material temperature, \( K \) is the material thermal conductivity, \( w \) is the energy released per fission event, \( \Sigma_f \) is the macroscopic fission cross-section, \( \Sigma_a \) is the macroscopic absorption cross-section \( \phi \) is the neutron scalar flux, \( v \) is the neutron velocity, \( D \) is the neutron diffusion coefficient, and \( \nu \) is the number of neutrons produced per fission event.

In this model \( D, K, \) and \( \Sigma_f \) all are functions of \( T \). This simple model problem is a nonlinear PDE system with multiple time and space scales. We can use this problem to help highlight the difference in tightly coupled multiphysics solution algorithms and loosely coupled multiphysics solution algorithms. Without loss of generality we ignore spatial discretization issues.

It is our opinion that modern multiphysics simulations should be second-order accurate in time, or higher. However, for simplicity of presentation we will discuss some issues within the context of a first-order time integration approach. A fully implicit first-order in time discretization of this problem is

\[
\rho C_p \frac{T^{n+1} - T^n}{\Delta t} - \nabla \cdot (K(T^{n+1}) \nabla T^{n+1}) = w \Sigma_f(T^{n+1}) \phi^{n+1}, \tag{3}
\]

\[
\frac{1}{v} \frac{\phi^{n+1} - \phi^n}{\Delta t} - \nabla \cdot (D(T^{n+1}) \nabla \phi^{n+1}) + \Sigma_a(T^{n+1}) \phi^{n+1} = \nu \Sigma_f(T^{n+1}) \phi^{n+1}. \tag{4}
\]

This specific time discretization will require nonlinear iteration within a time step.

The actual process of first-order splitting, or coupling two separate codes together, does not solve the above time discretization requiring nonlinear iteration. Typically \( D, K, \Sigma_f \) and \( \Sigma_a \) are evaluated from the previous time step. When coupling existing codes to execute a time step, sources are evaluated with an explicit dependence of variables from the other code, to simplify data exchange between codes. Such a time discretization would look like

\[
\rho C_p \frac{T^{n+1} - T^n}{\Delta t} - \nabla \cdot (K(T^n) \nabla T^{n+1}) = w \Sigma_f(T^n) \phi^n, \tag{5}
\]
An analysis can be performed of this time discretization to further understand the time integration error \[1\]. Frequently this error can be significant, especially if many time steps are considered. Additionally, this type of splitting and linearization can produce new time step constraints for nonlinear stability. Iteration on sources and temperature dependent coefficients within a time step can help minimize these errors and stability issues.

2.1. Jacobian-free Newton-Krylov methods

Newton’s method requires the solution of the linear system

\[ J^k \delta u^k = -F(u^k), \]  

(7)

followed by the update

\[ u^{k+1} = u^k + \delta u^k, \]  

(8)

where \( J \) is the Jacobian matrix, \( F(u) \) is the nonlinear system of equations, \( u \) is the state vector, and \( k \) is the nonlinear iteration index. In vector notation, the \((i, j)^{th}\) element of the Jacobian matrix is,

\[ J_{i,j} = \frac{\partial F_i(u)}{\partial u_j}. \]  

(9)

Forming each element of \( J \) requires taking analytic or numerical derivatives of \( F_i(u) \) with respect to \( u \) at each grid point. This can be difficult, time consuming, and error-prone.

In JFNK, a Krylov method \[7\] is used to solve Eq. (7). An initial linear residual, \( r_0 \), given an initial guess, \( \delta u_0 \), is evaluated as

\[ r_0 = -F(u) - J\delta u_0. \]  

(10)

Note that the nonlinear iteration index, \( k \), has been dropped. This is because the Krylov iteration is performed at a fixed \( k \). The approximate solution at the \( l\)-th Krylov iteration, \( \delta u_l \), is constructed from a linear combination of the Krylov vectors (search directions) \( \{r_0, Jr_0, (J)^2r_0, ..., (J)^{l-1}r_0\} \), which were constructed during the previous \( l-1 \) Krylov iterations. This linear combination of Krylov vectors can be written as,

\[ \delta u_l = \delta u_0 + \sum_{j=0}^{l-1} \alpha_j (J)^j r_0, \]  

(11)

where evaluating the scalars \( \alpha_j \) is part of the Krylov iteration. Upon examining Eq. (11) we see that a Krylov method requires the action of the Jacobian only in the form of matrix-vector products, which can be approximated by \[8\];

\[ Jv \approx [F(u + \epsilon v) - F(u)] / \epsilon, \]  

(12)

where \( v \) is a Krylov vector (i.e. one of \( \{r_0, Jr_0, (J)^2r_0, ..., (J)^{l-1}r_0\} \)), and \( \epsilon \) is a small perturbation. This matrix-free approach, besides its obvious memory advantage, has many unique capabilities. Namely, Newton-like nonlinear convergence without forming the true Jacobian. A fair amount of literature exists on JFNK motivated by ODEs and boundary value problems, while less exists for time-dependent PDEs. Standard “PDE motivated” references are \[8, 9, 3\]. It is also possible to form \( Jv \) analytically without forming \( J \). This approach has a number of positive features to consider, but we are not employing it in our work yet.
The JFNK method also has its advantages and disadvantages. The main advantage is that for a tight convergence tolerance there is no splitting or linearization error [1]. The algorithm provides a clean way to include other nonlinear phenomena. Another advantage is that one can implement a variety of time discretizations such as higher-order backward difference formula (BDF) methods or implicit Runge-Kutta methods.

The main disadvantage of the JFNK method is that one is solving a nonlinear problem, with iteration, to advance a time step. Furthermore, one is solving the full dimensional system implicitly. Thus a large matrix system must be approximately inverted on each Newton iteration. In order for this approach to be tractable one must produce an effective preconditioner.

2.2. Preconditioning of JFNK

The purpose of preconditioning a Krylov method is to efficiently cluster eigenvalues of the iteration matrix, which in turn will reduce the required number of Krylov iterations. Traditionally, standard iterative methods such as Jacobi, SSOR, or ILU are applied to the system matrix when constructing a preconditioner [7]. We refer to this as algebraic preconditioning methods. When applying physics-based preconditioning to the JFNK method the system matrix, $J$, is never formed.

When employing right preconditioning to solve the Newton step this is typically expressed as,

$$(JM^{-1})(Mδu) = -F(u).$$

(13)

$M^{-1}$ is a linear operator which symbolically represents the preconditioning process, and $M$, the inverse of the preconditioning process, is functionally not required. In the idea of physics-based preconditioning (to be discussed), $M^{-1}$ represents a linearized time step. The right preconditioned version of the JFNK matrix-vector multiply approximation is:

$$JM^{-1}v \approx [F(u + εM^{-1}v) - F(u)] / ε.$$  

(14)

This process is actually done in two steps;

(i) Perform $y = M^{-1}v$

(ii) Perform $Jy \approx [F(u + εy) - F(u)] / ε.$

2.3. Physics-based preconditioning

There exist numerous legacy algorithms to solve nonlinear systems, and these algorithms are typically based on linearization and time splitting. This linearization and time splitting has usually been developed with significant insight into the time scales or physical behavior of the problem. As a benefit of this insight, a reduced implicit system, or a sequence of segregated implicit systems may be solved in place of the fully coupled system. In the end, these legacy algorithms may make excellent preconditioners for the Jacobian-free Newton-Krylov method.

This algorithmic concept is referred to as physics-based preconditioning. Here, the Jacobian matrix is not directly manipulated algebraically to find an approximate inverse; rather, approximations are made to the original differential system that result in a modified, more tractable, approximate Jacobian system for preconditioning.

Consider using the first-order, explicit coupling, operator split method for our model problem as a preconditioner. The Newton step, $Jδu = -F(u)$, of Eqs. (3) and (4) can be expressed as:

$$\begin{bmatrix}
  J_{φ,T} & J_{φ,φ} \\
  J_{T,φ} & J_{T,φ}
\end{bmatrix}
\begin{bmatrix}
  δφ \\
  δT
\end{bmatrix}
= -\begin{bmatrix}
  F_φ \\
  F_T
\end{bmatrix},$$

4
with
\[ \delta \mathbf{u} = [\delta \phi, \delta T]^T, \quad \mathbf{F}(\mathbf{u}) = [F_\phi, F_T]^T. \]

Typically in problems such as this, simplification is made to the operators \( J_{T,T} \) and \( J_{\phi,\phi} \) within the preconditioner. As an example, one can linearize the temperature dependent coefficients, \( K(T), D(T) \), as \( K(T^n), D(T^n) \), simplifying the evaluation of \( J_{T,T} \) and \( J_{\phi,\phi} \). After enough simplification it may be more appropriate to call the preconditioning operators \( M_{T,T} \) and \( M_{\phi,\phi} \). A few v-cycles of a multigrid method are used to smooth the two separate elliptic systems in the preconditioner. This process is equivalent to using code coupling (i.e., the time discretization of Eqs. (5) and (6) as the preconditioner). Thus only simple matrices are formed for preconditioning purposes. One could add some source coupling while retaining a block lower triangular system such that:
\[
\mathbf{M} = \begin{bmatrix}
    M_{\phi,\phi} & 0 \\
    M_{T,\phi} & M_{T,T}
\end{bmatrix}.
\]

Then a preconditioning process \( \delta \mathbf{u} \approx \mathbf{M}^{-1}(-\mathbf{F}(\mathbf{u})) \) is done in two steps

(i) Approximate inversion of a parabolic system of order \( N \) (number of grid points) to solve for \( \delta \phi \).

\[
M_{\phi,\phi} \delta \phi = -F_\phi \tag{15}
\]

(ii) Approximate inversion of a parabolic system of order \( N \) (number of grid points) to solve for \( \delta T \).

\[
M_{T,T} \delta T = -F_T - M_{T,\phi} \delta \phi \tag{16}
\]

Other forms of physics-based preconditioning are discussed in [3].

2.4. Advanced uses of JFNK

In addition to the standard multiphysics solver application of JFNK we have started using the method to improve the convergence rates of other computational nuclear engineering problems [10]. We have successfully used the JFNK method to accelerate the convergence of the classic power iteration used to evaluate the eigenvalue in criticality searches. Here we use the standard power iteration as a preconditioner. The constraint equation for the fundamental mode is added as an additional nonlinear function, and the eigenvalue is added as an unknown to the system.

We have also started using JFNK as a multilevel solver. In reactor physics, a low-order diffusion-like solver on a coarse mesh is often used to accelerate the convergence of a higher-order transport method on a fine mesh. Both models can produce a version of the scalar flux. These two models are typically coupled via a fixed point iteration. We have employed JFNK where the lower order diffusion-like problem is used as the nonlinear function on the coarse mesh. Then, inside this function evaluation we execute a self-consistent transport sweep on a fine mesh. In this context the JFNK method has become a nonlinearly consistent scale-bridging algorithm. We are also using this algorithmic concept within BISON to do multiscale thermal transport in nuclear fuel.

3. Software and results

3.1. MOOSE

Motivated by the existence of strong multiphysics coupling in many computational nuclear engineering processes, we are developing the Multiphysics Object Oriented Simulation Environment, or MOOSE [4]. Algorithmically MOOSE is based on solving all systems in a fully coupled manner using JFNK and physics-based preconditioning. MOOSE is designed to support the development of engineering analysis applications. Unlike a “research code,” the architecture
and development methodology of MOOSE must focus on more than sheer performance. MOOSE was developed with the following in mind:

- Massively parallel performance and scalability is needed for large problems, but economics mandate that MOOSE applications also perform well on desktop computers and modestly parallel machines.
- MOOSE must be developed using modern software engineering principles, as its value as a framework is directly related to the cost of maintaining and extending applications based on it.
- MOOSE is a component of an engineering analysis system that includes domain geometry construction, mesh generation, mesh quality and adaptation, and visualization and analysis of the final results. As such, the usability of MOOSE applications depends on the efficacy of the entire simulation system; including the robustness of system-wide uncertainty quantification and the ability to assess the propagation of uncertainty throughout.

The design of MOOSE capitalizes on the fact that the (unpreconditioned) JFNK implementation requires only residual evaluations of the discrete system. Strategic use of this results in a modular, pluggable architecture that greatly simplifies adding and coupling new physics together. Further, nonlinear material properties and boundary conditions are handled consistently. MOOSE is based on a layered approach to providing the core services in support of multiphysics simulation. Underneath MOOSE is the libMesh finite-element framework developed by the CFDLab at the University of Texas in Austin [11]. libMesh provides a set of utilities for massively parallel finite-element based computations, including mesh I/O, a finite-element library, and interfaces to solver packages. Utilizing the interfaces in libMesh provides MOOSE, and subsequently the applications built using MOOSE, a considerable amount of flexibility including the ability to swap out solver libraries such as PETSc [12] and Trilinos [13], and to utilize diverse large scale parallel computing resources.

3.2. BISON

BISON [6], the fuel performance code developed using MOOSE, is designed for fully coupled steady and transient analysis and to be efficient on both desktop computers and in massively parallel environments. This discussion summarizes the current status of the BISON fuel performance analysis code and demonstrates results on selected three dimensional capabilities that support fuel performance calculations.

While the equation sets in BISON could be applied to a variety of fuel types, here we will focus on the application to traditional light water reactor fuel. The problem consists of three fully coupled partial differential equations for heat conduction, oxygen nonstoichiometry and linear elastic solid mechanics. Let \( \Omega \in \mathbb{R}^d \), \( d = 2, 3 \) define the computational domain (the fuel pellet).

The heat conduction model assumes fission reactions generate heat at a uniformly distributed constant rate, \( Q \),

\[
\rho C_p \frac{\partial T}{\partial t} + \nabla \cdot \mathbf{q} - Q = 0, \tag{17}
\]

where \( T, \rho \) and \( C_p \) are temperature, density and specific heat of UO\(_2\), respectively. The heat flux \( \mathbf{q} \) within the UO\(_2\) domain \( \Omega \) may be written as

\[
\mathbf{q} = -k \nabla T, \tag{18}
\]
where $k$ denotes thermal conductivity, A hyperstoichiometric model for oxygen is included as described in [6]. Let $J$ denote the oxygen flux in the hyperstoichiometric regime with,

$$J = -D \left( \nabla x - \frac{xQ^*}{FRT^2} \nabla T \right),$$  \hspace{1cm} (19)

where $D$ is diffusivity of UO$_2$, $F$ is the thermodynamic factor of oxygen, $Q^*$ is the heat of transport of oxygen, and $R$ is the universal gas constant. The nonstoichiometric model for oxygen diffusion, $x$, is given by

$$\frac{\partial x}{\partial t} + \nabla \cdot J = 0.$$

$$\frac{\partial^2 u}{\partial t^2} = A^T D A u + f = 0$$  \hspace{1cm} (21)

with

$$A = \begin{bmatrix} \partial_x & 0 & 0 \\ 0 & \partial_y & 0 \\ 0 & 0 & \partial_z \end{bmatrix} \quad \text{and} \quad D = c_1 \begin{bmatrix} 1 & c_2 & c_2 & 0 & 0 & 0 \\ c_2 & 1 & c_2 & 0 & 0 & 0 \\ c_2 & c_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_3 \end{bmatrix},$$

and

$$c_1 = \frac{E(1-\nu)}{(1+\nu)(1-\nu)}, \quad c_2 = \frac{\nu}{(1-\nu)}, \quad c_3 = \frac{(1-2\nu)}{2(1-\nu)}.$$

The coefficients $E$ and $\nu$ are Young’s modulus and Poisson’s ratio. The forcing term $f$ weakly enforces thermal expansion.

Figure 1 displays two items. The first is a convergence plot for GMRES iteration on a specific Newton iteration for a time dependent problem of thermal expansion of a fuel pellet. One can see that the simplest of physics-based preconditioning options can provide significant acceleration of GMRES in BISON. The second item is a graphic of a pre-cracked fuel pellet which has undergone thermal expansion.

Figure 2 displays parallel scaling plot from BISON on a multiscale problem [14]. Here, thermal conduction is being solved on the engineering scale and a mesoscale simulation is being performed inside the nonlinear function evaluation to self-consistently evaluate a lower-length scale informed thermal conductivity.

### 3.3. PRONGHORN

Very-high Temperature Gas-cooled Reactors (VHTR) are intrinsically different from conventional Light Water Reactors (LWR). For example, a large temperature difference between inlet and outlet creates a significant density variation in the coolant. In this case, an incompressible assumption with Boussinesq approximations for buoyancy may not give accurate heat transfer. The strong temperature gradients introduce strong nonlinear couplings between heat-conduction and neutronics calculations. The traditional approach to modeling these complex multiphysics phenomena via operator-splitting or code coupling may introduce significant numerical and/or modeling errors.

**PRONGHORN** [5] is being developed as a tightly coupled multiphysics 3-D code which enables simulations of complex transient phenomena that are encountered in VHTRs for both
operational and accident scenarios. VHTRs are expected to have long transients (\sim days). To model this, eliminating time step restrictions due to explicit discretization and operator-split errors by employing fully implicit discretizations is the only viable route. PRONGHORN is a two and three dimensional (with two dimensional cylindrical coordinate) pebble bed reactor simulation application. Current capabilities include heat conduction and transport using methods developed for porous media applications. The reactor fluid medium model is described by

\begin{align}
\frac{\partial \rho_f}{\partial t} + \nabla \cdot (\rho_f \vec{u}) &= 0, \\
\frac{\partial \rho_f \vec{u}}{\partial t} + \rho_f \vec{u} \cdot \nabla P - \rho_f g &= W \rho_f \vec{u} = 0, \\
\frac{\partial}{\partial t} \left[ \rho_f c_{pf} T_f \right] + \nabla \cdot (\rho_f c_{pf} \vec{u} T_f) - \nabla \cdot \kappa_f \nabla T_f + \alpha (T_f - T_s) &= 0,
\end{align}

where \( \epsilon \) is porosity, \( \rho \) is density, \( \rho_f \vec{u} \) is momentum, \( P \) is pressure, \( W \) is fluid resistivity coefficient, \( c_p \) is heat capacity, \( T \) is temperature, \( \kappa \) is thermal conductivity, \( \alpha \) is the heat transfer coefficient and the subscripts \( f \) and \( s \) denote fluid and solid properties. The reactor solid medium is
governed by
\[
\frac{\partial}{\partial t} [(1 - \epsilon) \rho_s c_p s T_s] - \nabla \cdot \kappa_{s \text{eff}} \nabla T_s + \alpha (T_s - T_f) - Q = 0, \tag{25}
\]
where \(\kappa_{s \text{eff}}\) is the effective thermal conductivity of the solid and \(Q\) is the heat source. The reactor neutronics is given by
\[
\frac{1}{v_g} \frac{\partial \phi_g}{\partial t} - \nabla \cdot D_g \nabla \phi_g + \Sigma_R g \phi_g - (1 - \beta) \chi_g \sum_{g' = 1} \nu \Sigma_{fg'} \phi_{g'} - \sum_{g' = 1, g' \neq g} \Sigma_{g' \rightarrow g} \phi_{g'} - \sum_k \lambda_k C_k = 0, \tag{26}
\]
\[
\frac{dC_k}{dt} + \lambda_k C_k - \beta_k \sum_{g'} \nu \Sigma_{fg'} \phi_{g'} = 0, \tag{27}
\]
where \(v\) is neutron speed, \(\phi\) is neutron flux, \(D\) is diffusion coefficient, \(\Sigma_R\) is removal cross-section, \(\beta\) is delayed neutron fraction, \(\chi\) is fission spectrum, \(\nu\) is average number of neutrons emitted per fission reaction, \(\Sigma_f\) is the fission cross-sections, \(\Sigma_s\) is the scattering cross-sections, \(\lambda\) is the decay constant, \(C\) is precursor concentrations. The subscripts \(g\) and \(k\) represent energy group indices for prompt and delayed neutrons, respectively.

Equations (22)–(27) represent the continuity, momentum, fluid and solid energy, and neutron diffusion equations, respectively. The coefficients (e.g. thermal conductivities, heat transfer coefficients, and neutron cross sections) are typically strong functions of the independent variables, which introduces significant nonlinearity. A closure relation is obtained by the equation of state,
\[
P = \rho_f RT_f. \tag{28}
\]

Figure 3 displays two items. The first is a materials geometry for a 2-D benchmark PBMR 400 test case [5]. The second is a steady state multiphysics results which shows the variation of the eigenvalue, \(k_{\text{eff}}\) with prescribed steady state power. As the desired steady-state power level is varied, the reactor temperature profile will vary. Since the neutronics cross-section are a function of temperature, the eigenvalue will change. The plot was made using a standard accelerated power iteration but in a multiphysics setting.

Figure 4 displays two items from a multiphysics transient. A power profile and a zoom of this profile using a variety of time integration approaches. The power transient is activated by removing a control rod. We can see that the second-order in time tightly coupled solver produces the most accurate answer.

Figure 3. Steady-State Multiphysics PRONGHORN Result
4. Conclusions
We have discussed the algorithmic issues related to accurate multiphysics time integration on problems in computational nuclear engineering. We have described the modern algorithmic approach of JFNK and physics-based preconditioning. We have presented the results of our software framework development activities that have occurred over the last year and we have presented results from two recently developed engineering application codes.

5. References
[1] D. A. Knoll, L. Chacón, L. G. Margolin, and V. A. Mousseau. On balanced approximations for the time integration of multiple time scale systems. J. Comput. Phys., 185:583–611, 2003.
[2] D.L. Ropp and J. N. Shadid. Stability of operator splitting methods for systems with indefinite operators: reaction-diffusion systems. J. Comput. Phys., 203:449–466, 2005.
[3] D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. J. Comput. Phys., 193:357–397, 2003.
[4] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandié. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. Nuclear Engineering and Design, 2009. in press.
[5] H. Park, D. Gaston, S. Kadiouglu, D. Knoll, D. Lebrun-Grandie, R. Martineau, and W. Taitano. Tightly coupled multiphysics simulation for pebble bed reactors. In American Nuclear Society 2009 International Conference on Advances in Mathematics, Computational Methods, and Reactor Physics, Saratoga Springs, NY, May 3–7 2009.
[6] C. Newman, G. Hansen, and D. Gaston. Three dimensional coupled simulation of thermomechanics, heat, and oxygen diffusion in UO$_2$ nuclear fuel rods. Journal of Nuclear Materials, 2009. In press, http://dx.doi.org/10.1016/j.jnucmat.2009.03.035.
[7] Y. Saad. Iterative Methods for Sparse Linear Systems. PWS Publishing Company, Boston, 1996.
[8] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. SIAM J. Sci. Stat. Comput., 11:450–481, 1990.
[9] Tony F. Chan and Kenneth R. Jackson. Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms. SIAM J. Sci. Stat. Comput., 5:533–542, 1984.
[10] D.A. Knoll, H. Park, and K. Smith. Application of the jacobian-free newton-krylov method in computational reactor physics. In American Nuclear Society 2009 International Conference on Advances in Mathematics, Computational Methods, and Reactor Physics, Saratoga Springs, NY, May 3–7 2009.
[11] Benjamin S. Kirk, John W. Peterson, Roy H. Stogner, and Graham F. Carey. libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. Eng Comput-Germany.
[12] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois C. McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.
[13] M. Heroux et al. Trilinos: an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. http://trilinos.sandia.gov, 2008.
[14] M.R. Tonks, G. Hansen, D. Gaston, C. Permam, P. Millett, and D. Wolf. Fully coupled engineering and mesoscale simulations of thermal conductivity in $\text{UO}_2$ fuel using an implicit multiscale approach. Journal of Physics: Conference Series, 2009, this conference.