# FINAL REPORT

| | |
|---|---|
| **Project Title:** | **Neutronic / thermalhydraulic coupling techniques for sodium cooled fast reactor simulations** |
| **Covering Period:** | June 1, 2007 through May 31, 2010 |
| **Date of Report:** | Date of Report: September 28, 2010 |
| **Recipient:** | Texas Engineering Experiment Station (TEES)<br>Office of Sponsored Research<br>3000 TAMU<br>College Station TX, 77843-3000 |
| **Award Number:** | DE-FC07-07ID14837 |
| **Subcontractors:** | None |
| **Other Partners:** | None |

**Project Team:**

Principal Investigator: Jean Ragusa
Department of Nuclear Engineering
Texas A&M University
3133 TAMU
College Station, TX, 77843
Phone: (979) 862 2033
FAX: (979) 845 6443
E-mail: ragusa@ne.tamu.edu

Co-Principal Investigator: Andrew Siegel
Department of Computer Science
University of Chicago
1100 East 58th Street, Chicago, IL 60637
Phone: 773.702.6614
FAX: 773.702.8487
E-mail: asiegel@cs.uchicago.edu

Co-Principal Investigator: Jean-Michel Ruggieri
Commissariat a l'Energie Atomique (CEA)
Centre de Cadarache, Laboratoire d'Etudes de Physique,
13108, St Paul-lez-Durance, FRANCE
Phone: +442257052
FAX: +442257009
E-mail: jean-michel.ruggieri@cea.fr

## Executive Summary

The objective of this project was to test new coupling algorithms and enable efficient and scalable multi-physics simulations of advanced nuclear reactors, with considerations regarding the implementation of such algorithms in massively parallel environments. Numerical tests were carried out to verify the proposed approach and the examples included some reactor transients. The project was directly related to the Sodium Fast Reactor program element of the Generation IV Nuclear Energy Systems Initiative and the Advanced Fuel cycle Initiative, and, supported the requirement of high-fidelity simulation as a mean of achieving the goals of the presidential Global Nuclear Energy Partnership (GNEP) vision.

For decades, the modeling of nuclear cores has been divided into several distinct domains of physics: neutronics, hydraulics, heat transfer, … Yet, these physical models describe physical processes that are tightly intertwined and rely heavily on the solution field of one another. In the last decade or so, various existing mono-disciplinary codes have been coupled together in a naive "black-box" fashion, where the output of one code serves as the input of another code, producing nonlinearly inconsistent multiphysics solutions. Such schemes, which are still the main coupling paradigm today for solving nonlinear nuclear reactor physics equations, are based on a linearization of the coupling terms that is never resolved and that can lead to a loss of accuracy and stability in the solution procedure. In order to address the inconsistencies of traditional coupling strategies, a 3-D test-bed code based on reduced physical models has been developed. It includes several physic components, namely, multigroup neutron diffusion, monophasic fluid conservation laws (mass, momentum, energy), nonlinear heat conduction. This test-bed code was used to demonstrate the loss of accuracy order due to traditional operator-split techniques used in conventional coupling schemes; to implement Jacobian-free full resolution coupling schemes; to develop adaptive preconditioning techniques; and to investigate high-order time discretizations and adaptive time stepping strategies. The test-bed code employs state-of-the-art algorithms and libraries, namely the Jacobian0free Newton-Krylov technique to solve consistently fully implicit tightly coupled simulations and the ANL's PETSc library of linear and nonlinear solvers for scalability. We have demonstrated that the multiphysics solution procedure yields highly accurate solutions in space and time and is amenable for space/time adaptivity. This portion of the work is discussed in Part A of this report.

It is widely agreed that high fidelity simulations is an utmost important tool for the conception, design and validation of complex physical systems such as nuclear reactors. Examples of such thrust can be seen in the ASCI program and the SciDAC program. GNEP strongly advocates the development of the next-generation simulation software with a far wider range of applicability than conventional tools. This will allow significant reductions in costs by diminishing (1) the cycle time for the development of new reactor designs and (2) the number of experimental verifications and the time needed to perform them. For instance, the CEA and ANL (among others), have initiated large efforts in developing new state-of-the-art neutronic [DESCARTES, UNIC] and thermal-hydraulic [NEPTUNE, TRIO, NEK] codes based on first principles. For instance, the UNIC and the NEK codes under development at ANL will be used to model with high accuracy prototypical fast reactors. The best performing techniques are proposed for implementation in the new neutronics / thermal-hydraulics code package, SHARP, for advanced fast reactor analyses. SHARP (Simulation-based High-efficiency Advanced Reactor Prototyping) is based on the UNIC code (neutronics) and Nek-5000 code

(computational fluid dynamics) developed at ANL. Part B of this report describes the recent advances in parallel computing applied to reactor simulations. The UNIC code system was developed and successfully performed with 294.912 processors on the XT-5 machine.

Our recommendation for subsequent work is as follows: to fully model transients for SFR, a 3D coupled thermo-mechanical– heat conduction model to assess the structural mechanic feedback effects from core radial expansion is needed. However, the development and verification of such a complex model that would need to account for metal fuel and cladding expansion, assembly and core plate expansion, and possibly assembly bowing and flowering was outside of the scope of the project and should be considered in future work.

One of the students supported by this project is now a computational post-doctoral fellow at the Mathematics and Computer Science division of ANL and works in close collaboration with Andrew Siegel, co-PI of this project.

The table below summarizes the milestones status.

| Task # | Task / Milestone Description | Planned Completion | Actual Completion |
|---|---|---|---|
| 1 | Bibliography, literature review | 9/1/07 | Completed |
| 2 | Development of a 2D neutronic, sodium thermalhydraulic code package | 12/1/07 | Completed |
| 3 | Stability and accuracy study of conventional coupling techniques | 3/1/08 | Completed |
| 4 | ABR benchmark using conventional techniques | 6/1/09 | Partial Completion |
| 5 | Development of nonlinearly consistent methods based on Newton's method | 12/1/08 | Completed |
| 6 | Development and testing of Jacobian-free techniques | 3/1/09 | Completed |
| 7 | Development of preconditioned Jacobian-free methods | 6/1/09 | Completed |
| 8 | ABR benchmark using nonlinearly consistent techniques | 6/1/09 | Partial Completion |
| 9 | Development of nonlinearly consistent methods based on Rosenbrock's method | 9/1/09 | Completed |
| 10 | Development and testing of Jacobian-free techniques | 9/1/09 | Completed |
| 11 | Time step control | 3/1/10 | Completed |
| 12 | ABR benchmark using Rosenbrock techniques | 3/1/10 | Partial Completion |
| 13 | archival journal articles write-up | 6/1/10 | Completed |
| 14.1 | Reporting to DOE | 6/1/08 | Completed |
| 14.1 | Reporting to DOE | 6/1/09 | Completed |
| 14.1 | Reporting to DOE | 6/1/10 | Completed |

# Part I

# Texas A&M University contribution

# Abstract

The modeling of nuclear reactors involves the solution of a multi-physics problem with widely varying time and length scales. This translates mathematically to solving a system of coupled, non-linear, and stiff partial differential equations (PDEs). Multi-physics applications possess the added complexity that most of the solution fields participate in various physics components, potentially yielding spatial and/or temporal coupling errors. This dissertation deals with the verification aspects associated with such a multi-physics code, i.e., the substantiation that the mathematical description of the multi-physics equations are solved correctly (both in time and space).

Conventional paradigms used in reactor analysis problems employed to couple various physics components are often non-iterative and can be inconsistent in their treatment of the non-linear terms. This leads to the usage of smaller time steps to maintain stability and accuracy requirements, thereby increasing the overall computational time for simulation. The inconsistencies of these weakly coupled solution methods can be overcome using tighter coupling strategies and yield a better approximation to the coupled non-linear operator, by resolving the dominant spatial and temporal scales involved in the multi-physics simulation.

A multi-physics framework, KARMA (K(c)ode for Analysis of Reactor and other Multi-physics Applications), is presented. KARMA uses tight coupling strategies for various physical models based on a Matrix-free Nonlinear-Krylov (MFNK) framework in order to attain high-order spatio-temporal accuracy for all solution fields in amenable wall clock times, for various test problems. The framework also utilizes traditional loosely coupled methods as lower-order solvers, which serve as efficient preconditioners for the tightly coupled solution. Since the software platform employs both lower and higher-order coupling strategies, it can easily be used to test and evaluate different coupling strategies and numerical methods and to compare their efficiency for problems of interest.

Multi-physics code verification efforts pertaining to reactor applications are described and associated numerical results obtained using the developed multi-physics framework are provided. The versatility of numerical methods used here for coupled problems and feasibility of general non-linear solvers with appropriate physics-based preconditioners in the KARMA framework offer significantly efficient techniques to solve multi-physics problems in reactor analysis.

# Chapter 1

# Introduction

'All models are wrong, but some are useful.'

– George Box

High fidelity computer simulations of coupled multi-physics problems require solving large systems of non-linear, stiff, coupled equations. Many examples of non-linearly coupled multi-physics phenomena exist in various scientifical fields, raising a need to develop stable and accurate numerical solution procedures. Some examples are:

1. Radiation diffusion where the radiation energy is strongly coupled to the material temperature field [1], [2].

2. Nuclear reactor analysis where the thermal power generated due to fission reactions in the fuel pin is strongly coupled with the thermal-hydraulics fields [3], [4].

3. Fluid-Structure-Interaction (FSI): the fluid and structural vibrations are coupled to each other. Applications in Automotive Systems, Nuclear Power Plants (NPP), Biomedical Applications, etc. [5], [6], [7].

4. Thermo-mechanical coupling: the temperature distribution affects the structural deformation and vice versa [8] [9].

Solution methods for non-linearly coupled multi-physics phenomena occurring have often relied on operator-split coupling strategies that introduce several types of errors in the solution fields. The new paradigm shift for multi-physics simulations is to quantify and reduce the sources of the errors due to the discretizations in space and time and the resolution technique used to solve the non-linear coupling between the physics models. This requires stable and accurate numerical schemes that can tackle non-linearly coupled, stiff multi-physics problems arising from the discretization of the various physics Partial Differential Equations (PDEs) with widely varying characteristic time and length scales. The use of verified physical models for problems of interest and the accurate resolution of these characteristic physical scales are not trivial. This work is aimed at combining consistent numerical methods with principles in software engineering to create a coupled physics framework that is verifiable and can help better quantify these simulation errors.

Numerical simulation using computers is considered as the third pillar of science, besides theory and experiments. This dependence on computers as a virtual laboratory has been recognized in recent years, due to rapid growth in computer speed and affordable memory. Hence, cost effective development and design of scientific applications can be considerably accelerated by the use of simulations on powerful computing systems. In order to make use of solutions from computer simulations for multi-physics problems, it is important to predict the behavior of these non-linearly coupled systems. Predictive science, defined *as the development and application of verified and validated*

*(V&V) computational simulations to predict the properties and dynamic response of complex systems particularly in cases where routine, separable, experimental tests, while important, are difficult.* This definition, borrowed from the Predictive Science Academic Alliance Program (PSAAP) of the US DOE-NNSA [10], is applicable to a wide variety of scientific and engineering applications. This emphasis on predictive capabilities has resurged the need to create and utilize robust numerical techniques for solving coupled problems with high resolution.

The current work focuses primarily on the development and usage of existing analysis and numerical methods for creating a unified and verified tool with predictive capability in the field of multi-physics nuclear reactor computation. Typically the current practice of multi-physics simulations in reactor applications, combines models or algorithms from a diverse set of disciplines. The path towards the predictability of such computations requires the effective integration of both software and numerical methods. Generally speaking, the three pillars of predictive science include code verification, model validation, and uncertainty quantification in the computed solution.

The project is concerned with the development of a multi-physics software platform and the verification of numerical methods for multi-physics applications and an application of uncertainty propagation. Verification is typically an exercise in mathematics, where one assures that the equations are solved correctly, i.e., the software has been coded precisely and implemented according to the physics specifications and requirements. This is an integral part of any software development cycle for simulating physical phenomena.

The need to quantitatively predict the behavior of physical phenomena requires that the sensitivity of the solution fields to uncertainties in the parameters involved in the simulated physical models need to be ascertained. If not, the value of the simulations in comparison to real world experimental results is limited. Noting these as the basic requirements for a complex multi-physics code, we shall systematically develop relevant physical models and use efficient, high resolution numerical techniques in the current work.

In the next subsection, a short background on the current state of coupling methods is provided along with an introduction to the coupling methods that are implemented in the current work.

## 1.1 Background

Let the non-linear vector-valued function representing a coupled PDE system be written in a general form as

$$\mathbf{F}(\mathbf{y}) = \mathbf{N}(\mathbf{y})\mathbf{y} - \mathbf{b} = 0, \tag{1.1}$$

where $\mathbf{y}$ is the solution vector that is dependent on both space and time respectively and $\mathbf{F} : \mathbf{R}^n \to \mathbf{R}^n$, $\mathbf{F}$ is the non-linear operator representing the coupled system and $n$ is the total number of unknowns. For ease of comprehension, we can write $\mathbf{F}$ as in the second equality of Eq. (1.1), where $\mathbf{N}$ is also a non-linear operator and $\mathbf{b}$ is the load vector. It helps to represent $\mathbf{y}$ as a vector comprised of the solution vector for each of the $M$ physics components involved, i.e., $[\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M]^T$. A similar definition holds for $\mathbf{F}(\mathbf{y})$ and its $m$-th component is the non-linear residual stemming from the $m$-th physics component and may depend effectively on all other fields, e.g., $\mathbf{F}_m(\mathbf{y}) = \mathbf{F}_m(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M)$.

In the next subsections, the application of different coupling strategies to resolve the non-linear problem in Eq. (1.1) is presented.

### 1.1.1 Loose Coupling Strategies

In the past few decades, high fidelity modeling of non-linear multi-physics problems has been subdivided into several distinct domains of physics and solved individually as mono disciplinary blocks with specialized codes, without rigorous coupling between the different physics. Although naive, this coupling strategy, mathematically described as Operator-Split (OS) technique, is widely used. With the advent of Parallel Virtual Machines (PVM) and Message Passing Interfaces (MPI) in the 1990's, the OS coupling of several existing specialized single physics codes has become the main multi-physics paradigm in reactor analysis. This kind of modeling is based on coupling several existing specialized mono-disciplinary codes using a 'black-box' strategy, where the input of one code is the output of other, thereby producing solutions that are weakly coupled. The schematics of such models is shown on Fig. 1.1, where the system of PDEs arising from the spatial and temporal discretization of physical models is decomposed into simpler sub-problems. Each physics component is solved by an independent, specialized single-physics code and the data between codes is exchanged through message passing paradigms. Often, this strategy is non-iterative and the non-linearities due to the coupling in between the physics components are not resolved over a time step, reducing the overall accuracy in the time stepping procedure to first-order $O(\Delta t)$, even though high-order time integration might have been used for the individual physics components; see [11, 12]. Note that this explicit linearization of the problem in the OS strategy does not resolve the non-linearities between the different physics. Yet, these isolated physical models in reality describe physical phenomena that are tightly intertwined and rely heavily on the solution field of one another.

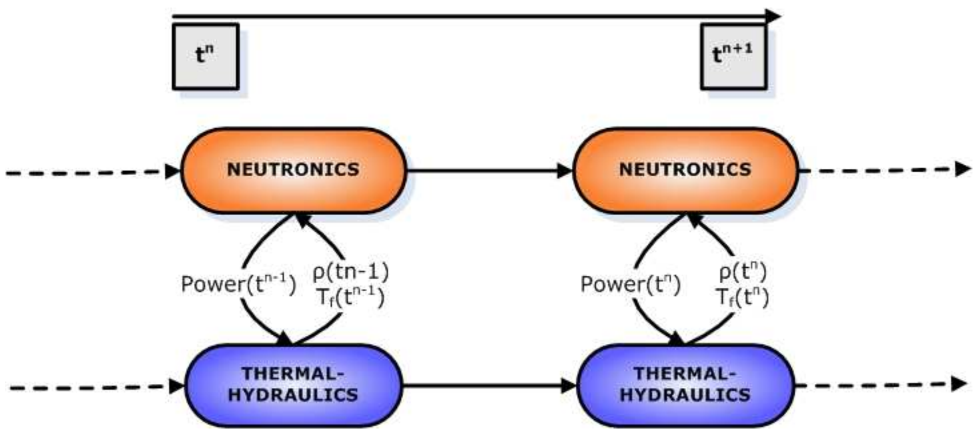
For illustration, consider the non-linear coupled system shown in Eq. (1.1). In OS loose coupling strategy, the non-linear operator is linearized as follows through an explicit treatment:

$$\mathbf{F}(\mathbf{y}^{\ell+1}) = \mathbf{N}(\mathbf{y}^{\ell})\mathbf{y}^{\ell+1} - \mathbf{b}, \tag{1.2}$$

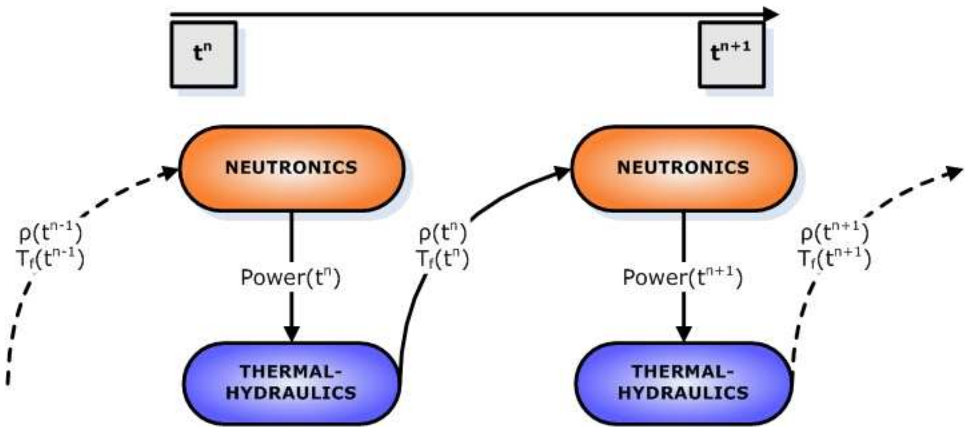
Hence the new update to the solution is obtained by solving the system

$$\mathbf{N}(\mathbf{y}^{\ell})\mathbf{y}^{\ell+1} = \mathbf{b}. \tag{1.3}$$

Although OS allows parts of the problem to be treated implicitly and others explicitly, the lack of iterations in the conventional strategy degrades the solution accuracy in time to first order and the explicit linearization imposes a conditional stability limits for the time-step selection. The direct implication of using smaller time steps to achieve a reasonable accuracy is that the computations need greater CPU time and resources. Despite these drawbacks, this is still one of the major coupling paradigms used today for solving non-linear multi-physics systems.



(a) Simultaneous OS coupling



(b) Staggered OS coupling

Figure 1.1: Two Low-order OS Coupling Strategies

The attractive feature of such a coupling strategy is that the legacy of many man-years of mono disciplinary code development and V&V (validation and verification) is preserved. It is of prime importance to analyze the coupling strategies that can produce highly accurate solutions even in the complex scenarios usually encountered in multi-physics applications. As mentioned before, nuclear reactor analysis is a good example of highly non-linear, coupled multi-physics problem and the non-linearities at the heart of reactor design, analysis and safety calculations provide a good state-space to test high-fidelity numerical methods for multi-physics problems. Physical phenomena such as the ones found in reactor accidents, involve rapidly varying transients that are represented by a stiff system of differential equations. Stiff problems are characterized by solutions having fast varying modes together with slower varying modes, requiring time integrators that can handle such disparate time scales. Stiff problems necessitate the use of implicit time discretization for stability reasons, indicating that OS coupling could prove disadvantageous in terms of efficacy (cost for obtaining a certain accuracy in the solution).

Current examples of OS coupling in the field of nuclear reactor analysis involve the following pairs of neutronics/thermal-hydraulics codes: CRONOS/FLICA [13, 14], PARCS/TRACE [15] and NESTLE/RELAP [16]. Even though more advanced OS strategies exist and can be up to second-order accurate in time, they are complicated to use in coupled legacy codes and hence are not currently employed. For more details regarding these higher order OS schemes, we refer the reader to [17, 18, 19, 20]

### 1.1.2 Tight Coupling Strategies

An alternative to loosely coupled OS strategies is to converge the non-linearities between the physics at every time level to obtain a tightly coupled solution that is consistent with the non-linear system of PDEs. This preserves higher order temporal accuracy of specialized schemes that can be used for resolving the disparate temporal scales in the different physics. Even though the cost/time step can be larger than that of an OS time step, it is essential to stress that the stability of the higher order discretization scheme can be maintained using this procedure, unlike the explicit linearization method where the solution is only conditionally stable.

To devise such a tightly coupled solution procedure, a non-linear iterative scheme needs to be applied to solve the coupled physics and converge the non-linearities to within user's specified tolerances. Two techniques for non-linear system of equations are mentioned next: the Fixed-point or Picard iteration technique and the well known Newton's method.

#### 1.1.2.1 Picard Iteration

Picard iteration technique is a simple non-linear iterative method can be used to converge the non-linearities over the different physics when an OS coupling technique is employed to couple multiple physics codes. Picard iterations can restore the convergence order of a higher order scheme and eliminate the loss of accuracy due to the crude explicit linearization in loosely coupled strategy. The schematic for such a method is shown in Fig. 1.2. This essentially involves iterating over the solution obtained by successively solving Eq. (1.3).

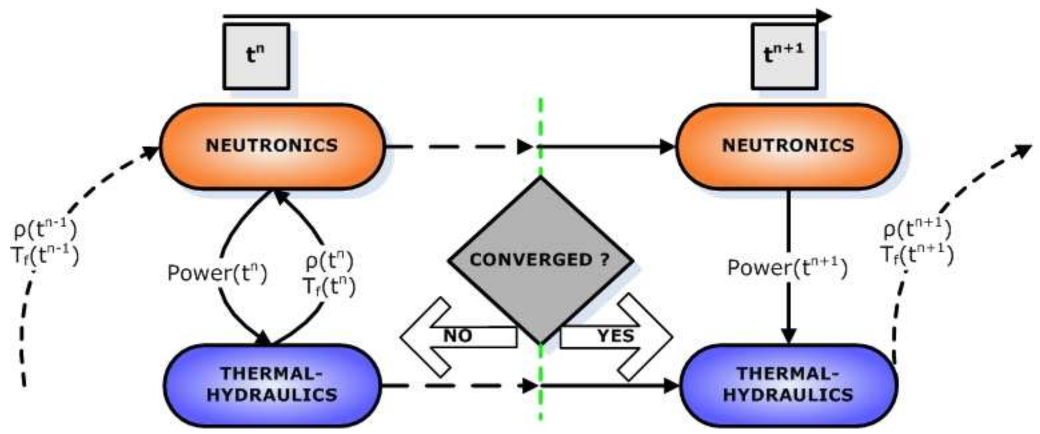


Figure 1.2: High-order, Converged OS Coupling Strategy

The advantage of such a coupling scheme is that it is non-intrusive and can easily use existing framework of codes to obtain a tightly coupled solution. But the primary disadvantage of using such a strategy to restore the accuracy is the increase in computational cost and memory usage to converge the solution.

Since Picard iteration is only linearly convergent, some form of non-linear acceleration techniques are necessary to make this scheme efficient and feasible [11]. Previous research using Aitken's iterated $\Delta^2$ technique suggests that usage of such acceleration schemes can be advantageous and efforts to apply Wynn-Epsilon [21] and other schemes should be pursued as future extensions.

#### 1.1.2.2 Newton Iteration

Current OS strategies may offer flexibility in the way the different physics are solved but involve complexities in terms of resolving the non-linearities and finding an high-order accurate solution. Instead, the coupled non-linear problem can be tackled by recasting it as a root finding problem, in a form amenable for the application of Newton-type methods.

Applying a Newton's method to system of equations in Eq. (1.1), we obtain the following recurrence equation:

$$\mathbf{J}(\mathbf{y}^{\ell})\delta\mathbf{y} = -\mathbf{F}(\mathbf{y}^{\ell}) \tag{1.4}$$

$$\mathbf{y}^{\ell+1} = \mathbf{y}^{\ell} + \delta\mathbf{y}, \tag{1.5}$$

where $\ell$ is the Newton iteration index, $\delta\mathbf{y}$ is the solution update, and $\mathbf{J}(\mathbf{y}^{\ell})$ is the Jacobian matrix evaluated at $\mathbf{y}^{\ell}$. The Jacobian matrix is defined as

$$\mathbf{J}(\mathbf{y}) = \frac{\partial \mathbf{F}(\mathbf{y})}{\partial \mathbf{y}}. \tag{1.6}$$

Note that in Eq. (1.6), the Newton linearization accurately accounts for the true Jacobian of the non-linear system while the OS linearization in Eq. (1.3) neglects a term in the Jacobian matrix expansion ($\mathbf{J}(\mathbf{y}) = \mathbf{N}(\mathbf{y}) + \partial_{\mathbf{y}}\mathbf{N} \cdot \mathbf{y} \approx \mathbf{N}(\mathbf{y})$). This additional term contributes to the stability and robust convergence properties of the Newton iteration as compared to the Picard iteration shown earlier.

At each Newton's iteration, a linear system of equations involving the Jacobian matrix, Eq. (1.4), needs to be solved. As the number of physics components grows, so do the total number of unknowns, resulting in a large memory usage to store the Jacobian matrix. However, employing a Jacobian-free approximation avoids the need for the expensive Jacobian calculation and storage of the matrix since only the action of the Jacobian on a vector is needed to solve the linear system.

Noticing the similarities between the tightly coupled methods with Picard iteration by solving recursively Eq. (1.3) and with Newton iteration by solving Eq. (1.4), we introduce a unified framework that is referred henceforth as the Matrix-free Non-linear Krylov (MFNK) method. Note that this is based on the Jacobian-free Newton-Krylov (JFNK) method proposed by Brown and Saad in the early 1990's [22], that has enjoyed much success in recent years in several multi-physics applications [23]. When Newton's iteration is used as the non-linear solver, MFNK reduces to the original JFNK technique.

Several researchers have analyzed (a) the applicability of this tightly coupled method to obtain high-order accurate solutions and (b) the feasibility of the method in terms of total computational cost [24, 25, 26]. These prior results indicate that this scheme can tackle the widely varying time scales occurring in multi-physics problems efficiently, as compared to an OS coupling strategy. Note that the application of these tight coupling methods based on Picard or Newton iteration is not only limited to PDEs written in the conservative form alone as in Eq. (1.1).

With the aforementioned background ideas, the motivations for the current research work is laid out next.

## 1.2 Research Motivation

To overcome the issues stated in section 1.1, a fully implicit treatment of the coupling terms needs to be used to preserve accuracy and obtain unconditional stability. The difficulties in implementing such a scheme is that the spatial and temporal discretizations of all the physics need to be non-linearly consistent. With such discretizations, the coupling terms in the physics are also treated implicitly and hence higher order accuracy is ensured by resolving the non-linearities accurately. In the current work, a new code system is created based on the MFNK framework with higher order spatio-temporal schemes for all the physics in addition to the ability to simultaneously test OS coupling schemes side-by-side. Also, most existing mono-disciplinary codes for reactor physics simulation were written one to three decades ago to run on computers that existed during that period. Due to the current advances in computing, it would be rather imprudent to develop a new multi-physics code that does not take advantage of the state-of-the-art multi-core, multi-processor parallel architectures that are available now and with expandability to more advanced technologies in the future.

Predictability of the solution is a driving factor in this research and hence it is imperative to obtain a completely verifiable code where the numerical convergence order from the spatial and temporal treatment of the coupled PDEs can be measured against the theoretical orders seamlessly. With computational efficiency in mind, the matrix-free approach through MFNK for the non-linear solve eliminates large storage requirements of the discretized systems and competent numerical and physics based preconditioning techniques [27] can be used to considerably reduce the cost of the linear Krylov iterations. Previous work using JFNK for non-linear diffusion-reaction and advection-reaction problems [28] have shown promising results and serve as the basis for the new coupling strategy being implemented here. It is expected that such a scheme will enable achieving the higher orders of spatio-temporal accuracy for all coupled solutions.

The prime motivation behind the new code is not to employ high fidelity physics models coupled to each other with high resolution but rather to create consistent coupling methodologies that can test the feasibility of using physics-based preconditioned MFNK schemes for real-world problems in reactor design and safety analysis.

## 1.3 Work Organization

The layout for this project is as follows: in Section. (2) we discuss the equations for the physics models used to describe nuclear reactor cores and the governing relations that couple the different physics. In Section. (3) we provide a detailed overview of the different spatio-temporal discretizations, the numerical techniques based on JFNK scheme, and the preconditioning methods employed to reduce the number of Krylov iterations. In Section. (4), a new code system that implements the physics models of Section. (2) and the numerical methods of Section. (3) is introduced and details regarding the software architecture are provided. Next, the code system is put to test using problems created with Method of Manufactured Solutions (MMS) and benchmarks to verify higher order treatment of all the physics models in Section. (5). Finally, in Section. (6) we discuss the details of using the MFNK framework to solve eigenvalue problems occurring commonly in nuclear reactor analysis and compare it to state-of-art schemes like Arnoldi and Jacobi-Davidson iterations. Also, the MFNK technique is applied to a stiff non-linear multi-physics problem based on a radiation diffusion physics model to emphasize the flexibility of applying the implemented code for problems not related to nuclear reactor simulations. Then, we draw conclusions and point out avenues for future research in Section. (7).

# Chapter 2

# Physics Models

'The more you see how strangely Nature behaves, the harder it is to make a model that explains how even the simplest phenomena actually work.'
– Richard Feynman

In this section, details regarding the physics models used in this work are provided. All models have been deliberately chosen to be of "coarse" fidelity as the purpose of this research work is not to validate the physical models themselves but to present a multi-physics verification study that will help develop better intuition regarding efficient coupling strategies. At a later stage, when the details about the implementation code are given, a description will be provided for employing higher fidelity physical models interchangeably within the MFNK framework, when they are deemed necessary for the physics being solved.

In the realm of reactor analysis, there are three primary domains of physics that play a pivotal role in determining core operation and safety. These are:

1. Neutronics - Describes the neutron population distribution and the interaction of neutrons with the material in the reactor core. The primary solution fields calculated is the scalar flux as a function of position, time and neutron energy.

2. Thermal conduction - Describes the distribution of temperature in the fuel pin due to the sensible heat generated from the energetic neutron fission reactions. The solution fields of interest are usually the temperature profile in the fuel element from which the peak fuel temperature and the maximum clad temperature at the surface of the pin are obtained.

3. Coolant channel flow - Describes the flow of coolant fluid through the core that removes the thermal energy from the fuel pins. The models used can be for single or multi-phase fluid to calculate the density, momentum in all directions, total energy, temperature and the pressure drop across the core.

Other physics components include structural mechanics that describes the behavior of thermal expansion in the fuel pins and structures comprising the core, kinetics of chemical reactions occurring due to flow of borated water in PWRs. In the current research, only the three basic physics models listed above have been used to create a multi-physics model to analyze nuclear reactor transients.

## 2.1 Neutronics

Neutronics is the branch of physics that deals with the calculation of neutron flux and neutron reaction rates in the different materials inside the reactor core. These reaction rates need to be calculated accurately in order to determine the power produced in a nuclear reactor and to calculate the temperature solution fields, which are strongly coupled to thermal energy generated in the fuel.

High-fidelity description of neutronics is usually provided by a neutron balance equation or the 'neutron continuity equation' for discrete energy groups that describes the neutron population in the phase-space domain. But finding a numerical solution to the neutron scalar flux $\phi$ from the neutron continuity equation is an arduous task in itself, without coupling to other physics, especially when the reactor domain is large and heterogeneous and when many neutron energy groups $G$ and delayed precursor groups $K$ are employed. We base our neutronics model on the time-dependent Multi Group Neutron Diffusion (MGND) equation to solve for the neutron scalar flux.

$$\begin{aligned}\frac{1}{v^g}\frac{\partial\phi^g(\vec{r},t)}{\partial t} &- \vec{\nabla}\cdot D^g(\vec{r},t)\vec{\nabla}\phi^g(\vec{r},t) + \Sigma_{t,g}(\vec{r},t)\phi^g(r,t) \\ &= \sum_{g'=1}^{G}\Sigma_s^{g'\to g}(\vec{r},t)\phi^{g'}(\vec{r},t) + \chi_p^g\sum_{g'=1}^{G}(1-\beta^{g'})\nu\Sigma_f^{g'}(\vec{r},t)\phi^{g'}(\vec{r},t) \\ &\quad + \sum_{j=1}^{J}\chi_{d,j}^g\lambda_j C_j(\vec{r},t) \qquad \forall g\in[1,G], \quad \forall\vec{r}\in\mathcal{D} \qquad (2.1)\end{aligned}$$

The notations used here are standard [29]. The system of equations is closed with appropriate boundary and initial conditions. We can see that the neutron flux $\phi$ is dependent on the position in the core, the energy of neutrons and on time.

A nuclear core is typically composed of hundreds of different materials and isotopes, each with different crosssections. The crosssection of a material is greatly affected by the temperature and density of the material and depends on the energy of the incident neutron. In this coarse grain neutron diffusion model, the heterogeneity of the materials have been averaged to create fuel assembly homogenized material crosssections (piece-wise constant crosssection values per assembly) that preserve the total reaction rates in the core. The crosssections are usually tabulated, or provided in a closed form approximation, as a function of fuel and coolant temperatures (extension to additional parameters, such a boron concentration, void history, control rod history, ..., is straightforward). The tabulated crosssection values are obtained using table look-up and $\mathbf{R}^p$ interpolation, where $p$ is the total number of parameters used.

The Ordinary Differential Equations (ODEs) for the evolution of delayed neutron precursor concentrations are given by

$$\frac{dC_j(\vec{r},t)}{dt} + \lambda_j C_j(\vec{r},t) = \beta_j \sum_{g'=1}^{G} \nu\Sigma_f^{g'}(\vec{r},t)\phi^{g'}(\vec{r},t) \qquad \forall j \in [1,J]. \tag{2.2}$$

The precursor concentration balance is obtained based on the rate of production from fission reactions and losses due to radioactive decay given by the half-life $\lambda$.

The energy production due to the fission or radiative capture events is given by

$$Q(r,t) = \sum_{g=1}^{G} \left[\kappa_f^g \Sigma_f^g + \kappa_c^g \Sigma_c^g\right](\vec{r},t)\phi^g(\vec{r},t), \tag{2.3}$$

where the $\kappa$ coefficients represent the amount of energy released per reaction event, the subscript $f$ represents fission reactions and subscript $c$ represents radiative capture reactions.

The design of reactors is often carried out in Steady-State (SS) where the distribution of the neutron flux is considered to be in equilibrium. In this static state with fissile material present and no external source, the MGND equation reduces to an eigenvalue problem. The dominant eigenvalue of the system, called the effective multiplication factor $k_{\mathit{eff}}$, is defined as

$$k_{\mathit{eff}} = \frac{\text{Number of neutrons in one generation}}{\text{Number of neutrons in the preceding generation}} \tag{2.4}$$

The determination of this parameter is done by solving the following modified form of the MGND equation Eq. (2.1),

$$\begin{aligned} -\vec{\nabla}\cdot D^g(\vec{r},t)\vec{\nabla}\phi^g(\vec{r},t) \quad &+ \quad \Sigma_{t,g}(\vec{r},t)\phi^g(\vec{r},t) - \sum_{g'=1}^{G} \Sigma_s^{g'\to g}(\vec{r},t)\phi^{g'}(\vec{r},t) \\ &= \frac{\chi_p^g}{k_{\mathit{eff}}} \sum_{g'=1}^{G} \nu\Sigma_f^{g'}(\vec{r},t)\phi^{g'}(\vec{r},t) \qquad \forall g \in [1, G] \end{aligned} \tag{2.5}$$

with appropriate boundary conditions.

This generalized eigenvalue problem relates the fundamental eigenvalue (dominant) representing the $k_{\mathit{eff}}$ and its corresponding eigenmode representing the scalar flux $\phi^g(r,t)$ for SS conditions. Since the flux is obtained as a solution of the eigenproblem, only the shape of the flux can be ascertained and the magnitude is determined based on the total power load chosen during operation. Criticality provides information for the design of a reactor and also serves as a tunable parameter to determine the conditions for continuous power output.

In the current work, the statics are governed by Eq. (2.5) and the dynamics of solution field evolution is described by the MGND equation Eq. (2.1) and precursor equations Eq. (2.2). These closed set of equations, along with boundary and initial conditions, form the neutronics model of this work.

## 2.2 Thermal Conduction Model

The energy production due to the fission reaction in the fuel elements generate sensible heat energy which is deposited locally in the fuel. This energy is conducted outward towards the surface of the fuel pellet, the gap and the outer cladding so that it can be transferred to the coolant. The conservation equation to model this physics in Cartesian coordinates ($\vec{r} = x, y, z$) can be written simply as

$$\rho(T)C_p(T)\frac{\partial T(\vec{r},t)}{\partial t} - \vec{\nabla}\cdot k\vec{\nabla}T(\vec{r},t) = q(\vec{r},t), \tag{2.6}$$

with appropriate boundary conditions on the outer trace of domain $\mathcal{D}$.

Here, the density ($\rho$) in $kg/m^3$, specific heat ($C_p$) in $J/kg -^{\circ} C$, and conductivity ($k$) in $W/m/^{\circ}C$ can depend on the temperature $T(\vec{r},t)$ and hence Eq. (2.6) is a non-linear equation by itself.

The boundary term coupling the conducting solid to the fluid is given by

$$k(T)\partial_n T|_w = h_c(T_w, T_f)(T_w - T_f), \tag{2.7}$$

where $T_w$ ($^{\circ}C$) is the (solid) wall temperature, $T_f$ ($^{\circ}C$) is the coolant temperature at the interface, and $h_c(T_w, T_f)$ ($W/m^2/^{\circ}C$) is the convective heat transfer coefficient obtained by means of a closure relation.

The non-linear heat conduction model employed here represents the core as a porous medium where the fuel, the fluid flowing in the channel and the supporting structures are homogenized together and properties found accordingly. This is stricty used to verify and to test the code implementation since the model used is described typically in the same space as neutronics and the equation is a simple scalar non-linear parabolic equation with non-constant heat source and mixed or Robin BC at the solid-fluid interface.

As a refinement of the porous model described above, a two dimensional diffusion-reaction equation in cylindrical coordinates can be used to find the fuel profile in a pin with the given average power density distribution. This model is described by

$$\frac{\partial \rho C_p T(\vec{w},t)}{\partial t} - \frac{1}{r}\frac{\partial}{\partial r}\cdot(r k_r \frac{\partial T(\vec{w},t)}{\partial r}) - \frac{\partial}{\partial z}\cdot(k_z \frac{\partial T(\vec{w},t)}{\partial z}) = q_{avg}(\vec{w},t), \quad (2.8)$$

where $q_{avg}(\vec{w},t)$ is the average power density in a fuel pin.

Using this model, the average temperature profile and behavior of a region (traditionally a full assembly or part of it in a lattice) can be ascertained and used to find parameters to estimate the peak clad temperature (based on oxidation limits) and other safety parameters such as the maximum fuel temperature in order to eliminate the possibility of fuel melting. A sample schematic 1-d subchannel model that is traditionally used in reactor analysis codes is shown in Fig. 2.1.

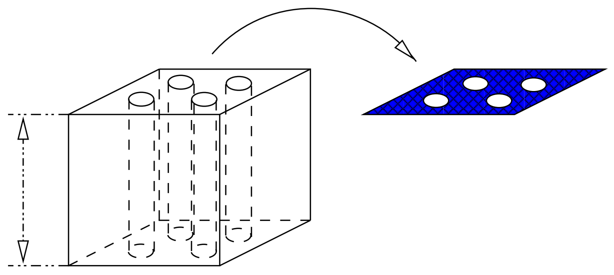


Figure 2.1: Subchannel Model

In such a subchannel model, the average power density corresponding to an assembly location, for a given axial region can be calculated to provide the necessary source terms for determining the temperature profile in the fuel pin. This is representative of the average fuel behavior in that region. The fuel surface temperature is coupled also to the coolant flow in the channel and is accounted using appropriate boundary conditions Eq. (2.7).

## 2.3 Coolant Fluid Flow Model

The coolant flowing in a channel outside the clad of the fuel element gains enthalpy by convection and removes heat generated and conducted in the fuel elements. The thermal hydraulics physics and heat conduction are coupled due to the heat transferred from the fuel pin surface to the coolant by means of convection. The temperature of the coolant is directly dependent on the temperature of the outer clad surface, which, in turn, is a direct function of the fission reaction rate, thereby making all physics coupled to one another. In addition, a volumetric heat source can also be present in the bulk of the coolant to model radiative capture energy release and direct gamma heating.

Typically, in nuclear reactors, the flow of the coolant/moderator fluid occurs in channels of vertical columns. Higher fidelity descriptions may use three-dimensional Navier-Stokes equations in either the conservative or non-conservative variable sets with appropriate turbulence models. In the current work, a simplified approach is taken and the coolant is modeled using a single-phase fluid flowing vertically in one-dimensional channels. The model allows for one or multiple 1-d average channels (the maximum number of channels being the number of right prisms describing the fuel assemblies in the neutronics model; a simple user-defined mapping is employed to assign channels to fuel assemblies). The fluid convects the heat generated either in the bulk of the fluid or at the fuel pin clad interface.

The governing equations for the fluid flow are solved in terms of the conservative variables and are given as:

$$\begin{aligned} \partial_t \rho + \partial_z(\rho u) &= 0 & (2.9)\\ \partial_t(\rho u) + \partial_z(\rho u^2) + \partial_z P &= f_w \rho |u| u + \rho b_f & (2.10)\\ \partial_t(\rho E) + \partial_z(u(\rho E + P)) &= \partial_z q + S, & (2.11) \end{aligned}$$

where $\rho$ is the fluid density, $\rho u$ its momentum, $\rho E$ its total energy, $P$ the pressure, $f_w$ is the ratio of dimensionless wall-friction factor and the hydraulic diameter $D_h$, $q$ the conduction of thermal energy in the fluid, $b_f$ is the net body force acting in the direction of velocity $v$ (for instance, acceleration due to gravity in the downward direction) and $S$ external source terms (energy from the fuel pin) through convective transfer. An equation of state closes the system of fluid equations:

$$\mathbf{P} = f^{\mathrm{EoS}}(\rho, \rho e), \tag{2.12}$$

where the internal energy is $\rho e = \rho E - \frac{1}{2}\rho u^2$.

An example of a closure relation for the equation of state Eq. (2.12) is given by the ideal gas law:

$$\mathbf{P} = \rho e(\gamma - 1), \tag{2.13}$$

where $\gamma = \frac{C_p}{C_v}$, the ratio of specific heat at constant pressure to constant volume. Alternately, a more generic closure relation can be written by means of a linearized relation that is dependent on density and temperature:

$$\mathbf{P} = P_0 + \alpha(\rho - \rho_0) + \beta\left(T - T_0\right), \tag{2.14}$$

where $\alpha, \beta$ are the constants that are valid about the linearization point $(P_0, \rho_0, T_0)$. Note that $\alpha$ is related to the speed of sound in the fluid and provides a simple way to alter the Mach number ($Ma$) in calculations employing manufactured solutions. This is useful in verifying the numerical scheme used to treat this system of equations since they are stiff in the flow regimes of concern in nuclear reactors where low Mach flows dominate.

$$\alpha = \left[\frac{\partial P}{\partial \rho}\right]_0 \propto \frac{1}{Ma^2}, \tag{2.15}$$

As the fluid velocity becomes small in magnitude compared to the speed of sound in the medium, it is very difficult to solve the low-speed flow equations with a conventional compressible algorithm because of their slow convergence. The difficulty in solving the compressible equations for low Mach numbers [30] is associated with the large disparity between the acoustic wave speed and the fluid speed, which contributes to stiffness, resulting in a indefinite system of equations.

Efforts to derive schemes that can tackle all speed flows (from low Mach to supersonic) using physics-based semi-discrete formulations [31, 32], linearized perturbation equations [30] and methods based on asymptotic expansions (pressure separation formulation) in terms of $Ma$ [33] have been investigated previously. In the current research we consider a variation of the method introduced by Harlow [31] to tackle the stiff and low Mach flow regimes that are encountered in nuclear reactor applications. It is also important to note that the semi-discrete and asymptotic expansion methods share similar traits in tackling low Mach flows and further investigations to derive an elegant relation between these family of solvers is necessary to fully understand their mathematical implications.

## 2.4 Closing Remarks

The aim of the research presented in this project is to focus primarily on using better coupling techniques for a given physical equation model. Hence the physics models chosen in the current work are coarse but descriptive enough to analyze transient problems occurring in nuclear reactors.

# Chapter 3

# Methods for Multi-physics Simulations

‘Knowing thus the Algorithm of this calculus, which I call *Differential Calculus*, all differential equations can be solved by a common method... not only addition and subtraction, but also multiplication and division, could be accomplished by a suitably arranged machine.’
– Gottfried Wilhelm von Leibniz

In this section, details regarding the numerical methods employed to tackle the coupled physics problems are provided. In addition to these coupling techniques, which include a discussion of solution methods for nonlinear and linear systems and preconditioners, we also describe space and time discretization techniques with adequate references to supporting materials.

## 3.1 Spatial Discretization

Boundary Value problems (BVP) and Initial BVPs for PDEs are often used to model physical phenomena and hence a consistent and accurate discretization of these equations to resolve the length and time scales correctly is pertinent. Parabolic and Hyperbolic systems of PDEs or mixed systems are typically encountered as governing equations for multi-physics applications. Let the bounded solution domain $\Omega$ be in a $d$-dimensional space $\mathbf{R}^d$ with boundary $\Gamma$. Appropriate boundary conditions should also be prescribed in order to close the system and yield a well-posed problem.

There are several options available for treating the spatial terms in these PDEs; Finite Difference (FD), Finite Volume (FV) and Finite Element (FE) methods. All of these methods, in one form or other, rely on replacing the true solution for the original differential equation with a discrete form of the solution using approximate expansions in terms of piecewise (higher order) polynomials. This reduces the problem to a finite system of coupled equations.

The spatial discretization of the mathematical models in the current work is performed using FE methods. This method is based on the variational form of the boundary value problem. The primary reasons for this choice are the ease of use for arbitrary geometries and irregular domains, the ability to employ nonuniform meshes to reflect sharp solution gradients, and the ability to obtain more easily high-order approximations. Also, rigorous a-priori error estimates of the discretization error based on the order of the polynomial basis functions are available, at reasonable costs, and can be used to adapt the finite element mesh to automatically refine/coarsen a subportion of the mesh based on a user-defined accuracy level.

Here, a Continuous Galerkin (cG) FE method [34] is utilized for Elliptic/Parabolic PDEs and a Discontinuous Galerkin (dG) FE method [35] is employed for Hyperbolic systems. Details regarding the variational form and the discrete equations obtained by applying solution approximations for Elliptic/Parabolic and Hyperbolic systems are given in the following subsections.

### 3.1.1 Elliptic Systems: Continuous Galerkin Discretization

Consider a non-linear, second-order BVP given by the following Elliptic PDE

$$-\vec{\nabla}\cdot D(\vec{r},u)\vec{\nabla}u + c(\vec{r},u)u = q \quad \forall \vec{r} \in \Omega. \tag{3.1}$$

where $D(\vec{r},u), c(\vec{r},u)$ are smooth functions with $D(\vec{r},u) \geq D_0 > 0, c(\vec{r},u) \geq 0$ in $\Omega$ and $q \in L_2(\Omega)$ with appropriate boundary conditions specified at the boundary $\Gamma$. The Galerkin weak form of Eq. (3.1) is obtained by multiplying the equation with a test function $v$ and integrating by parts over domain $\Omega$ to obtain

$$\int_\Omega D(\vec{r},u)\vec{\nabla}u \cdot \vec{\nabla}v + v(c(\vec{r},u)u - q)d\Omega - \int_\Gamma vD(\vec{r},u)\vec{\nabla}u \cdot \vec{n}ds = 0 \quad \forall \vec{r} \in \Omega. \tag{3.2}$$

where Green's theorem or divergence theorem given below is employed.

$$\int_\Omega d\Omega \vec{\nabla}\cdot(vD\vec{\nabla}u) = \int_\Gamma vD\vec{\nabla}u \cdot \vec{n}ds, \tag{3.3}$$

with $\vec{n}$ being the outward unit normal vector on the boundary.

The variational form of the above problem is to find the solution $u$ belonging to the Sobolev space $H^1$ such that

$$a(u,v) = (q,v), \quad \forall v \in \Omega. \tag{3.4}$$

where

$$a(u,v) = \int_\Omega (D(\vec{r},u)\vec{\nabla}u \cdot \vec{\nabla}v + c(\vec{r},u)uv)d\Omega - \int_\Gamma vD(\vec{r},u)\vec{\nabla}u \cdot \vec{n}ds \tag{3.5}$$

$$(q,v) = \int_\Omega (qv)d\Omega \tag{3.6}$$

Now, for the purpose of finding the approximate numerical solution, a non-overlapping partition of the domain $\Omega$ is introduced such that $\bigcup_{\mathbf{K}\in\mathbf{T}} \mathbf{K} = \Omega$ and $\mathbf{T}$ is a Triangulation of $\Omega$. For simplicity, an assumption is made that the geometry is exactly represented by the sum of the parts of the finite partition. The discrete solution is sought in the finite dimensional trial space $S_h$ of piecewise continuous polynomial functions of order $p$. For Galerkin FE method, the trial space and the test space are the same but continuity requirements on the test space are usually less restrictive.

Expanding the numerical solution $u$ and the weight function $v$ in terms of basis functions $\Phi(r)$,

$$u(\vec{r}) \approx U(\vec{r}) = \sum_{i=1}^{K} U_i\Phi_i(\vec{r}), \tag{3.7}$$

$$v(\vec{r}) \approx V(\vec{r}) = \sum_{i=1}^{K} V_i\Phi_i(\vec{r}), \tag{3.8}$$

where $U_i$ and $V_i$ are the degrees of freedom for the FE discretization. If the basis functions $\Phi(r)$ are chosen to be interpolatory, e.g., Lagrange basis functions, then the degrees of freedom satisfy $U_i = U(\vec{r}_i)$ and $V_i = V(\vec{r}_i)$.

The finite dimensional form of the problem can now be restated as follows: Find $u_h \in S_h$ such that

$$a_h(u_h, v_h) = (q, v_h), \quad \forall v_h \in S_h. \tag{3.9}$$

Inserting Eq. (3.8) in Eq. (3.9), the following weak form is obtained.

$$\sum_{k=1}^{K} U_k a_h(\Phi_k(r), \Phi_i(r)) = (f, \Phi_i(r)) \quad \text{for } i = 1, \ldots, K \tag{3.10}$$

This discrete system of equations may be expressed in matrix-operator form as

$$\mathbf{f}(\mathbf{U}) = \mathbf{S} + \mathbf{H} - (\mathbf{K} + \mathbf{M} + \mathbf{B})(\mathbf{U}) = 0, \tag{3.11}$$

where the $\mathbf{K}(\mathbf{U})$, $\mathbf{M}(\mathbf{U})$, and $\mathbf{B}(\mathbf{U})$ are operators (vector functions) corresponding to the stiffness (diffusion), mass (reaction), and boundary terms, respectively; $\mathbf{S}$ and $\mathbf{H}$ are the volumetric load vector and boundary load vectors, respectively; and $\mathbf{U}$ is the vector of unknowns that approximates the solution in the domain $\Omega$. If the operators are evaluated using an appropriate linearization, the Jacobian matrix for the non-linear equations system is simply

$$\widetilde{\mathbf{J}}_{\mathbf{elliptic}}(\mathbf{U}) = -(\mathbf{K} + \mathbf{M} + \mathbf{B}), \tag{3.12}$$

where $\mathbf{K}$, $\mathbf{M}$, and $\mathbf{B}$ are now the stiffness, reaction, and boundary matrices (evaluated at the linearization point). Appropriate preconditioners for diffusion or reaction dominated problems based on the knowledge of the physics can also be created based on the above description of the spatial discretization for elliptic problems.

#### 3.1.1.1 Boundary Conditions: Essential and Natural Conditions

Most often in BVPs, three boundary conditions, namely Dirichlet, Neumann and Robin, are employed. To preserve generality, let boundary $\Gamma = \Gamma_D + \Gamma_N + \Gamma_R$. It is neccessary to understand how these conditions need to be included in the variational formulation itself in order to avoid inconsistency in the discretization. The derivation above for non-linear Elliptic/Parabolic problems is general and does not tie itself down to any specific boundary condition. In this section, we will discuss the methods to impose these various conditions for second order elliptic problems for the boundary integral term in Eq. (3.6) with a continuous Galerkin (cG) discretization.

#### 3.1.1.2 Dirichlet BCs

On $\Gamma_D$, the Dirichlet essential boundary conditions are specified as follows

$$u(\vec{r},t) = \alpha(\vec{r},t), \quad \vec{r} \in \Gamma_D \tag{3.13}$$

There are several ways Dirichlet boundary conditions can be imposed. A simple approach, which works for most interpolary bases like the standard Lagrange polynomials used in the current work for continuous Galerkin discretization, is to assign function values Eq. (3.13) directly to the degrees of freedom on the domain boundary $\Gamma_D$. This idea of imposing Dirichlet conditions directly on the solution is 'strong' in the sense that it does not change the Dirichlet solution as a function of the mesh discretization.

Dirichlet conditions can also be imposed with a "'penalty"' method. In this approach, essentially the $L_2$ projection of the boundary values are added to the linear system matrix. The projection is multiplied by some large factor so that, in floating point numeric arithmetic, the existing (smaller) entries in the matrix and right-hand-side load vector are effectively ignored. This leads to modifying the boundary terms $\mathbf{B}(\mathbf{U})$, $\mathbf{H}(\mathbf{U})$ in Eq. (3.11) as

$$\mathbf{B}_i(\mathbf{U}) = \int_{\Gamma_D} \Phi_i \sum_k \Phi_k U_k (1 + \varrho\delta_{ik}) \tag{3.14}$$

$$\mathbf{H}_i(\mathbf{U}) = \varrho \int_{\Gamma_D} \alpha(r,t)\Phi_i \tag{3.15}$$

where $\varrho$ is the penalty parameter, such that $\varrho >> 1$.

#### 3.1.1.3 Neumann BCs

On $\Gamma_N$, the Neumann natural boundary conditions are specified as follows

$$D(\vec{r},u)\frac{\partial u(\vec{r},t)}{\partial n} = \beta(\vec{r},t), \quad \vec{r} \in \Gamma_N \tag{3.16}$$

These conditions are called 'natural' because they are imposed as part of the variational formulation itself. Consider the boundary term in Eq. (3.6) and applying the conditions Eq. (3.16),

$$\int_\Gamma vD(\vec{r},u)\vec{\nabla}u \cdot \vec{n} ds = \int_\Gamma v\beta(\vec{r},t) ds, \tag{3.17}$$

where $\vec{n}$ is the outward unit normal vector on the boundary.

This can be seen as the boundary $L_2$ inner product on $\Gamma_N$. This contribution is added to the boundary operator $\mathbf{H}(\mathbf{U})$ and is imposed weakly on the variational form.

#### 3.1.1.4 Robin BCs

On $\Gamma_R$, the Robin or mixed boundary conditions are specified as follows

$$D(\vec{r},u)\frac{\partial u(\vec{r},t)}{\partial \vec{n}} + \gamma u(\vec{r},t) = \beta(\vec{r},t), \quad \vec{r} \in \Gamma_R. \tag{3.18}$$

Imposing these mixed boundary conditions is very similar to that of the Neumann conditions since it requires same modifications on the variational formulation. Again, take the boundary term in Eq. (3.6) and applying the conditions Eq. (3.18),

$$\int_\Gamma vD(\vec{r},u)\vec{\nabla}u \cdot \vec{n} ds = \int_{\Gamma_R} v(\beta(\vec{r},t) - \gamma_R u(\vec{r},t)) ds. \tag{3.19}$$

This boundary contribution is added to the operators $\mathbf{H}(\mathbf{U})$ and $\mathbf{B}(\mathbf{U})$.

### 3.1.2 Hyperbolic Systems: Discontinuous Galerkin Discretization

Consider a non-linear hyperbolic conservation equation with advection and reaction of the form

$$\vec{\nabla}\cdot\vec{G}(u,\vec{r},t) + c(\vec{r},u)u(\vec{r},t) = q(\vec{r},t) \tag{3.20}$$

where $\vec{G}(u), c(\vec{r},u)$ are smooth functions with $c(\vec{r},u) \geq 0$ in $\Omega$ and $q \in L_2(\Omega)$ with boundary conditions specified on the inflow boundary $u(\vec{r},t) = \alpha(\vec{r},t), \forall\vec{r} \in \Gamma_i$, where $\vec{G}(u)\cdot\vec{n} < 0$ and $\vec{n}$ is the outward unit normal vector on $\Gamma_i$.

Then the Galerkin weak form of Eq. (3.20) is obtained by multiplying the equation with a test function $v$ and integrating over the domain $\Omega$, like in the elliptic case, to obtain

$$\int_\Omega v(\vec{\nabla}\cdot\vec{G}(u,\vec{r},t) + c(\vec{r},u)u - q)d\Omega - \int_\Gamma \vec{G}(u(\vec{r},t),\vec{r},t)\cdot\vec{n}v d\Omega \quad \forall\vec{r} \in \Omega. \tag{3.21}$$

Let the numerical solution $u$ be expanded in terms of basis functions (Legendre polynomials) $\Phi(r)$ that are discontinuous functions of order $p$, defined on the mesh Triangulation $\mathbf{T}$ of $\Omega$, $\bigcup_{\mathbf{K}\in\mathbf{T}} \mathbf{K} = \Omega$.

$$u_k(\vec{r}) \approx U_k(\vec{r}) = \sum_{i=1}^{p} U_i \Phi_i(\vec{r}), \tag{3.22}$$

where $U_i$ are the degrees of freedom of the FE discretization.

Eq. (3.21) can now be rewritten as

$$\sum_K\{-\int_K\{\vec{G}(u)\cdot\vec{\nabla}v\} + \sum_K\{\int_K\{c(\vec{r},\vec{u})uv\} + \int_{\partial K}\{\vec{G}(u)\cdot\vec{n}v\}\} = \sum_K\int_K\{v\,S\}. \tag{3.23}$$

Because of the discontinuous nature of the solution approximation, the true flux $\vec{G}(\vec{u})\cdot\vec{n}$ is not defined at the cell's boundaries and this quantity is usually replaced by a numerical flux $H_K(u^+, u^-, \vec{n})$ which approximates $\vec{G}(\vec{u})$. Here, $u^\pm$ represents the traces on the boundary edges from the interior/exterior of an element $K$.

With the introduction of the numerical flux, the weak form for the dG method can be rewritten as

$$\sum_K\{-\int_K\{\vec{G}(u)\cdot\vec{\nabla}v\}+\int_K\{c(r,u)uv\}+\int_{\partial K}\{H_{LLF}(u^+,u^-,\vec{n})v^+\}\} = \sum_K\int_K\{v\,S\}, \tag{3.24}$$

where $H_{LLF}(u^+, u^-, \vec{n})$ is the Rusanov, or Local Lax-Friedrichs (LLF), numerical flux given by

$$H_{LLF}(u^+,u^-,\vec{n}) = \frac{1}{2}\left\{\vec{G}(u^+)\cdot\vec{n} + \vec{G}(u^-)\cdot\vec{n} + \lambda(u^+ - u^-)\right\}, \tag{3.25}$$

with $\lambda$ the largest eigenvalue (in absolute value) of the Jacobian matrix of $\vec{G}$. For the 1-dimensional non-linear conservation law used to model fluid flow for reactor applications, the eigenvalue $\lambda = \mathbf{sup}\{v_x, v_x + c, v_x - c\}$ where $v_x$ is the velocity in the direction of flow and $c$ is the sound speed that depends on the medium pressure, density and temperature.

Alternately, an Upwind flux can be used instead of the Rusanov flux, where the numerical flux function is given as

$$H_{up}(u^+,u^-,\vec{n}) = \begin{cases} \vec{G}(u^+) & \text{if } \vec{u}\cdot\vec{n} \le 0 \\ \vec{G}(u^-) & \text{otherwise} \end{cases} \tag{3.26}$$

As an aside, it is interesting to note that in higher order accurate dG methods, the choice of Riemann solver is not that crucial to resolve the spatial scales correctly [36]. Hence, in the current work, the Upwind and Rusanov flux function are used [37, 38] as the solver since it is easy and less expensive to implement, whereas many other choices such as the Godunov, Roe, Osher, HLL, HLLC, and HLLE solvers are available. Future tests to affirm the conclusions of these previous results for problems occurring in nuclear reactor analysis will be necessary to validate the current choice of Riemann solver.

In operator notation, Eq. (3.24) can be written in a general form as

$$\mathbf{f}(\mathbf{U}) = \mathbf{G}(\mathbf{U}) + \mathbf{B}(\mathbf{U}) + \mathbf{M}(\mathbf{U}) - \mathbf{S} = 0 \tag{3.27}$$

where the $\mathbf{G}(\mathbf{U}), \mathbf{M}(\mathbf{U})$, and $\mathbf{B}(\mathbf{U})$ are vector operators corresponding to the advection, reaction, and boundary terms, respectively; $\mathbf{S}$ is the volumetric load vector; and $\mathbf{U}$ is the vector of unknowns that approximates the solution in the domain $\Omega$. If the operators are evaluated using appropriate linearization, the Jacobian matrix for the non-linear equations system is simply

$$\widetilde{\mathbf{J}}_{\mathbf{hyperbolic}}(\mathbf{U}) = -\frac{\partial(\mathbf{G}+\mathbf{M}+\mathbf{B})}{\partial \mathbf{U}}, \tag{3.28}$$

where $\frac{\partial \mathbf{G}}{\partial U}, \frac{\partial \mathbf{M}}{\partial U}$, and $\frac{\partial \mathbf{B}}{\partial U}$ are the partial Jacobian of advection, reaction, and boundary operators respectively, evaluated at the linearization point. Forming the Jacobian for the conservation law with higher order dG discretization can be expensive and complicated. But recent work on steady state problems [39] emphasizes that accurate evaluation of the Jacobian matrix can be crucial to speed up convergence of the non-linear Newton iteration. Also, since the numerical flux functions can be arbitrarily chosen for a given problem, it is only required that the derivatives of the numerical flux $H'_{u^+}$ and $H'_{u^-}$ need to be calculated to assemble the Jacobian matrix correctly. Analytic forms for the derivative are sometimes not available directly but a numerical finite difference procedure can be performed to obtain these values. For the

Upwind and Rusanov fluxes, these values are straightforward to compute and the analysis for other types will be left for future work.

Apart from directly computing the Jacobian from the dG residual in Eq. (3.27), approximate Jacobian matrix for preconditioning the linear system can be obtained based on the Implicit Continuous Eulerian (ICE) technique [31], in which a semi-implicit linearization treats the advection operators explicitly. The unknowns are then eliminated through a Gaussian elimination and substitution process, yielding a single pressure-Poisson equation [28]. This formulation is widely used for low Mach flow regimes as a solver by itself and thus could be quite effective when utilized as a preconditioner within the non-linear matrix-free framework used in the current work. Detailed description of the linearized Jacobian matrix obtained via perturbation of the numerical flux and the ICE preconditioner is provided in Section. (3.3.3).

#### 3.1.2.1 Boundary Conditions: Inflow and Outflow

For 1-dimensional conservation laws that resemble the inviscid Euler equations, there are three characteristic speeds corresponding to the eigenvalues of $G'(u)$ [40], namely

$$\lambda_1 = v_x - c, \quad \lambda_2 = v_x, \quad \lambda_3 = v_x + c \tag{3.29}$$

According to the sign of the these characteristics, four different boundary conditions are usually employed at the inflow and outflow boundaries.

1. Subsonic Inflow: $\lambda_i < 0, \quad i = 1, 2$ and $\lambda_3 > 0$
2. Subsonic Outflow: $\lambda_i > 0, \quad i = 2, 3$ and $\lambda_1 < 0$

The supersonic inflow and outflow conditions have not been considered here since the regimes that are dominant in reactor analysis problems for fluid flows are primarily subsonic.

In order to provide details on the application of these boundary conditions, notations regarding the boundary faces need to be specified. Let us first subdivide the boundary $\Gamma$ into the inflow boundary $\Gamma_i$ and the outflow boundary $\Gamma_o$. Then split the element boundary terms into interior and boundary face terms such that $\sum\limits_K \int_{\partial K} = \sum\limits_K \int_{\partial K \setminus \Gamma} \bigcup \sum\limits_K \int_{\partial K \bigcap \Gamma}$. Now define the bilinear form of the weak statement to include the boundary face terms as follows:

$$a_\Gamma(u, v) = \sum_{K \in \Gamma} \int_{\partial K \bigcap \Gamma} H(u^+, u^-, n) v^+ ds \tag{3.30}$$

This boundary term consists of two parts in our case:

$$a_\Gamma(u, v) = a_{\Gamma_i}(u, v) + a_{\Gamma_o}(u, v) \tag{3.31}$$

Depending on the domain boundary, these terms are specified in the weak form as follows:

1. At the inflow boundary $\Gamma_i$, the outer trace $u^-$ is replaced by the given boundary function $g$ as

$$a_{\Gamma_i}(u, v) = \sum_{K \in \Gamma_h} \int_{\partial K \bigcap \Gamma_i} H(u^+, g, \vec{n}) v^+ ds \tag{3.32}$$

2. At the outflow boundary $\Gamma_o$, only one characteristic variable need to be imposed. In many cases, the outflow variable specified is *pressure* $p = p_o$. Hence on $\Gamma_o$, the outer trace $u^-$ is replaced by a modified solution $u^- = u_o^-(u)$. Then

$$a_{\Gamma_o}(u, v) = \sum_{K \in \Gamma_h} \int_{\partial K \bigcap \Gamma_o} H(u^+, u_o^-(u), \vec{n}) v^+ ds \tag{3.33}$$

Often the modified solution depends on the inner trace $u^+$ and prescribed pressure $p_o$ such that $u_o^- = (\rho, \rho v, \rho E(\rho e, p_o))$.

The specification and implementation of these boundary terms are different from that for elliptic PDE. Even though the Dirichlet conditions are specified for each of the solution variables in the inflow boundary, imposing these conditions occur naturally through the use of the numerical flux functions. Even time-dependent Dirichlet conditions do not require special treatment in order to be enforced consistently.

### 3.1.3 Spatial Coupling Error in Multi-mesh Approaches

Often times in multi-physics applications, each physics component is discretized on its own mesh, and the solution field from a given physics needs to be exported onto another mesh. $L_2$ projection or interpolation of the solution between the source and target meshes may cause non-negligible spatial error [41]. In order to minimize the spatial coupling error due to the data transfer between the different physics defined on non overlapping meshes, several techniques have been developed [42]. Jiao and Heath [43] have derived rigorous cost estimates for different remapping methods along with the solution costs. The spatial coupling error due to, for instance, the use of different meshes, is still an ongoing topic of research [44].

In the current research work, we employ high order quadrature rules for the numerical integration of the terms residing on the target mesh, that approximates the spatial integrals to capture the multi-physics solution behavior. This idea is applicable for arbitrary meshes, provided that the solution for a physics can be evaluated at any point based on the expansion of the solution in terms of the basis functions. Then, as the number of quadrature points is increased, the multi-mesh coupling error becomes 'small enough' as compared to the non-linear error that is not resolved in the coupled physics solution.

For illustration, let us consider two physics, indexed by 1 and 2. In the weak formulation, the non-linear residual of physics 1, $\mathbf{f}_1(\mathbf{y}_1, \mathbf{y}_2)$ is multiplied by a test function, $\mathbf{b}_1^j$. The following integral needs to be computed accurately for every cell $K_1$ of physics 1:

$$\int_{K_1} \mathbf{f}_1(\mathbf{u}_1(\mathbf{x}), \mathbf{u}_2(\mathbf{x}))\mathbf{b}_1^i(\mathbf{x})d\mathbf{x}. \tag{3.34}$$

Expanding the solution fields onto the basis functions, $\mathbf{u}_1(\mathbf{x}) = \sum_i \mathbf{b}_1^i(\mathbf{x})\hat{\mathbf{u}}_1^i$ and $\mathbf{u}_2(\mathbf{x}) = \sum_i \mathbf{b}_2^i(\mathbf{x})\hat{\mathbf{u}}_2^i$, and replacing the integral by a numerical quadrature $(w_q, \mathbf{x}_q)$ yield

$$\sum_q w_q \mathbf{f}_1\Big(\sum_i \mathbf{b}_1^i(\mathbf{x}_q)\hat{\mathbf{u}}_1^i, \sum_i \mathbf{b}_2^i(\mathbf{x}_q)\hat{\mathbf{u}}_2^i\Big)\mathbf{b}_1^i(\mathbf{x}_q). \tag{3.35}$$

For identical meshes, the values $\mathbf{b}_1^i(\mathbf{x}_q) = \mathbf{b}_2^i(\mathbf{x}_q)$ are simple to obtain: mapping $K_1$ onto a reference element is advantageous since the basis functions need only to be evaluated once at the quadrature points of the reference element. However, when the meshes are different, (1) the numerical integration needs to be carried out on the physical element $K_1$, and, (2) all the cells of physics 2 overlapping $K_1$ need to be retrieved and the basis functions $\mathbf{b}_2$ need to be evaluated at the quadrature points, $(\mathbf{x}_q)$. For general unstructured meshes, one cannot obtain straightforwardly $\mathbf{b}_2^i(\mathbf{x}_q)$ in the reference element since this involves reverse lookups to find the correct target element for physics 2 containing the physical point. Hence, the numerical integration over a cell is carried out on the real geometry (the actual cell itself), and not on its mapped reference element. Here, high order quadrature rules for each physics are employed along with inverse mapping of the meshes in different physics in order to evaluate the basis functions at the given physical points. This computation is necessary each time the residual for a given physics needs to be evaluated and an efficient linked list data-structure is created to store the required information in memory and speed up the integration over cells.

The current work does not delve indepth into the issues related to coupling the solution fields from several completely different legacy codes developed independently. These scenarios have solutions residing in meshes that conform to spatial scales of each physics and the a-priori determination of the number of quadrature points to perform the $L_2$ projection accurately is difficult. A workaround would be to create a 'super' mesh which is the union of all the individual physics meshes given by $\Omega_h^{Super} = \Omega_h^1 \bigcup \Omega_h^2 \bigcup \ldots, \bigcup \Omega_h^N$. Then, the solution at all the mesh points can be interpolated, projected and used uniformly with affordable loss of accuracy.

It is also important to note that, making use of available degrees of freedom, certain quantities such as total mass and energy need to be conserved through these projections [42]. This needs special attention while devising schemes to project these variables on a different mesh to be coupled with another physics. Since this subject in itself involves considerable research, only the ideas have been proposed here and demonstrations using two physics will be presented in Section. (5).

## 3.2 Time Discretization

Tackling the whole coupled non-linear system provides tremendous flexibility to use high-order implicit time integrators. Implicitness is required for stability due to the great disparity in time scales of the various phenomena involved in the simulations. Even though traditional codes dealing with stiff individual physics systems tend to use semi-implicit (treat fast scales implicitly and others explicitly), these schemes might not be as effective when used in the context of coupled physics problems due to the introduction of time scales that cause increased stiffness. But since the temporal treatment in single physics problems are based on intimate knowledge of the physics, these solution schemes and discretizations serve as excellent preconditioners for fully coupled physics problems.

Consider a vector valued non-linear system of equations $\mathbf{f}$, that is obtained after appropriate spatial discretization using cG or dG FEM for the different physics. This non-linear residual includes all the coupling details, i.e., the contributions from one physics to another is accounted correctly. The large system of time-dependent coupled non-linear ODEs describing the problem can be generally written as

$$\mathbf{M}\frac{d\mathbf{U}}{dt} = \mathbf{f}(\mathbf{t}, \mathbf{U}). \tag{3.36}$$

where $\mathbf{M}$ is the mass matrix resulting from the spatial discretization of the temporal derivative term (the use of a finite difference technique in space or a lumped numerical quadrature results in $\mathbf{M}$ being the diagonal matrix whose entries contain the cell volume).

The initial BVP has an initial solution prescribed at some given time $t_{initial}$. Let the 1-dimensional time domain $\Theta = [t_{initial}, t_{final}]$ be partitioned in to $N$ steps with $\sum_{n=1}^{N} \Delta t_n = t_{final}$.

Without loss of generality, consider a Runge-Kutta (RK) method for temporal discretization represented using the standard Butcher tableaux notation. Then, any RK method can be specified using the following notation:

$$\begin{array}{c|c} C & A \\ \hline & B^T \end{array}, \tag{3.37}$$

where $B = [b_1, \ldots, b_s]^T$, $C = [c_1, \ldots, c_s]^T$, and $A = (a_{ij})_{i,j=1,\ldots,s}$. Let $s$ be the number of intermediary stages for the RK method, and $\Delta t_n$ the size of step $n$. The application of the RK method to Eq. (3.36) yields the solution at $t_{n+1}$ as

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \Delta t_n \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{3.38}$$

where the intermediate vectors $\mathbf{k}_i$ ($i = 1, \ldots, s$) are obtained by solving the following $s$ non-linear systems

$$\mathbf{M}\mathbf{k}_i = \mathbf{f}\left( t_n + \Delta t_n c_i \, , \, \mathbf{U}_n + \Delta t_n \sum_{j=1}^{s} a_{i,j} \mathbf{k}_j \right), \tag{3.39}$$

The above equation shows that the computation of a solution at $t_{n+1}$ involves performing at least $s$ non-linear iterations for one single sweep and it is necessary to converge the stage vectors $\mathbf{k_i}$ in order to obtain the time solution at the end of $n^{th}$ step. Since the derivation is still general, no assumptions have been made about the structure of the Butcher matrix $A$ to simplify the equations. This will be dealt with separately once we have a fully discrete system of equations.

Based on ideas by Hairer [45], a simple substitution of variables is introduced next. Let

$$\mathbf{Z}_i = \Delta t_n \sum_{j=1}^{s} a_{i,j} \mathbf{k}_j. \tag{3.40}$$

Substituting Eq. (3.40) in Eq. (3.39), the modified set of $s$ non-linear problems is

$$\mathbf{M}\mathbf{k}_i = \mathbf{f}\left( t_n + \Delta t_n c_i \, , \, \mathbf{U}_n + \mathbf{Z}_i \right), \tag{3.41}$$

and, by recursion after simplification, this yields the modified non-linear 'temporal' residual equation defined by

$$\mathbf{F}(\mathbf{Z}) = (\mathbf{M} \otimes \mathbf{I_s})\mathbf{Z} - \Delta t_n A \mathbf{f}(t_n + \Delta t_n C,\, \mathbf{U}_n + \mathbf{Z}) = 0, \qquad (3.42)$$

where $\mathbf{Z} = \{\mathbf{Z_1} \ldots, \mathbf{Z_s}\}$ and $\mathbf{f}(\mathbf{Z}) = \{\mathbf{f}(\mathbf{Z_1}) \ldots, \mathbf{f}(\mathbf{Z_s})\}$.

Now that we have arrived at a final non-linear system, the solution to Eq. (3.42) for $\mathbf{Z}$ can be obtained by some form of non-linear iteration, using either Picard or Newton method. Once $\mathbf{Z}$ is found and converged for all $s$ stages, we can substitute in Eq. (3.38) to find the solution at end of time level $n$ using

$$\mathbf{M}\mathbf{U}_{n+1} = \mathbf{M}\mathbf{U}_n + \Delta t_n B^T \mathbf{f}(t_n + \Delta t_n C,\, \mathbf{U}_n + \mathbf{Z}), \qquad (3.43)$$

It is important to note that all derivations leading up to Eq. (3.43) are applicable to explicit and fully-implicit RK methods. Then, the selection of the appropriate RK methods that can handle stiff PDEs [46, 45, 28] is necessary in order to obtain high-order accurate solutions using the above discretization method. These choices are usually based on several optimal properties of the RK methods such as:

1. explicitness vs implicitness.

2. $A$-stability, (absolute stability) determines whether a method is conditionally stable or unconditionally stable for all time step sizes $\Delta t_n$ [47] (i.e., it is the domain $S \in \Re(z)$ such that $S = \{z \in \mathbf{C}; |\Re(z)| \leq 1\}$ where $\Re(z)$ is the method's characteristic polynomial applied to Dahlquist's equation $y' = \lambda y$ and $z = \Delta t_n \lambda$). In coupled physics systems, the disparity in the time scales leads to stiff systems that require $A$-stable methods in order to resolve the behavior of the physics correctly.

3. $L$-stability, is an essential property that indicates the rate of damping of highly oscillatory modes independent of time step size [48] i.e., a method is $L$-stable if it is $A$-stable and $\lim_{z \to \infty} \Re(z) = 0$. This property is crucial to determine the success of a given method for stiff systems since if all modes are not damped quickly over a transient, the solution procedure can become unstable due to oscillations, neccessitating smaller time steps.

4. Efficiency: cost of solution method per time step. This is critical since there needs to be a balance in terms of cost per step ($s$ * Average CPU cost per stage) versus accuracy in solution (*Local* Truncation Error (LTE)) for solving the system.

A RK method of order $p$ with $s$ stages can be compared to the actual Taylor series expansion of a non-linear system, to derive the order conditions. For higher order methods, it gives the user great flexibility in deriving a scheme with optimal order and stability properties to fit the needs of the problem. This plasticity of the method and the ease of adjusting the coefficients to obtain embedded formulas make them attractive to adaptive time-stepping when needed.

Next, specializations for different families of RK schemes will be discussed and the specific changes in the non-linear equation Eq. (3.42) and step solution Eq. (3.43) will be shown.

### 3.2.1 Explicit-RK (ERK) Methods

If the Butcher coefficient matrix $A$ is strictly lower diagonal, i.e., $a_{i,j} = 0, \forall i = 1 \ldots, s, j \geq i$, then the RK method is said to be explicit. This is because the solution for any time step explicitly depends only on the previous solution and stages and hence these methods do not require any non-linear iterations.

All explicit methods are conditionally stable but due to the reduced cost in finding the solutions, they could be valuable when the physics dictates the usage of very small time steps to resolve the temporal scales. This 'asymptotic regime', when the user-specified tolerance for LTE dominates the solution, is suitable for the usage of such schemes.

$$\mathbf{M}\mathbf{k}_i = \mathbf{f}\left(t_n + \Delta t_n c_i \, , \, \mathbf{U}_n + \Delta t_n \sum_{j=1}^{i-1} a_{i,j}\mathbf{k}_j\right) \quad \forall i = 1\ldots, s \tag{3.44}$$

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \Delta t_n B^T \mathbf{K} \tag{3.45}$$

where $\mathbf{K} = \{\mathbf{k}_1, \ldots, \mathbf{k}_s\}$.

Hence ERK methods are easy to implement and have cheap computational cost per step since there are no non-linear iterations or Jacobian matrix solves other than the Mass matrix $\mathbf{M}$ at the end of each stage. However, they have poor stability properties and are unable to resolve very fast changing modes (explicit schemes are not suitable for stiff equations). To overcome this problem and to utilize the advantages of these one step schemes, modifications to the existing ERK schemes can be made, as shown by Eriksson et. al. [49], to extend the stability region.

In the current work, for the sake of completeness, we have chosen to implement Forward Euler (FE), a two stage ERK method of order 2 and the four stage ERK method by Kutta based on $3/8^{th}$ Quadrature Rule of order 4. Apart from these standard schemes, an embedded ERK method, DOPRI by Dormand-Prince [46] with stiffness detection, has been implemented as well.

The notation for naming each of the RK methods is usually given as

RK$p, p'(s)$ where $p$ is the true order of the method, $p'$ is the embedded order and $s$ is the number of stages. With this notation, the Butcher Tableaux for each of the above methods are given below.

$$\begin{array}{c|c} 0 & 0 \\ \hline B^T & 1 \end{array} \tag{3.46}$$

FE 1(1)

$$\begin{array}{c|cc} 0 & 0 & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline B^T & \frac{1}{4} & \frac{3}{4} \end{array} \tag{3.47}$$

ERK 2(2)

$$\begin{array}{c|cccc} 0 & 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & -\frac{1}{3} & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline B^T & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array} \tag{3.48}$$

ERK 4(4)

$$
\begin{array}{c|ccccccc}
0 & 0 & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46832}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
B^T & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
\hline
E^T & \frac{71}{57600} & 0 & -\frac{71}{16695} & \frac{71}{1920} & -\frac{17253}{339200} & \frac{22}{525} & -\frac{1}{40}
\end{array}
\tag{3.49}
$$

DOPRI 4,5(7)

In Eq. (3.49), $\mathbf{E^T}$ is the error estimator coefficient that is useful in obtaining the Local Truncation Error (LTE) for the specified RK method. This is derived along with the optimal higher order ($p$) step coefficients $\mathbf{B^T}$ for the embedded method. Then,

$$\mathbf{E^T} = B^T - \tilde{B}^T \tag{3.50}$$

where $\tilde{B}^T$ are the coefficients for the lower order ($p'$) method. For brevity, $\tilde{B}^T$ have not been shown and can be easily obtained if necessary.

The LTE ($\epsilon$) for such an embedded method is

$$\epsilon_n = \Delta t_n E^T \mathbf{K} + O(\Delta t_n^{p'+1}) \tag{3.51}$$

A-priori estimates for the LTE are useful to create adaptive solution procedures that can change the step size $\Delta t_n$ and order $p$ of the method to reduce the local and global temporal error in the solution based on user specified tolerance. Based on principles in control theory, Gustafsson [50] introduced the PI controller and applied it to adaptive step-size selection for stiff ODE problems. Previous work for reactor problems [11] using these adaptive controllers were successful and hence have been used in the current research for use with embedded methods.

The PI controller predicts the new step size based on the evolution in LTE, the selection of previous step size and a user specified tolerance. Then,

$$\Delta t_{n+1} = \Delta t_n \left(\frac{Tol}{|\epsilon_n|}\right)^{\alpha} \left(\frac{|\epsilon_{n-1}|}{|\epsilon_n|}\right)^{\beta} \tag{3.52}$$

where $\alpha$ and $\beta$ are problem dependent constants. Gustafsson found after some numerical computation that $\alpha \approx \frac{0.7}{min|p,p'|+1}$ and $\beta \approx \frac{0.4}{min|p,p'|+1}$ are usually good choices for stiff problems. The paper cited above provides detailed derivation of the controller and the optimal parameters in Eq. (3.52).

### 3.2.2 Implicit RK (IRK) Methods

Implicit methods are either usually unconditionally stable ($A$-stable) or at least have much larger stability regions than ERK methods. Even if an IRK method is $A$-stable, it may not satisfy the required $L$-stability conditions that are essential to accurately resolve stiff systems of equations. We can also classify IRK methods based on the structure of the Butcher matrix $A$ in to two broad categories. We will discuss each family below along with the implication on the cost for obtaining a solution per time level.

#### 3.2.2.1 Diagonally-Implicit RK (DIRK) Methods

For DIRK methods, the Butcher coefficient matrix $A$ is lower diagonal, i.e., $a_{i,j} = 0, \forall i = 1\ldots,s, j > i$. Note that the diagonal term is non-zero and hence the solution at each stage requires an implicit non-linear solve, unlike with ERK methods.

The equations for the simplified non-linear system Eq. (3.42) at each stage can be modified as

$$\mathbf{F}(\mathbf{Z}_i) = \mathbf{M}\mathbf{Z}_i - \Delta t_n \sum_{j=1}^{i} a_{i,j}\mathbf{f}(t_n + \Delta t_n c_i,\, \mathbf{U}_n + \mathbf{Z}_i) = 0 \quad \forall i = 1\ldots, s \tag{3.53}$$

Then, if the Jacobian matrix $\mathbf{J}(\mathbf{U})$ for the SS residual $\mathbf{f}(\mathbf{U})$ can be computed approximately, the non-linear iteration to compute the solution update proceeds as

$$\hat{\mathbf{J}}(\mathbf{Z}_i^l)\delta\mathbf{Z}_i^l = -\mathbf{F}(\mathbf{U}_n + \mathbf{Z}_i^l) \quad \forall i = 1\ldots, s \tag{3.54}$$

$$\mathbf{Z}_i^{l+1} = \mathbf{Z}_i^l + \delta\mathbf{Z}_i^l \tag{3.55}$$

where $l$ is the non-linear iteration index and the transient Jacobian matrix $\hat{\mathbf{J}}(\mathbf{Z})$ is

$$\hat{\mathbf{J}}(\mathbf{Z}_i^l) = \mathbf{M} - \Delta t_n a_{i,i}\mathbf{J}(t_n + \Delta t_n c_i,\, \mathbf{U}_n + \mathbf{Z}_i^l) \quad \forall i = 1\ldots, s \tag{3.56}$$

Now that we have determined the necessary components to solve the transient non-linear system, the solution to Eq. (3.54), Eq. (3.55) for $\mathbf{Z} = \{\mathbf{Z}_0\ldots, \mathbf{Z}_s\}$ can be obtained. The use of either a Picard or Newton method as a non-linear solver will be discussed in the next section and the focus will be shifted to solve this system efficiently under constraints of memory and time.

Once $\mathbf{Z}$ is found and converged for all $s$ stages, we can substitute in Eq. (3.38) to find the solution at end of time step $n$. This step involves inverting the Mass matrix $\mathbf{M}$ which can be performed using a lumped-mass approach [51] that has been proven to be quite effective for several test problems.

Several DIRK methods possess unconditional stability and optimal properties that help improve the efficiency of solution procedure. For instance, it is advantageous to have the diagonal elements of the Butcher matrix $A$ to be the same i.e., $a_{i,i} = \gamma$. These DIRK methods are popularly called Singly-DIRK (SDIRK) methods. A variation of the SDIRK methods with an explicit first stage, Explicit SDIRK (ESDIRK), was investigated introduced by Kvaerno [52] and investigated further by Kennedy et al. [53] for advection-diffusion-reaction equations. These methods simplify the solution procedure to solving the non-linear system given in Eq. (3.54) since the transient Jacobian matrix $\hat{\mathbf{J}}(\mathbf{t}, \mathbf{U})$ that needs to be inverted is the same in all stages. Hence if a direct method such as $LU$ factorization can be used, then the factorization need be performed only once and utilized for all the stage computations. Note that in this case, the Jacobian is also lagged (computed at start of step).

Based on the analysis of the properties of DIRK methods, few of them have been chosen to be implemented: Backward Euler (BE), Implicit Midpoint (IM), SDIRK2(2), SDIRK3(2), SDIRK3(3) [45]. Note that BE, SDIRK2(2), SDIRK3(3) are $A-, L-$ stable schemes but IM2(1) and SDIRK3(2) are only $A-$stable and not $L-$stable. Since the provision for including arbitrary DIRK methods exists in the framework introduced thus far, any DIRK/S-DIRK method that can be represented by a Butcher Tableau can be tested and used in the software implemented as part of the current work.

$$\begin{array}{c|c} 1 & 1 \\ \hline B^T & 1 \end{array} \tag{3.57}$$

BE 1(1)

$$\begin{array}{c|c} 0.5 & 0.5 \\ \hline B^T & 1 \end{array} \tag{3.58}$$

IM 2(1)

$$\begin{array}{c|cc} \gamma & \gamma & \\ 1 & 1-\gamma & \gamma \\ \hline B^T & 1-\gamma & \gamma \end{array} \tag{3.59}$$

SDIRK 2(2) with $\gamma = 1 - \frac{1}{\sqrt{2}}$

$$\begin{array}{c|cc} \gamma & \gamma & \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline B^T & 0.5 & 0.5 \end{array} \tag{3.60}$$

SDIRK 3(2) with $\gamma = \frac{3-\sqrt{3}}{6}$

$$\begin{array}{c|ccc} \gamma & \gamma & & \\ \frac{1+\gamma}{2} & \frac{1-\gamma}{2} & \gamma & \\ 1 & \frac{-6\gamma^2+16\gamma-1}{4} & \frac{6\gamma^2-20\gamma+5}{4} & \gamma \\ \hline B^T & \frac{-6\gamma^2+16\gamma-1}{4} & \frac{6\gamma^2-20\gamma+5}{4} & \gamma \end{array} \tag{3.61}$$

SDIRK 3(3) with $\gamma = 0.435866521508459$

### 3.2.3 Fully-Implicit RK (FIRK) Methods

FIRK methods have a full Butcher coefficient matrix, i.e., $a_{i,j} \neq 0, \forall i, j = 1 \ldots, s$. One way to solve these systems would be to consider the full block non-linear system, all unknowns from the $s$ stages, i.e., $\mathbf{Z}$ is the unknown instead of $\mathbf{Z}_i$ for individual stages, and perform non-linear iterations on these. Due to memory restrictions for large scale fully discretized problems, this could be prohibitive.

Alternately, an outer iteration can be used in conjunction with ideas for solving DIRK methods, in order to converge the temporal step solution. This procedure is based on splitting the block matrix operator as $A = D + L + U$ where $D, L, U$ are the diagonal, strictly lower triangular and strictly upper triangular terms of the coefficient matrix. With this splitting, a Block Gauss-Seidel (BGS) iteration can be applied to obtain the residual as

$$\mathbf{F}(\mathbf{Z}^l_{ibgs}) = \quad \mathbf{M}\mathbf{Z}^l_{ibgs} - \Delta t_n (L + D) \otimes I_n \mathbf{f}(t_n + \Delta t_n C,\, \mathbf{Z}^l_{ibgs}) - \tag{3.62}$$

$$\Delta t_n U \otimes I_n \mathbf{f}(t_n + \Delta t_n C,\, \mathbf{Z}^l_{ibgs-1}) = 0 \tag{3.63}$$

with the transient Jacobian matrix $\hat{\mathbf{J}}(\mathbf{t}, \mathbf{U})$ given by

$$\hat{\mathbf{J}}(\mathbf{Z}^l_{ibgs}) = \mathbf{M} - \Delta t_n (L + D) \otimes I_n \mathbf{J}(t_n + \Delta t_n c_i,\, \mathbf{Z}^l_{ibgs}) \tag{3.64}$$

where *ibgs* is the BGS iteration number. This iteration can also be relaxed to improve the outer iteration convergence using block SOR scheme but due to the difficulty in determining the optimal relaxation factor for all multi-physics problems, this is left for future work.

Simply put, the solution procedure for FIRK methods involves performing multiple DIRK solves until convergence. Hence the cost of these methods is *cost per DIRK step**Number of *outer iterations*. Due to the cost involved in computing the solution for FIRK methods, this is often not preferred unless extremely stiff problems are encountered.

Hairer [45] notes that collocation methods based on Gauss and Radau quadrature formulas can lead to FIRK methods with excellent stability properties. These methods are in general $A-$ and $L-$ stable and stiffly accurate (do not degrade convergence for stiff problems) [54].

An adaptive method using the RADAU5 scheme with a good error estimator was implemented previously [11] for coupled simulations using Point Reactor Kinetics Equations (PRKE) and lumped hydraulics models and the success of these methods in predicting sudden changes in temporal scales make them attractive. The use of such implicit adaptive techniques will be essential to capture complex waxing and waning of temporal scales from different physics during critical transients [26] and needs further investigation.

The Butcher matrix for the fourth order methods based on Gauss quadratures and third, fifth order methods based on RADAU IIA family are given below.

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline B^T & \frac{1}{2} & \frac{1}{2} \end{array} \tag{3.65}$$

Gauss 4(2)

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline B^T & \frac{3}{4} & \frac{1}{4} \end{array} \tag{3.66}$$

Radau IIA 3(2)

$$\begin{array}{c|ccc}
\frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\
\frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\
1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\
\hline
B^T & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9}
\end{array} \tag{3.67}$$

Radau IIA 5(3)

Until now, the subject of obtaining the solution of a non-linear system was only briefly discussed. This is because the crux of the work in the temporal solution procedure lies solely in this non-linear solve. Details regarding the usage of Picard or Newton iteration as non-linear solvers are provided next.

## 3.3 Methods for Solving Large-scale Non-linear Systems

This section discusses the numerical techniques employed for solving the non-linear equations arising from the fully discretized coupled physics system. By controlling how the non-linearities are resolved, a tight coupling or traditional loose coupling paradigm can be obtained. This allows testing existing coupling strategies and comparing to new tightly coupled methods in terms of accuracy and efficiency since all of these methods can be implemented within the same framework.

The basis for this idea stems from the fact that if the linear operator representing the Jacobian matrix used for solving the non-linear system is block-diagonal, it represents the decoupled treatment of the different physics and collapses to a Picard iteration strategy. This procedure can be iterated to any given tolerance as long as the spectral radius of the linearized operator is less than one i.e., $\rho(\hat{\mathbf{J}}) < 1$. In other words, the convergence through Picard iterations for coupled physics problems is guaranteed if the eigenmodes due to the linearized terms are not dominant. Then, these iterations are a natural formulation for weakly coupled physics models. But if $\frac{\rho(\mathbf{J})}{\rho(\hat{\mathbf{J}})} > 1$ where $\mathbf{J}$ is the consistent fully coupled Jacobian matrix, then the physics are strongly coupled and much smaller time step sizes will be necessary in order to make the linearization valid. Hence, with a combination of time step control and appropriate linearizations, such iterative procedures over the different physics can produce tightly coupled solutions.

The current framework employs Picard or Newton methods (outer non-linear solves) and Krylov methods (inner linear solves) to solve the set of discrete non-linear equations effectively and accurately. The Matrix-Free (MF) approximation can be included such that the algorithm can be implemented without explicitly building the Jacobian matrix needed in the linear solve. Often, building the Jacobian matrix can be costly in CPU time and memory, especially when different physics components reside in multiple codes. The MF nature of the solvers relies on (i) the fact that Krylov solvers build a solution subspace using only matrix-vector operations and (ii) these matrix-vector operations can be approximated using a finite difference formula that does not require knowledge of the matrix elements at all. Nevertheless, Krylov methods may require a certain number of basis vectors to be stored in order to find an accurate solution (i.e., the size of the subspace may be large). The Krylov space size and the overall computing time can be significantly reduced by the use of an appropriate preconditioner for the linear solves. Therefore, the MF non-linear algorithm consists of 3 levels of iterations:

1. Nonlinear iteration,
2. Linear iteration,
3. Preconditioner iteration.

Since the equations and the methods provided here are generic and are applicable to arbitrary non-linear systems, the same scheme can be utilized for solving linear, non-linear single- and multi-physics coupled systems. The following subsections provide details on the three levels of iterations that are part of the framework used to perform these multi-physics simulations.

### 3.3.1 Nonlinear Iteration Methods

Consider a system of non-linear equations of the form

$$\mathbf{F}(\mathbf{Z}) = 0 \tag{3.68}$$

obtained by space-time discretization of a problem with $\xi$ physics components coupled non-linearly to each other, leading to a system of ordinary differential equations. Let us apply the traditional Picard iteration and the Newton iteration introduced earlier to solve the fully-discrete non-linear problem.

#### 3.3.1.1 Picard Iteration

Picard iteration, also known as Fixed Point Iteration (FPI), is a viable and an easy method to implement since it makes use of existing OS coupling paradigm to linearize the coupled physics solution terms. In solving differential equations, Picard iteration is a constructive procedure for establishing the existence of a solution to a discretized system of equations Eq. (3.68), that passes through the fixed point ($\mathbf{Z}_0$).

However, it is not very effective to iterate at every time step to converge the non-linearities in order to restore the higher convergence order. This is due to the fact that the Picard iterations are only linearly convergent and hence the scheme converges slowly to the true solution. Such a solution procedure takes a high iteration cost and usually requires longer computation times. Additional modifications could accelerate the rate of convergence for the vanilla non-linear Picard iterations in order to make it a viable candidate for reactor analysis problems. Schemes such as Steffensen [55] and vector Wynn-Epsilon algorithm [56] can be used to accelerate the convergence rate of the sequence of vectors found using Picard iterations.

Also, by the nature of the Picard linearization, the coupling between the different physics are treated explicitly. The system matrix arising from the space-time discretization of these physics reflect this weak coupling between different physics components. Let $\mathbf{Z}_P$ be the solution fields corresponding to a particular physics $P$. Then, the non-linear residual equation describing the Picard linearization for each physics $P$ can be written by splitting the non-linear contributions from each physics, as:

$$\mathbf{F}(\mathbf{Z}^{\ell+1}, \mathbf{Z}^{\ell}) = \{\mathbf{Z}_P^{\ell+1} - \mathbf{N}_{PP}(\mathbf{Z}_P^{\ell+1})\} - \sum_{\substack{P'=1 \\ P' \neq P}}^{\xi} \tilde{\mathbf{N}}_{P'P}(\mathbf{Z}^{\ell}) \tag{3.69}$$

where $\ell$ is the Picard iteration number, $\mathbf{N}_{PP}(\mathbf{Z}_P)$ represents the non-linear residual describing the individual physics and $\tilde{\mathbf{N}}_{P'P}(\mathbf{Z})$ represents the non-linear residual due to the coupling of physics $P$ with physics $P'$.

Since the diagonal coupled physics terms $P'$ are computed at the previous Picard iterate, the new solution can be obtained by performing the following sequence of iterations:

$$\mathbf{J}_{FPI}(\mathbf{Z}^{\ell})\delta\mathbf{Z}^{\ell} = -\mathbf{F}(\mathbf{Z}^{\ell+1}, \mathbf{Z}^{\ell}) \tag{3.70}$$

$$\mathbf{Z}^{\ell+1} = \mathbf{Z}^{\ell} + \delta\mathbf{Z}^{\ell} \tag{3.71}$$

where $\mathbf{J}_{FPI}$ is simply in this case,

$$\mathbf{J}_{FPI}(\mathbf{Z}) = \begin{pmatrix} \mathbf{N}_{11} & 0 & \cdots & 0 \\ 0 & \mathbf{N}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{N}_{\xi\xi} \end{pmatrix} \tag{3.72}$$

Since the blocks $\mathbf{N}_{ii}$ require only the solution to the single physics itself, this fixed point iteration procedure can be continued to generate a sequence of solutions that converge to the true coupled physics fields $\mathbf{Z}_P$. This Picard iteration procedure has a Jacobian matrix that is Block-Jacobi structured and hence could be feasible to couple existing mono-physics codes. This OS coupling paradigm uses schematically represented by Fig. 1.1 in Section. (1).

The Picard iteration over multiple physics explained above is the least efficient and computationally expensive mode for performing multi-physics simulations although it is easy to implement for coupling existing legacy codes. Alternately, any level of tighter coupling can be enforced by accouting for the knowledge gained about the physics. These variations in Picard linearization involve simply evaluating the non-linear contribution $\tilde{\mathbf{N}}_{P'}$ from physics $P' \to P$ at the current iterate solution $\mathbf{Z}^{\ell+1}$ in Eq. (3.71) and correspondingly including the implicit contribution of the non-linear operators in the Jacobian matrix Eq. (3.72). These modified Picard variants are usually made such that the Jacobian matrix can be represented as a Block-Lower-Triangular or Block-Upper-Triangular matrix which would involve Block-Backward/Forward substitution respectively, in order to obtain the solution for the Picard iteration Eq. (3.71). A representation of the Block-Lower-Triangular Picard linearized matrix is given below.

$$\mathbf{J}_{FPI}(\mathbf{Z}) = \begin{pmatrix} \mathbf{N}_{1,1} & 0 & \cdots & 0 \\ \mathbf{N}_{2,1} & \mathbf{N}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \mathbf{N}_{\xi,1} & \cdots & \mathbf{N}_{\xi,\xi-1} & \mathbf{N}_{\xi\xi} \end{pmatrix} \tag{3.73}$$

#### 3.3.1.2 Newton Iteration

Instead of employing Picard iterations, one can apply Newton's method to solve the non-linear system of equations in Eq. (3.68) and obtain the solution iteratively as follows:

$$\mathbf{J}(\mathbf{Z}^{\ell})\delta Z = -\mathbf{F}(\mathbf{Z}^{\ell}) \tag{3.74}$$

$$\mathbf{Z}^{\ell+1} = \mathbf{Z}^{\ell} + \delta\mathbf{Z} \tag{3.75}$$

where $\mathbf{J}(\mathbf{Z}^{\ell}) = \frac{\partial \mathbf{F}(\mathbf{Z}^{\ell})}{\partial \mathbf{Z}^{\ell}}$ is the Jacobian matrix of the system at the current Newton iterate $\mathbf{Z}^{\ell}$, $\delta\mathbf{Z}$ is the increment update, solution of the linear solve, and the next Newton iterate is given by $\mathbf{Z}^{\ell+1}$.

It is clear that the Eq. (3.74) requires forming the Jacobian matrix explicitly in order to solve the system for $\delta\mathbf{Z}$. In the case where the coupling between the different physics is complex and requires more memory storage, this option may not be feasible. Also, the convergence of Newton's method strongly depends on the consistency of the Jacobian matrix with respect to the residual description.

One may compute a numerical approximation to the Jacobian, based on a finite difference procedure by perturbing $\mathbf{F}(\mathbf{Z})$. Provided that enough memory is available, $\mathbf{J}$ can be built element by element or column by column.

This is usually referred to as the numerical Jacobian. If recomputed at every Newton iteration, it is very expensive in terms of computational time, especially if the size of the non-linear system $N$ is quite large since $\mathbf{F}(\mathbf{Z})$ needs to be perturbed at least $N$ times. The cost of this numerical Jacobian is hence $O(N)$ non-linear residual function evaluations.

Alternately, when storing the entire Jacobian is not feasible due to memory constraints or when the computational cost of forming the numerical Jacobian itself is prohibitive, a matrix-free approach is preferred. Based on the ideas by Brown and Saad [22], the Jacobian-free approach can be used to efficiently tackle the non-linear system where the linear solve can be performed with only the action of the Jacobian matrix on a given vector. Using only this defined operation, the linearized system in Eq. (3.74) can be solved using an efficient linear solver.

Generally speaking, the action of the Jacobian on a given vector $v$ can be computed using the following finite-difference approximation:

$$\mathbf{J}\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{Z} + \epsilon\mathbf{v}) - \mathbf{F}(\mathbf{Z})}{\epsilon} \tag{3.76}$$

where $\epsilon$ is a parameter used to control the magnitude of perturbation.

Note that the accuracy of the approximation depends strongly on the choice of $\epsilon$. A typical simple choice is usually the square root of machine precision $\epsilon^2 = \Upsilon \approx 1E - 16$. Other optimal equations for choosing the perturbation parameter $\epsilon$ have been derived in the reference papers [22, 23]. For completeness, this optimal form of $\epsilon$ is given by

$$\epsilon = \frac{\sqrt{(1 + ||\mathbf{Z}||)\Upsilon}}{||\mathbf{v}||} \tag{3.77}$$

Further analysis done on the optimization of this finite difference parameter by Xu [57], in the context of coupled multi-physics problems, can also be useful to determine the error in the approximation and increase the efficiency of the algorithm explained above.

Other types of finite difference procedures such as, two-sided difference formulas instead of the one-sided difference formula used in Eq. (3.76), can increase the accuracy of the approximation. But such modifications involve extra computational work and increase the number of function evaluations needed for better estimations. Hence, we have only considered the one-sided difference approximation in this current work and the applicability of these alternate Jacobian-free approximations can be analysed in the future.

The exact Newton method involves solving the linear system in Eq. (3.74) exactly, i.e., to a tight tolerance at every Newton iteration. This is a waste of computational effort when the solution to the non-linear problem is far away from the bowl of asymptotic convergence. Hence, an adaptive technique to change the linear tolerance in the Newton iteration based on the non-linear residual amplitude can decrease the CPU cost during the initial stages of the iteration. Such a formulation is super-linearly convergent and approaches quadratic convergence in the asymptotic regime. The linear tolerance for this inexact Newton iteration can be generally chosen as

$$||LinearResidual|| = \left|\left|\mathbf{J}(\mathbf{Z}^{\ell})\delta\mathbf{Z}^{\ell} + \mathbf{F}(\mathbf{Z}^{\ell})\right|\right|_2 < \gamma \left|\left|\mathbf{F}(\mathbf{Z}^{\ell})\right|\right|_2 \tag{3.78}$$

where $\gamma$ is a forcing term, generally chosen to be smaller than unity. Generally the choice of $\gamma$ results in a tradeoff in the number of non-linear iterations versus linear iterations since too large a value results in more Newton iterations or even divergence and too small a value results in more time spent in the linear solver. Several strategies for optimizing the computational work with a variable 'forcing term' $\gamma$ are given in the work by Eisenstat and Walker [58]. Due to the potential savings in this inexact Newton strategy coupled with the Jacobian-free formulation, this non-linear iteration scheme to solve the coupled non-linear multi-physics problem will be used as the primary solver algorithm in the current work. Note that as $\gamma \to 0$, one recovers the exact Newton algorithm.

In addition to the basic inexact Newton iteration, line search strategies to obtain the global solution satisfying the non-linear system can be used. Such modifications can avoid local stagnation and helps to stabilize Newton's method by scaling the update appropriately. This modification is of the form

$$\mathbf{Z}^{\ell+1} = \mathbf{Z}^{\ell} + d^{\ell}\delta\mathbf{Z}^{\ell+1} \tag{3.79}$$

where $d^{\ell}$ is the scaling factor that restricts the update. The standard Newton algorithm is recovered when $d^{\ell} = 1$. Further reading regarding these global line search methods is available in [22, 59, 58]. The methods for linear systems arising in Eq. (3.71) and Eq. (3.74) are discussed next.

### 3.3.2 Krylov Methods for Solving Linear Systems

The linear system obtained from the Picard or Newton linearization applied to the non-linear equation Eq. (3.68) can be efficiently solved using a Krylov method in which an approximation to the solution of the linear system is obtained by iteratively building a Krylov subspace of dimension $m$ such that

$$K(\mathbf{v}, \mathbf{J}) = \text{span}\{\mathbf{v}, \mathbf{J}\mathbf{v}, \mathbf{J^2}\mathbf{v}, \mathbf{J^3}\mathbf{v}, \ldots, \mathbf{J^{m-1}}\mathbf{v}\} \tag{3.80}$$

where $\mathbf{v}$ is the initial Krylov vector.

Most coupled multi-physics problems produce linear systems that are block unsymmetric, even if the individual blocks may be symmetric due to the type of spatial discretization, e.g., Continuous Galerkin for elliptic problems. Hence robust Krylov methods are needed to tackle these unsymmetric systems. Previous studies on the usage of GMRes (Generalized Minimum RESidual), BiCGStab (Bi-Conjugate Gradient Stabilized) and Transpose-free Quasi Minimal Residual (TFQMR) methods to tackle such systems [60] in the context of non-linear multi-physics problems suggest the feasibility of these choices.

In the current work, since a general framework is required to solve large-scale coupled linear systems, the robustness of the linear solver is an important factor in determining the total computational time of the algorithm. It is also necessary that the linear solver used be insensitive to the numerical roundoff and finite difference errors that are created as part of the approximations used in the Jacobian-free formulation. It is worthwhile to note that the use of Arnoldi-type of Krylov iterative methods yields the best convergence since complete orthogonalizations of all the subspace vectors aids in correcting the numerical errors introduced by the finite difference approximation. Although it is not possible to select one efficient linear solver for all types of unsymmetric problems, such an Arnoldi based GMRes solver is expected to be reliable and provide monotonically decreasing residuals.

The success of the GMRes iterative method, introduced by Saad and Schultz [61], and its popularity due to its efficiency in solving nonsymmetric system of equations make it attractive for the usage in tightly coupled multi-physics systems. The GMRes algorithm generates a sequence of orthogonal vectors, and because the matrix being inverted is not symmetric, short recurrence relations cannot be used as in the case of the Conjugate Gradient algorithm. Instead, all previously computed vectors in the orthogonal sequence have to be retained. In current study, the modified Gram-Schmidt algorithm for orthogonalization is used instead of the classical Gram-Schmidt algorithm in order to create a stable solver that is insensitive to roundoff errors. In the GMRes algorithm, one matrix-vector product is required per iteration and the matrix-free approximation introduced earlier in Eq. (3.76) can be used to obtain the action of the Jacobian matrix on any vector. Detailed information on the exact numerics and implementation of GMRes in the MFNK framework can be found in [23].

The cost of the GMRes algorithm strongly depends on the size of its Krylov subspace that is created through the matrix-vector products. The memory cost increases linearly with every iterations and the number of Inner-Products (IP) required for orthogonalization increases quadratically. Hence, when solving large systems of equations, it is necessary to limit the size of Krylov subspace used. To limit the Krylov subspace size, a restarted variant of GMRes algorithm, GMRes($r$), where $r$ is the size of Krylov space, can be employed.

Flexible versions of the restarted GMRes algorithm, FGMRes($r$), are useful in cases where the matrix-vector products are computed inexactly, and a need for robust Krylov solvers that can provide monotonic convergence to the solution is necessary. FGMRes($r$) algorithm differs from the standard preconditioned GMRes($r$) implementation by allowing variations in preconditioning at each iteration. This is especially important since the preconditioned solve at each Krylov iteration is performed inexactly (varying number of iterations or tolerance for each preconditioner solve). Because of these advantages, in the current research, FGMRes($r$) is the preferred linear solver for unsymmetric systems of equations.

Optimizations beyond restricting the size of the subspace $r$ due to memory reasons involve reducing the total number of linear iterations through the use of appropriate preconditioners. A discussion of the preconditioner implementations and the options available for different kinds of physics is provided next.

### 3.3.3 Preconditioners for the Linear Iteration

The preconditioner $P$ is usually a good approximation of the Jacobian and should be easier to form and solve as compared to the Jacobian matrix itself. The inherently two-step process for this stage requires the computation of the action of $P^{-1}$ on any vector $v$, rather than actually forming the preconditioning matrix itself. This algorithm can be made strictly Matrix-Free and studies for real-world problems previously [27] have shown possible increased efficiency when using this approach.

The right-preconditioned Matrix-Free Nonlinear-Krylov (MFNK) algorithm which involves using Eq. (3.76) for the Jacobian-vector products and an appropriate numerical or physics-based preconditioner results in a modified form of the non-linear iteration. The right-preconditioned non-linear equation is given by

$$(\mathbf{J}\mathbf{P}^{-1})(\mathbf{P}\delta\mathbf{Z}) = -\mathbf{F}(\mathbf{Z}). \tag{3.81}$$

The application of the right preconditioner requires only the action of $\mathbf{J}\mathbf{P}^{-1}$ on any Krylov vector $\mathbf{v}$ and has to be performed at each Krylov iteration. This is realized in a two-step process:

1. First, apply the preconditioner and solve for $w$

$$\mathbf{J}\mathbf{P}^{-1}\mathbf{w} = -\mathbf{F}. \tag{3.82}$$

2. Next, the update is obtained by solving the linear system

$$\mathbf{P}\delta\mathbf{Z} = \mathbf{w}. \tag{3.83}$$

The right-preconditioned version of Eq. (3.76) is used to solve Eq. (3.82) and is expressed as follows

$$\mathbf{J}\mathbf{P}^{-1}\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{Z}+\epsilon\mathbf{P}^{-1}\mathbf{v})-\mathbf{F}(\mathbf{Z})}{\epsilon} \tag{3.84}$$

where $\mathbf{v}$ is any GMRes vector. Upon convergence of the linear solve in Eq. (3.82), one more preconditioner application is necessary using Eq. (3.83) to obtain the true Newton update for the non-linear iteration.

Up until now, the algorithm has been described in a general fashion, in the sense that the non-linear residual can be obtained after space-time discretization, the approximate action of the Jacobian on a vector can be computed using Eq. (3.84) and finally an appropriate preconditioner $P$ can be chosen to reduce the conditioning number of the true Jacobian matrix.

Generally, preconditioners can be subdivided into two broad categories: Algebraic and physics-based preconditioners. The former deals with creating approximate sparse inverse factorizations, using numerical strategies to reduce the spectral radius of the linear system being solved. Some examples of such preconditioners include Incomplete Cholesky($\ell$) factorization, Incomplete-LU($\ell$) factorization along with reverse Cuthill-Mckee (RCM) reorderings, Sparse Approximate Inverses (SPAI) [62], Block-Jacobi splitting, Additive-Schwartz methods and Algebraic multigrid [63]. Algebraic preconditioners are often times also referred to as ‘numerical’ preconditioners. Algebraic methods are often easier to develop and use, and are particularly well suited for irregular problems that arise from discretizations involving unstructured meshes of complicated geometries. Furthermore these Algebraic methods can be fine tuned to make use of multi-processor architectures intricately in order gain improved scalability in the solution procedure.

Alternately, with intuitive understanding of the governing physics PDEs, the geometry, boundary conditions and details of the discretization for the problem under consideration, specialized preconditioners, usually based on physics-based OS linearizations, can be devised and used as very efficient preconditioners to damp the dominant modes, thereby leading to a well conditioned system. Multilevel methods usually fall in this category since they solve 'nearby' problems based on lower order discretizations. Few examples in this category of physics-based preconditioners include multigrid preconditioners [27], the method of Diffusion-Synthetic-Acceleration (DSA) [64] often used as a preconditioner for Transport equation and Implicit Continuous Eulerian (ICE) [31] for near-incompressible fluid flow problems. These problems are usually optimal for specific types of problems and might not be effective as generic preconditioners for all scenarios.

Note that in these physics-based preconditioners, the use of Algebraic preconditioners themselves is most often seen and hence such Algebraic preconditioners can be considered as building blocks for more advanced preconditioners. In the current work, both these approaches will be used in a mixed fashion, depending on the problem being solved in order to reduce the total number of linear iterations in Krylov solves. Also, care is needed while using preconditioners in a multi-processor architecture since traditional sequential preconditioners may sometimes fail in these scenarios. Hence, scalable preconditioners that can be used in both sequential and parallel linear Krylov solvers are preferred. A thorough survey of many state-of-art preconditioning methods used in computational physics problems was presented by Benzi [65].

Below, a brief description at some specific physics-based preconditioning techniques used in this research is provided and the reader is referred to previous work on these techniques for further details.

### 3.3.4 Physics-based Preconditioners

Legacy codes written to tackle the mono-physics models typically contain approximations for specific problems that usually result in increased efficiency even with a little loss of generality. A physics-based preconditioner is usually derived by the linearization of the non-linear physics components, in both the Elliptic and Hyperbolic equations based on semi-implicit treatment of the stiff terms. Such intricate knowledge of the physics systems for problems of interest can considerably improve the efficiency of the simulation. In the context of utilizing the MFNK framework introduced earlier, these algorithms that currently exist in such codes can accelerate the linear solver convergence, thereby preserving man-years of testing and verification.

#### 3.3.4.1 Linearized Jacobian for Elliptic Systems

Consider the SS terms in the non-linear Elliptic system shown in Eq. (3.11), linearized about the last non-linear iteration $(*)$ as

$$\mathbf{f}(u^{n+1}) = q - (-\vec{\nabla}\cdot D(\vec{r}, u^*)\vec{\nabla}u^{n+1} + c(\vec{r}, u^*)u^{n+1}) \quad \forall \vec{r} \in \Omega. \tag{3.85}$$

Let us define a new variable as $\delta u = u^{n+1} - u^*$ which represents the true update for the non-linear iteration. Then if the physics-based preconditioner $\mathbf{P}$ approximates the Jacobian matrix for the non-linear Elliptic system, the preconditioner solve is

$$\mathbf{P}(\delta u) = -\mathbf{f} \tag{3.86}$$

Substituting this definition into Eq. (3.85), we obtain

$$\mathbf{f}(u^{n+1}) = q - (-\vec{\nabla}\cdot D(\vec{r}, u^*)\vec{\nabla}(\delta u + u^*) + c(\vec{r}, u^*)(\delta u + u^*)) \quad \forall \vec{r} \in \Omega. \tag{3.87}$$

Expanding and simplifying Eq. (3.87), results in a modified residual equation of the form,

$$\mathbf{f}(u^{n+1}) = \mathbf{f}(u^*) - (-\vec{\nabla}\cdot D(\vec{r},u^*)\vec{\nabla}(\delta u) + c(\vec{r},u^*)(\delta u)) = 0 \quad \forall \vec{r} \in \Omega. \tag{3.88}$$

Hence in a Nonlinear-Krylov iteration framework, the coefficients $D(r,u)$ and $c(r,u)$ are evaluated about the linearized point to yield a linear elliptic equation system and the forcing function (source) for this linear equation for $\delta u$ is $\mathbf{f}(u^*)$. Applying the Continuous-Galerkin FE discretization to Eq. (3.88) results in the standard stiffness and mass matrices along with appropriate boundary conditions applied to the solution. Hence, the preconditioner iteration is simply, in this case,

$$(K^* + M^* + B^*)(\delta u) = \mathbf{f}(u^*), \tag{3.89}$$

and the preconditioner matrix $\mathbf{P} = (K^* + M^* + B^*)$. Once this matrix is formed, a Krylov method such as Conjugate Gradient (CG) or GMRes with appropriate Algebraic preconditioners can be used to effective find the update for the solution $\delta u$.

The linearized Jacobian matrix is an effective preconditioner when the linearization point (*) is closer to the true non-linear solution. The use of Incomplete-Cholesky and ILU factorization for symmetric and unsymmetric systems respectively can considerably reduce the total cost of the preconditioner solve itself. The current work utilizes such a linearized Jacobian matrix in conjunction with Algebraic preconditioners for non-linear scalar/vector elliptic/parabolic equation systems in order to reduce the total cost of FGMRes($r$) Krylov solves.

#### 3.3.4.2 Nearly Incompressible, Low-Mach Fluid Flow Systems

Fluid flows in reactor analysis problems fall under the low Mach ($Ma$) flow regime. In the conservative variable formulation, as the flow velocity of the fluid decreases, it is very difficult or almost impossible to solve low-speed flows with a conventional compressible algorithm because of slow convergence. The difficulty in solving the compressible equations for low Mach numbers is associated with the large disparity between the acoustic wave speed and the waves propagating at the fluid speed, which is called eigenvalue stiffness. To overcome this difficulty, several ideas have been proposed. In the current study, we will specifically use the Implicit Continuous Eulerian (ICE) scheme [31, 32] for solving these low-speed problems. Some theoretical asymptotic analysis on the semi-discrete Euler equations using the ICE scheme using the implicit Backward-Euler method is shown in this section. The extension to higher order ERK/IRK methods is trivial.

Let us start with the non-linear inviscid Euler-like equations for unsteady fluid flow in the conservative form. For generality, a source term is also included. The equation system is given as:

$$\frac{\partial U(x)}{\partial t} + \frac{\partial}{\partial x} F(U) = S(U), \tag{3.90}$$

where

$$U = \begin{bmatrix} \rho : Density \\ \rho v : Momentum \\ E : Total\ energy \end{bmatrix}; \qquad F(U) = \begin{bmatrix} \rho v \\ \rho v^2 + P \\ vE + vP \end{bmatrix},$$

and the source term $S(U)$ is non-zero when solving manufactured problems (for verification studies), when the effects due to friction and gravity are included or when the fluid equations are coupled to energy transfer from a heated surface (conjugate heat-transfer). The pressure $P$ is usually given by the closure relation, the Equation of State (EOS), in a linearized form as

$$P = P_0 + \left.\frac{\partial P}{\partial \rho}\right|_0 \rho + \left.\frac{\partial P}{\partial E}\right|_0 E. \tag{3.91}$$

The spatial discretization is performed using the Discontinuous Galerkin method with appropriate numerical flux functions (Upwind flux or Rusanov flux). Higher order spatial discretizations can be obtained by increasing the polynomial order of the Legendre basis functions.

For simplicity, let us redefine the momentum variable as $M = \rho v$. Then the semi-discrete form of the equations, which are essentially the non-linear residual for the continuity, momentum and energy equations, can be written as:

$r_C(n+1)$ :

$$\frac{\rho^{n+1} - \rho^n}{\Delta t} + \partial_x \left(M\right)^{n+1} = S_C \left(\rho, M, E\right)\Big|^n, \tag{3.92}$$

$r_M(n+1)$ :

$$\frac{M^{n+1} - M^n}{\Delta t} + \partial_x \left(\frac{M^2}{\rho}\right)^n + \partial_x P^{n+1} = S_M \left(\rho, M, E\right)\Big|^n, \tag{3.93}$$

$r_E(n+1)$ :

$$\frac{E^{n+1} - E^n}{\Delta t} + \partial_x \left(M^{n+1} \frac{E^n + P^n}{\rho^n}\right) = S_E \left(\rho, M, E\right)\Big|^n, \tag{3.94}$$

where $S_C$, $S_M$ and $S_E$ are the continuity, momentum and energy source terms.

(Eq. (3.92))-(Eq. (3.94)) are the non-linear residual functions about the point $(n+1)$. Now, choose a linearization point (*) that is typically chosen as the last non-linear iteration, about which a change in the state variable can be defined. This can then be given as

$$\begin{aligned}
\delta\rho &= \rho^{n+1} - \rho^* && (3.95)\\
\delta M &= M^{n+1} - M^* && (3.96)\\
\delta E &= E^{n+1} - E^* && (3.97)\\
\delta P &= P^{n+1} - P^*. && (3.98)
\end{aligned}$$

Substituting these new variables in (3.92)-(3.94), the following conservation equations for the delta form of the state variables are obtained.

$$\begin{aligned}
\frac{\delta\rho}{\Delta t} + \partial_x(\delta M) &= -r_C^* && (3.99)\\
\frac{\delta M}{\Delta t} + \partial_x \delta P &= -r_M^* && (3.100)\\
\frac{\delta E}{\Delta t} + \partial_x\left(\delta M\left(\frac{E+P}{\rho}\right)^*\right) &= -r_E^*, && (3.101)
\end{aligned}$$

where the linearized discrete residuals evaluated at the linearization point $(*)$ are

$$\begin{aligned}
r_C^* &= \frac{\rho^* - \rho^n}{\Delta t} + \partial_x(M)^* - S_C(\rho^*, M^*, E^*), && (3.102)\\
r_M^* &= \frac{M^* - M^n}{\Delta t} + \partial_x\left(\frac{M^2}{\rho}\right)^* + \partial_x P^* - S_M(\rho^*, M^*, E^*), && (3.103)\\
r_E^* &= \frac{E^* - E^n}{\Delta t} + +\partial_x\left(M^*\frac{E^* + P^*}{\rho^*}\right) - S_E(\rho^*, M^*, E^*), && (3.104)
\end{aligned}$$

Note that the advection terms in the momentum Eq. (3.100) and energy Eq. (3.101) conservation equations and along with the source terms have been linearized about the point (*).

Rearranging the momentum equation Eq. (3.100), we obtain

$$\delta M = -\Delta t \partial_x \delta P - \Delta t r_M^*. \tag{3.105}$$

This expression can then be substituted in the continuity Eq. (3.99) and the energy Eq. (3.101) equations to obtain a system of equations in $\delta\rho, \delta E$.

$$\begin{aligned}\delta\rho &= -\Delta t\partial_x(\delta M) - \Delta t r_C^* && (3.106)\\ \delta E &= -\Delta t\partial_x\left(\delta M\left(\frac{E+P}{\rho}\right)^*\right) - \Delta t r_E^*. && (3.107)\end{aligned}$$

Now using the linearized Equation of State (EOS) introduced in Eq. (3.91), we can then substitute

$$\delta\rho = \frac{1}{\left.\frac{\partial P}{\partial \rho}\right|_0}\left(\delta P - \left.\frac{\partial P}{\partial E}\right|_0 \delta E\right). \tag{3.108}$$

Substituting the above equation in Eq. (3.106), we get

$$\delta P = \left.\frac{\partial P}{\partial E}\right|_0 \delta E - \left.\frac{\partial P}{\partial \rho}\right|_0 \left(\Delta t\partial_x(\delta M) + \Delta t r_C(*)\right). \tag{3.109}$$

Rearranging the above equation and substituting Eq. (3.107) and Eq. (3.105) for $\delta M$ and $\delta E$ respectively, we get the semi-discrete form of the pressure Poisson equation, given as

$$\begin{aligned}\delta P &= \Delta t \left.\tfrac{\partial P}{\partial E}\right|_0 \left(\Delta t\partial_x\left(\partial_x\delta P + r_M^*\left(\tfrac{E+P}{\rho}\right)^*\right) - r_E^*\right)\\ &\quad + \Delta t \left.\tfrac{\partial P}{\partial \rho}\right|_0 \left(\Delta t\partial_x\left(\partial_x\delta P + r_M^*\right) - r_C^*\right), && (3.110)\end{aligned}$$

This system shown in Eq. (3.110) can be solved for $\delta P$ and back-substituted to obtain $\delta M$ from Eq. (3.105), $\delta E$ from Eq. (3.107), $\delta\rho$ from Eq. (3.106) respectively. It is important to note that the new system expressed as an elliptic pressure equation is exactly same as the original semi-discrete ICE linearized Euler equations. It is quite clear that by doing the algebraic manipulation shown in Eq. (3.108) for EOS and substitution of Eq. (3.105) into the continuity and energy equations, an equivalent Gaussian elimination on a system of size $4N$ $(\delta\rho, \delta E, \delta M, \delta P)$ has been performed analytically to convert it to a block upper triangular form that is solved by back substitution.

Hence solving the original ICE system Eq. (3.99) – Eq. (3.101) and the elliptic pressure equation Eq. (3.110) do yield the exact same result as long as the spatial discretization of the PDE's are consistent in both cases. Since the pressure waves are resolved with an ICE solve, it results in eliminating all the dominant eigenmodes occurring due to the pressure waves, i.e., acoustic scales in the medium. Hence, the resulting system has a smaller spectral radius, especially for low-Mach flows where the spread between the eigenvalues in the original non-linear fluid flow equations is the quite large.

The gain in computational time when using ICE as a preconditioner and as a solver by itself has been shown previously in [66]. Now let us consider the advantages and disadvantages of the ICE preconditioner of size $N$ introduced earlier (denoted hereafter as $N$-ICE).

**Pros**

1. The elliptic pressure matrix in the fully discrete form is clearly only $N \times N$ while the original equation system was a $3N \times 3N$ hyperbolic system. The gain in terms of reduction in the size of the system, without any major approximations or loss of accuracy therein makes this a valuable method in low-Mach regime flow calculations.

2. Additional cost savings in terms of forming and solving the modified linear system in Eq. (3.110) using Algebraic preconditioners can significantly decrease total Krylov iterations for solving the Jacobian matrix.

**Cons**

1. It is difficult to maintain the consistency of the fully discrete ICE system w.r.t. the original dG discretization of the non-linear hyperbolic system, due to the requirements to evaluate the derivatives of momentum residuals in the right hand side in Eq. (3.110). Care is needed if a consistent preconditioner is to be created from the $N$-ICE system.

The $N$-ICE solver can typically be used as a solver by itself but the semi-implicit treatment leads to conditional stability only. However, when used as a preconditioner, the updates provided by such a linearization approximate the updates necessary for the outer Newton iteration. Hence in low-Mach regimes, these schemes are valuable to resolve the stiffness in the linear system quickly and, hence, act as efficient preconditioners to reduce the total linear iterations, thereby requiring fewer actions of the Jacobian on a Krylov vector.

## 3.4 Closing Remarks

In this section, we have covered the space-time discretization methods for different PDE systems and described the process along with the constraints to resolve the different spatial and temporal scales in multi-physics computations. Based on these discretizations, the algorithm for a Matrix-free non-linear iteration method based on finite-difference approximations was introduced. The available options for using robust linear solvers along with different kinds of preconditioning techniques to increase overall efficiency of the algorithm were presented.

The ability to precondition Newton-type iteration methods with Picard linearized matrix falls under the category of multilevel preconditioning. This idea is at the core of the proposed MFNK framework wherein consistent actions of a Jacobian matrix on a vector are obtained through finite-difference approximations and inexpensive physics-based linearizations open up the possibility to make use of existing legacy code algorithms on top of powerful and scalable Algebraic preconditioners.

The current research implements all of these algorithms with help of some external software, to accurately couple multi-physics models in a computationally efficient unified code system.

# Chapter 4

# A Non-linear Multi-physics Coupled Code System

'The function of good software is to make the complex appear to be simple.'
– Grady Booch

The methods for spatio-temporal discretization of different physics and the methods for tackling the non-linear system of coupled equations arising from the discretization were introduced in Section. (3). Here, we describe the implementation of the MFNK framework, from a software perspective.

Software engineering of a coupled multi-physics code involves several considerations in the design and implementation of the interaction between different parts of a code. Even though the numerics laid out in Section. (3) have a well defined structure regarding coupling multi-physics models tightly, without careful planning in the software design, even loosely coupled physics using the OS paradigm can be quite complicated to implement. Hence, utilizing the different numerics and physics models strongly depends on creating a software framework that is flexible, extendable and follows a plug-in architecture that can evolve as new or better methodologies for coupling multi-physics components are devised.

Some of the software requirements for a coupled multi-physics code framework include:

1. To re-use existing libraries to minimize development time, and to base the framework on already well verified discretization and non-linear/linear solver libraries. This thought stems from the basic Object-Oriented (OO) philosophy in avoiding code replication and modularizing implementation to accelerate development and testing phases.

2. To provide flexible data containers and physics objects to facilitate and simplify the evaluation of the non-linear residuals for different physics components.

3. To be able to use coarse grain physics models for rapid prototyping, testing and verification and the functionality to interchangeably use higher fidelity physics models to describe the physical phenomena in a straightforward fashion through common API contracts, as and when required by problem constraints.

4. To be able to use within the same architecture, different kinds of multi-physics coupling strategies with minimal changes from an end-user perspective. For instance, using OS with simultaneous or staggered updates, or using OS with Picard iterations to converge the coupling between physics, or employing MFNK tight coupling approaches side-by-side without changing how the non-linear residual describing the discretized PDEs is evaluated.

5. To have the flexibility to add different types of preconditioners, both Algebraic and physics-based, for each of the different physics component models and the option to choose how they are applied to reduce the total cost of linear iterations.

6. To use of recent advances in computer engineering for state-of-the-art multi-core, multi-processor parallel shared memory architectures that can significantly reduce run times for simulation of a physical phenomena.

7. To make the coupled physics code system independent from any specific spatio-temporal discretization. This involves the usage of different spatial discretization with any temporal discretization allowing the possibility to verify the implementation of the same equation system through more than one use-case.

The philosophy behind the software framework for multi-physics applications is to "solve tightly coupled phenomena using a loosely coupled software methodology". The loosely coupled architecture is primarily made possible by requiring a software contract or a defined set of methods to be implemented. This is often called the API (Application Programming Interface) and needs to be defined clearly to allow future extensions.

In order to verify the numerical algorithms proposed in the current work, the need for a new code system was inevitable. Efforts to address this has led to the development of the KARMA framework (**K**(c)ode for **A**nalysis of **R**eactor and other **M**ulti-physics **A**pplications).

## 4.1 KARMA

KARMA is a fully implicit, non-linearly coupled multi-physics eigenvalue and transient analysis test-bed code written completely in C++ programming language. Its primary intended application is to analyze and model coupled problems for nuclear reactor applications although it is not only limited to this family of problems. KARMA makes extensive use of the advanced OO concepts such as abstraction, encapsulation and inheritance, to create loosely coupled objects that allow seamless integration of new physics and numerics models.

The plug-in architecture employed in KARMA makes it straightforward to modify/add any number of coupled physics components. It also serves as a framework to conduct experiments on code architectures and software design for the next generation of consistent coupled multiphysics codes. The framework can be used to seamlessly integrate such physics models with consistent numerical algorithms that were introduced for non-linear PDE systems. In creating such a flexible framework, one of the prime concerns is the ability to achieve high levels of efficiency while still maintaining the ease of development, testing and maintenance. Careful planning of the computational domain has led to a decision to use well tested linear algebra data-structures and methods in order to reduce the overhead in re-implementing these standard algorithms, thereby eliminating the possibility of introducing errors in these basic building blocks for the numerical algorithms proposed in the current work. This also follows closely the OO principles and code re-use whenever possible, thereby preserving man-years of effort pertaining to code verification.

The requirements enumerated earlier are at the core of the design of the

KARMA framework. These abilities in a multi-physics physics code framework are considered representative of current and future trends in solving coupled problems. Similar motivations have also led to the recent development of other coupled multi-physics codes like MOOSE [67].

KARMA is built on top of the state-of-the-art scientific library PETSc, the Portable, Extensible Toolkit for Scientific computation [68] from Argonne National Laboratory (ANL), for fast, scalable and robust data-structures and solvers. It provides tools for the parallel (and serial) numerical solution of PDEs that require solving large-scale, sparse non-linear systems of equations. It includes non-linear and linear equation solvers that employ a variety of Newton-type methods with line-search techniques and Krylov subspace methods. It also offers several parallel vector formats and sparse matrix formats, including compressed row, block compressed row, and block diagonal storage. The primary advantage of using PETSc is that well tested black-box methods and codes that can tackle non-linear systems arising from discretization of Parabolic/Hyperbolic system of equations are obtained implicitly by just linking with with the library. Also, usage of several different kinds of home-grown and external Algebraic preconditioners are obtained by interfacing KARMA with PETSc. Since PETSc is designed to facilitate extensibility, users can incorporate customized solvers and data structures when using the package. PETSc also provides an interface to several external software packages, including Matlab, PVODE, and SPAI and is fully usable from C and C++. Due to the advanced design, users can create complete application programs for the parallel solution of non-linear PDEs without writing much explicit message-passing code themselves. Parallel vectors and sparse matrices can be easily and efficiently assembled through the mechanisms provided by PETSc. Furthermore, PETSc enables a great deal of runtime control for the user without any additional coding cost. The runtime options include control over the choice of solvers, preconditioners and problem parameters as well as the generation of performance logs. Since KARMA uses `PETSc`, all programs using the framework benefit from these ubiquitous options to control the program at a fine-grained level. For instance, options can be specified whether to run a program using a completely matrix-free approach with '-snes_mf', where the action of the Jacobian is found through Eq. (3.76) and no preconditioner is used to solve the coupled system. This approach can lead to large number of Krylov iterations and hence adversely affect the CPU time. Alternately, if an approximation to the Jacobian is available, then a tightly coupled solution procedure can be used with the option '-snes_mf_operator', where the action of Jacobian is again through Eq. (3.76) but the preconditioner matrix is created using the approximate Jacobian matrix representation which is usually some form of linearization about the last Newton iteration (physics-based preconditioner). Additional options using '-pc_type' can be specified for the mode of solving the preconditioner itself which can either be an Algebraic variation (ILU, ICC, AMG) or using a much lower fidelity representation of the Jacobian matrix. Note that any level of recursion in the level of preconditioning, depending on the problem, can be implemented using such a MFNK technique based framework.

Efficient spatial discretizations using cG and dG FE methods along with FD and FV methods can be implemented for each of the physics PDEs. KARMA currently uses the general FEM library, `libMesh` [69]. It is written in C++ and provides support for first and second order Lagrange, arbitrary order $C^0$ hierarchic, $C^1$ continuous and discontinuous finite element bases. `libMesh` also facilitates writing dimension-independent code assembly of the non-linear residual and the Jacobian matrix for each of the physics component, which greatly simplifies the verification process for complex non-linear problems. `libMesh` has interfaces to the parallel vector, matrices, linear algebra data-structures provided by `PETSc`, and hence reduces the overhead to write distributed algorithms that are capable of utilizing the features inherently provided by `PETSc`.

KARMA also supports several different input and output formats that are convenient to generate the correct geometry, assign material region attributes and specify boundary markers. For convenience, in the current work, `Gmsh` [70] is used as the primary mesh generator. `Gmsh` uses the popular Delaunay mesh generators namely `Triangle` in 2-d and `TetGen` in 3-d. The parallel decomposition of the mesh can be performed using `ParMETIS` [71] to minimize the net communication time for a given geometry.

The chosen output format for writing out the solution fields is the `VTK` format [72] which is supported by several visualization packages.

Optionally, KARMA can also be linked with a suite of eigensolvers exposed through the `SLEPc` library [73] that is based on `PETSc`. These state-of-art eigensolvers can handle symmetric and unsymmetric generalized eigenvalue problems that arise from the discretization of the elliptic and hyperbolic systems. For instance, the eigenvalue problem to find the fundamental mode in nuclear reactor design calculations is typically solved using the traditional Power Iteration method but we can also employ one of the eigensolvers provided by `SLEPc` e.g., use the more efficient Krylov subspace methods. The application of these solvers will be discussed in Section. (6).

Several other utility codes can also be optionally used to deal with XML and CSV input/output formats. For instance, TinyXML [74], a small C++ library that can handle reading, manipulation and writing of XML data with very little memory overhead can be used for specifying input options through a file which is forwarded to PETSc. And CSVParser can be employed as a parser to read and write Comma Separated Values (CSV) to aid in reading data from spreadsheets like Excel or data exported from MATLAB (*csvwrite*, *csvread*).

A schematic diagram showing these different parts of the KARMA framework and their interaction with the above mentioned packages is shown in Fig. 4.1.

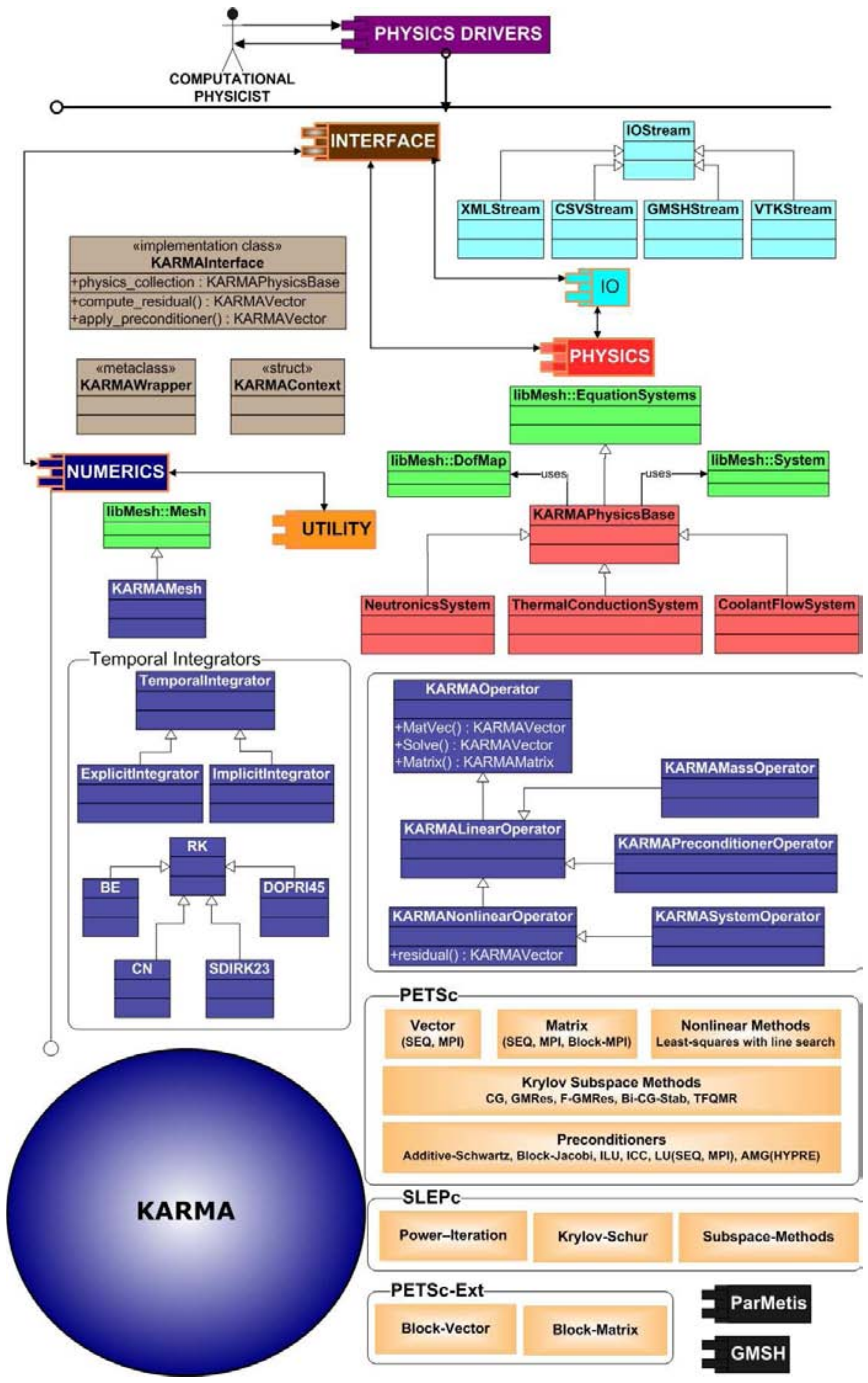


Figure 4.1: Schematic Diagram of KARMA Framework

## 4.2 Modules in KARMA Framework

The KARMA framework has four modules that are responsible for the implementation of the numerical schemes discussed in Section. (3). Brief details regarding each of these modules is given below.

1. `INTERFACE`: This module is the heart of KARMA and is responsible for managing the different physics components and applying the numerical discretizations seamlessly to the coupled non-linear problem. Apart from maintaining a uniform interface to all the physics, it serves as the primary rendezvous point for all user interactions to obtain the coupled non-linear residual or the approximate Jacobian matrix. It also provides 'C' wrappers that serve as function pointers in order to interface with the `PETSc` and `SLEPc` solvers for the non-linear/linear and eigenvalue solvers respectively.

2. `PHYSICS`: This module contains all the physics descriptions, including the non-linear residual, the approximate Jacobian matrix and preconditioners, if any, that are spatially discretized forms of the corresponding physics PDE. All the different Physics objects derive from `KARMAPhysicsBase` that specifies the required methods that need to be implemented by all physics. Since this contract is known a-priori, a generic code to solve the physics can be written in the `INTERFACE`, making use of this polymorphic behavior for generating non-linear residuals and preconditioners.

3. `NUMERICS`: This module comprises of the necessary spatial and temporal discretization objects that are used by the `PHYSICS` module to provide the discretized form of the PDE. The use of `libMesh` in the

current work eliminates additional overhead for spatial discretization. All necessary definitions of temporal integration methods in the form of a Butcher Tableaux are available along with generic integrators for ERK, DIRK and FIRK methods with adaptive time-stepping capability. This module also contains all the necessary higher level wrappers necessary to use the non-linear, linear, eigenvalue solvers and preconditioners provided by `PETSc` and `SLEPc` libraries. These higher level objects make use of the wrappers provided by `INTERFACE` internally and hence the API provided by these objects remain the same, immaterial of whether, say, FGMRes or CG, is used as a solver.

4. `IO`: As the name of the module suggests, it contains all the necessary interfaces to the parsers and data writers to read/write the mesh format (msh), CSV, XML, and VTK for input data processing and output data manipulation in a generic fashion.

## 4.3 Solving a Non-linear Coupled Elliptic Problem

In this section, a step by step example of creating and solving a coupled elliptic problem involving two physics components $\text{Phy}_1$ and $\text{Phy}_2$ is provided below.

### 4.3.1 Adding a New Physics

A new physics model can be added in a straightforward manner by just deriving from the `KARMAPhysicsBase` class and implementing primarily three operators that are essential to solve any physics component. These operators are given below.

1. `SystemOperator`: This operator implements the steady-state non-linear residual definition, which is essentially the spatially discretized PDE representing the physics evaluated at a given time $t$. This operator may also optionally provide an approximate Jacobian matrix, if it is easy to form. This matrix can be invoked and used during a solve with the '-snes_mf_operator' option. The implementation of this operator completes the description of the SS form of the problem.

2. `MassOperator`: This operator represents the mass matrix or a lumped version of it, resulting from the spatial discretization of the time derivative term for the Implicit Differential Equation Eq. (3.36). Note that this operator can either be time dependent itself (property dependent mass matrix or mesh changes with time) or be static, in which case it is only necessary to compute it once.

3. `PreconditionerOperator`: Any physics object can contain an array of preconditioner operators. These can be viewed as multi-level preconditioners where one could use $P_0$ to precondition the approximate Jacobian matrix $J$ and $P_1$ to precondition the solve for $P_0$, and so on. In practice, this sort of recursion might not be very efficient unless care is taken, for each physics, to resolve the stiff components first and systematically reduce the modes that are responsible for the high condition number of the true Jacobian matrix.

Once a physics system implements these three operators, the framework has all the necessary information to solve the system. All the material properties are provided through a problem context as function pointers and hence facilitate the use of arbitrary user-specified properties based on table lookups or correlations.

## 4.3.2 Writing a Non-linear Residual Function

The non-linear residual function that is part of the `SystemOperator` is at the heart of any physics description since it represents the discretized PDE for the physics model. Based on a sample implementation using `libMesh` library, a snippet C++ code is given for a single element residual assembly.

Let $e$ be the finite element under consideration that is a subset of the discrete mesh $\Gamma_h$. The local residual contribution can be computed based on the family of basis functions used to discretize the solution field, the points and location of the quadrature for element integration, and the degrees of freedom for the local solution unknowns. For a diffusion-reaction physics system, the different components of the residual are obtained using Code Snippet 4.1.

```
for (unsigned int iqp=0; iqp<qrule.n_points(); iqp++)
{
  qp = quadrature_point[iqp] ;

  qp_solution = 0. ;
  for (unsigned int i=0; i<phi.size(); i++)
    qp_solution += phi[i][iqp] * solution(dof[i]) ;

  // evaluate the properties at quadrature point with
      qp_solution
  diffusion_coefficient = properties.Diffusion(qp, time,
      qp_solution) ;
  reaction_coefficient = properties.Reaction(qp, time,
      qp_solution) ;
  source_term = properties.Source(qp, time, qp_solution) ;

  for (unsigned int i=0; i<phi.size(); i++)
  {
    Se(i) += JxW[iqp] * source_term * phi[i][iqp] ;

    for (unsigned int j=0; j<phi.size(); j++)
      Ae(i,j) += JxW[iqp] * (
          diffusion_coefficient * dphi[i][iqp] * dphi[j][iqp]
                +
          reaction_coefficient * phi[i][iqp] * phi[j][iqp]
          );
  }
}
```

Listing 4.1: Element Residual Components

where `JxW` is the Jacobian for the element transformation multiplied by the quadrature weight and the diffusion, reaction and source terms are computed at every quadrature point, based on the non-linear solution at those points. The variational form of the diffusion-reaction problem yields the stiffness and mass matrices, which are stored in `Ae`, while the source term is assembled and stored in the load vector `Se`.

With these computed contributions, the local SS residual is simply obtained by performing a local assembly followed by multiplication with the local solution dofs, as shown in Code Snippet 4.2.

```
for (unsigned int i=0; i<phi.size(); i++)
{
  Re(i) = Se(i) ;
  for (unsigned int j=0; j<phi.size(); j++)
    Re(i) -= Ae(i,j)*solution(dof[i]) ;
}
```

Listing 4.2: Residual Computation

It is obvious that all of the above steps for a single element are independent of the next element and hence provide a great deal of inherent parallelism. Also, since the basis functions and quadrature weights are calculated based on the problem's dimension, the residual contribution code is dimension-independent.

If the above physics was coupled to a solution from another physics, it is then necessary to compute projection of the coupled physics solution on to the physical quadrature points used in the element assembly. This is greatly simplified if the coupled physics components use the same mesh since the projection operator is its own interpolant but, in the case of multi-mesh scenarios Section. (3.1.3), the $L_2$ projection of the solution with increased quadrature points will be necessary in order to reduce the spatial coupling errors.

It is important to note that none of the above code snippets mandate any specific discretization method to be used for the physics. A FD or FV method could have been implemented as well instead of the FE method shown above, as long as the non-linear PDE is discretized accurately.

### 4.3.3 Obtaining Coupled Global Residuals

Once the individual physics components are computed, the driver code invokes the residual function in the `KARMAInterface` object which then calls the non-linear residual routines that are part of `SystemOperator` for each physics, as illustrated in Code Snippet 4.3.

```
// distribute the provided solution vector to each physics
synchronize_physics_solution(X) ;

// loop over all the physics and compute the residuals
// and assemble it in the interface residual
for (; pos != end; ++pos)
{
  // Get a reference to the current physics system using the
      iterator
  KARMAPhysicsBase* physics = pos->second ;

  KARMAVector& phy_residual = physics->get_residual() ;
  KARMAVector& phy_solution = physics->get_local_solution() ;

  // call the residual function for the physics
  physics->system_operator->residual (phy_solution,
      phy_residual);
}

// assemble the computed residual to the output vector
synchronize_interface_residual(R) ;
```

Listing 4.3: Coupled Multi-physics Non-linear Residual Computation

The two synchronization routines namely `synchronize_physics_solution` and `synchronize_interface_residual` merely copy data back and forth between the physics object and the interface that maintains a global vector (for all physics). Hence, these residuals could be computed in a 'black-box' code and the KARMA framework could be utilized to compute high-order coupled temporal solution, provided that the coupled physics components are treated consistently (as modeled in the PDE).

### 4.3.4 Summary

In order to compute the non-linear solution for the coupled problem, only the non-linear residual vector is necessary. However, due to the inefficiency in this solution procedure, a linearized form of Jacobian matrix can be computed, with a similar implementation as 4.3.2 based on the nature of the physics.

Temporal discretization can be performed with any one of the ERK, DIRK or FIRK methods (using constant time-stepping or adaptive time-stepping controllers) available as part of KARMA package. With the SS non-linear residual $\mathbf{f}$, Jacobian matrix $\mathbf{J}$ and the `MassOperator`, Eq. (3.43) can be used to compute the time evolution of the solution.

A pseudo-code for the calling stack from the driver to solve the coupled multi-physics problem is shown below. This is general and can be extended easily to several physics components:

1. Create discrete meshes for the physics: $\Omega_1$ and $\Omega_2$
2. Create physics objects: $Phy_1$ and $Phy_2$
3. Add coupled physics reference to one another
4. Intialize `KARMAInterface` and all physics objects $Phy_i \quad \forall i = 1, 2$
5. Set initial temporal solution
6. Invoke the TemporalIntegrator to compute final step solution
    - initialize and allocate intermediate stage vectors

- loop while time $\leq$ final_time
- solve the coupled non-linear problem at each stage
  - use finite difference approximation for action of Jacobian on a vector using Eq. (3.76)
  - if option -snes_mf_operator is used, the approximate global Jacobian is built based on the Jacobian matrix of the individual physics by either ignoring the coupling terms in a Block-Jacobi fashion Eq. (3.72) or variations of this operator with additional coupled terms in order to resolve more stiff components. This problem-dependent preconditioner formulation can be quite effective to reduce the number of FGMRes iterations.
  - if additional preconditioners are available for each physics, apply them to the linear solves in the MFNK solution procedure
  - perform non-linear Newton or Picard iteration until convergence
- if adaptive, compute new time step
- if requested, write solution fields to VTK file
- end

7. Write final solution fields to VTK file

Since command-line options are provided by KARMA to choose the type of coupling strategy (loose (OS) or tight (MFNK)), comparisons on the performance of each coupling strategy can be analyzed without changing any of the user code implemented. This enables computation of sensitivity of a numerical method wrt the coupling strategy for multi-physics problems.

## 4.4 Closing Remarks

KARMA is designed with extensibility in mind and provides a flexible API to couple an arbitrary physics modules together. These modules are written either as part of the library itself or provided through an external code. Since different kinds of coupling methods can be tested under one code system, KARMA serves as a very valuable test-bed code to gain intuition on the optimal strategy for a particular multi-physics simulation. Adaptive techniques in both space and time have already been tested for few non-linear physics and these preliminary results show the feasibility of extending these ideas to multi-physics coupled problems.

# Chapter 5

# Results

'What most experimenters take for granted before they begin their experiments is infinitely more interesting than any results to which their experiments lead.'
– Norbert Wiener

In previous sections, the numerical methods for coupled multi-physics problems and the code framework implemented based on these methods to solve the non-linear system of equations arising from the discretization of physics models have been given. First, the verification of the methods and the code is necessary, in order to understand the efficiency of these multi-physics coupling methods and to stress the need for tightly coupled solution methods. Then, the efficacy of these tightly coupled schemes will be analyzed for some problems that have multiple time scales.

The results section is organized as follows: First, each of the physics models are tested for spatial and temporal consistency (convergence order); Next, verification studies are performed for a conjugate heat-transfer model using semi-analytical techniques that will be discussed in subsequent sections. For problems with widely varying time scales, an efficacy (efficiency in terms of computational time versus accuracy of the solution fields) study is performed to analyze the computational gain due to tightly coupled methods. The coupled treatment of neutronics/heat-conduction physics is simulated next with problems that verify the implementation and quantify the uncertainty propagation due to errors in cross-section data. In the same setting, a stiff transient benchmark problem for the coupled neutronics/heat-conduction physics is used to present the advantages in terms of accuracy and stability of using MFNK tight coupling methods in contrast to traditional OS schemes.

## 5.1 Solution Verification

Typically, the accuracy and convergence of numerical models are established by using simplified problems for which analytical solutions are available. For time-dependent coupled multiphysics problems, analytical solutions are more difficult to obtain and we need to resort to (1) the Method of Manufactured Solutions (MMS) [75] (2) semi-analytical methods [76] and (3) formal convergence order studies to check behavior of discretization errors. In the current work, MMS problems are used extensively to analyze and prove code correctness. The basic philosophy behind MMS is as follows: an exact solution $U_{ref}$, with enough smoothness in space and time, is chosen a-priori and substituted into the continuous form of the PDE to obtain a suitable forcing function (i.e., right-hand-side of the PDE) that is then employed in the numerical simulation. Therefore, the numerically obtained values can be compared to the exact ones, providing a measure of the error. Since the discretization of the source terms are performed consistently in both space and time, the discrete solution can be driven towards the exact solution as the mesh and time step sizes are reduced. This procedure can be applied also for non-linear and coupled physics problems and is an useful tool for verification purposes. The second leg of the code verification is performed by using well established benchmarks involving the physics component that need be tested.

The global error in a numerical solution $U_{num}$ is usually measured in the $L_2$ norm,

$$\|Error\|_2 = \|U_{ref} - U_{num}\|_2 = \sqrt[2]{\frac{1}{|\Omega|}\int_\Omega \left(U_{ref}(\mathbf{r}) - U_{num}(\mathbf{r})\right)^2 d\mathbf{r}}. \qquad (5.1)$$

A method is of order $p_s$ in space and order $p_t$ in time if the error varies as $O(\Delta r^{p_s})$ and $O(\Delta t^{p_t})$, respectively. The order of convergence in space is measured by computing the global error in a transient simulation for which the spatial mesh is successively refined. A small temporal grid is necessary to ensure that the temporal error is small enough so that the error observed is due to the spatial grid only. A similar procedure holds for the temporal error calculation, where the spatial mesh is fine enough so that the spatial errors do not pollute global error and successive simulations are performed with uniformly refined time step sizes. The measurement of the error and the comparison of the space/time accuracy orders obtained with expected convergence rates prove that the code implementation is consistent with the mathematics and that the numerical solution converges to the true solution of the PDE.

The MMS forcing functions used in the simulations are given in the Appendix for different coupled physics scenarios. For further reading regarding the MMS, refer to [75, 77].

## 5.2 Verification of Individual Physics Models

In order to verify the implementation of the physics and the numerics models, a step-by-step process is adopted. For verifying a coupled multi-physics code, we first verify the single physics models, some of which are can be non-linear by themselves. To perform this step, analytical solution methods and/or the MMS techniques explained earlier are utilized.

### 5.2.1 Nonlinear Scalar Parabolic Problem

A general scalar non-linear Parabolic PDE is considered and spatial/temporal discretization of the equations are performed using cG with Lagrange basis functions and ERK, IRK methods respectively. The verification studies for problems involving such systems is shown here.

#### 5.2.1.1 Verification

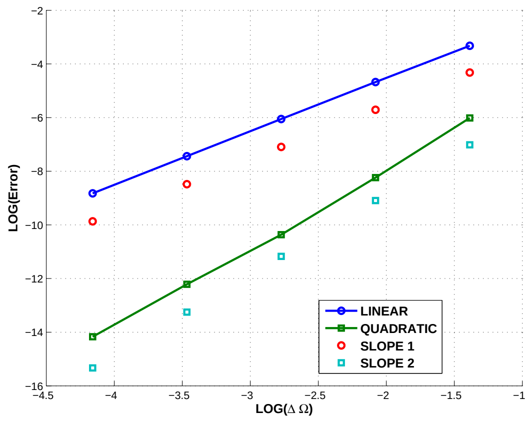
Using the MMS technique, a dimension-independent non-linear heat conduction problem is modeled. The thermal conductivity is chosen to be the non-linear functional $k(T) = T^2$. The exact solution is taken to be

$$T(\vec{r}, t) = \tanh(t) \prod_{i=1}^{dim} \sin(\pi r_i), \tag{5.2}$$

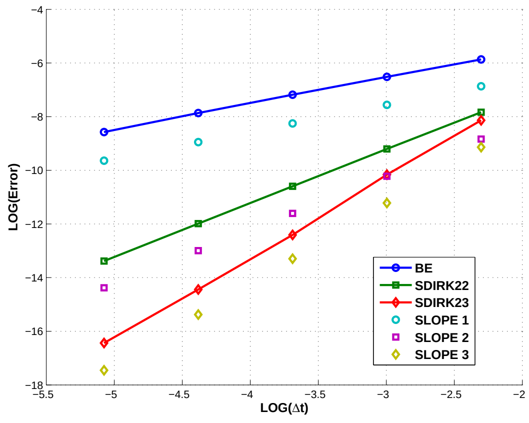
where $\vec{r} = \{x, y, z\}$ and $dim = 1, 2, 3$. Because the exact solution is known, the spatial and temporal error discretization can be readily quantified using Eq. (5.1).

This test is presented with linear and quadratic Lagrange basis functions for spatial discretization, in a two-dimensional domain. As theoretically expected, second and third order spatial convergence rates are observed for linear and quadratic Lagrange basis functions respectively, see Fig. 5.1(a).

Using a fine spatial mesh, the temporal order of accuracy for BE1(1), IM2(1) and the SDIRK3(2) schemes are obtained and plotted in Fig. 5.1(b). We can clearly note that the non-linear solution method based on the MFNK framework is high-order accurate in space and time and presents the expected theoretical orders of accuracy based on the space/time discretization.



(a) Spatial Accuracy



(b) Temporal Accuracy

Figure 5.1: Non-linear Heat Conduction Problem: Spatial and Temporal Accuracy

### 5.2.2 Nonlinear Fluid Flow Problem

Next, the discretization of the problem arising from hyperbolic fluid flow equations are verified. We will consider a manufactured solution to verify accuracy and discretization errors.After these verification studies, the efficiency of the ICE preconditioner for the conservative equations is shown.

#### 5.2.2.1 Verification

Using the MMS, profiles for the state variables are assumed and corresponding forcing functions for the continuity, momentum and energy equations are derived. The exact solutions assumed for density, velocity and total energy [28] are shown below

$$\rho = \rho_{\min} + (\rho_{\max} - \rho_{\min}) \operatorname{sech}\left(\frac{x - \varpi t}{\delta}\right) \tag{5.3}$$

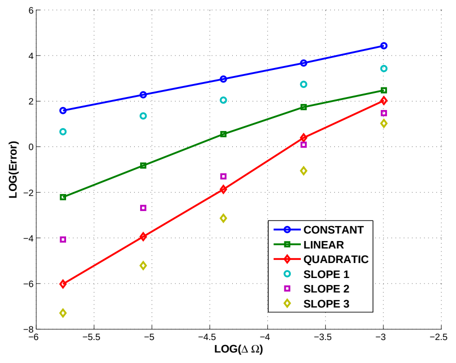
$$v = v_{\min} + (v_{\max} - v_{\min}) \operatorname{sech}\left(\frac{x - \varpi t}{\delta}\right) \tag{5.4}$$

$$E = E_{\min} + (E_{\max} - E_{\min}) \tanh\left(\frac{x - \varpi t}{\delta}\right). \tag{5.5}$$

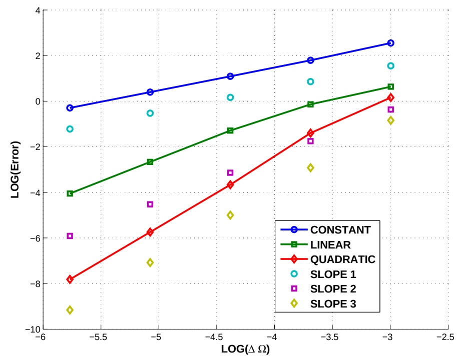
The closure relation for the pressure equation was chosen to be the ideal gas equation represented by

$$P = \rho e(\gamma - 1), \tag{5.6}$$

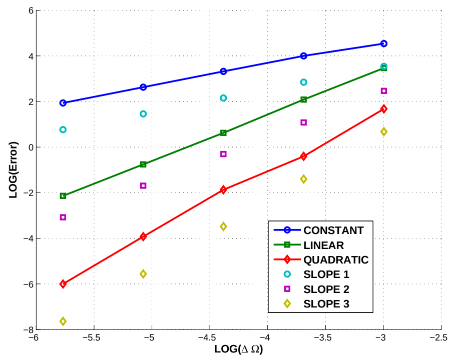
where $\gamma = \frac{C_p}{C_v}$. Using these exact solutions and the equation of state, the source terms for each conservation equation were obtained and the spatial and temporal convergence orders were computed.



(a) Spatial accuracy in $\rho$



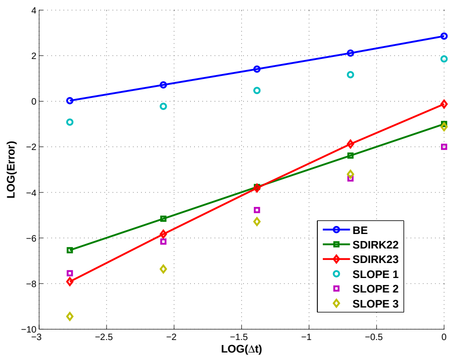
(b) Spatial accuracy in $\rho U$



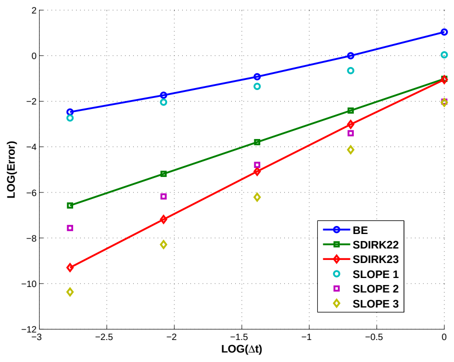
(c) Spatial accuracy in $\rho E$

Figure 5.2: Fluid Flow Problem: Spatial Accuracy

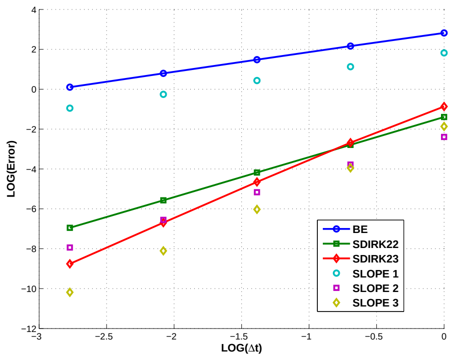
The 1-d Navier-Stokes equations are solved in a fully implicit manner using the MFNK method. The spatial order of accuracy was measured for different polynomial orders of Legendre basis functions with dG FEM discretization. The obtained accuracy orders are as expected theoretically and prove that the spatial treatment of the 1-d fluid equations are consistent. With a fine spatial mesh and second order dG finite elements with Legendre basis functions, the temporal order of convergence for density solution for a final time $t = 2$ sec. was obtained and plotted on Fig. 5.2. The convergence plot for different methods is shown in Fig. 5.3.



(a) Temporal accuracy in $\rho$



(b) Temporal accuracy in $\rho U$



(c) Temporal accuracy in $\rho E$

Figure 5.3: Fluid Flow Problem: Temporal Accuracy

## 5.3 Verification of Coupled Physics Models

Now that individual physics models have been verified, the next step involves verifying the numerical solution for coupled problems based on a combination of these single-physics models. The results obtained from these studies is shown in the following subsections.

### 5.3.1 Coupled Conjugate Heat Transfer Example

Coupled conjugate heat-transfer problems are common in all thermal-hydraulic calculations. These coupled phenomena are primarily boundary condition based and hence lead to locally strong coupling effects. First, the MMS is applied to verify the implementation of such a coupled system of equations and then the efficacy of using different kinds of coupling schemes for these problems is presented.

#### 5.3.1.1 Verification

To verify conjugate heat transfer between a conducting solid and a fluid, we employed a manufactured solution. A Matlab script was written to obtain the forcing functions based on the following assumptions. The script is given in Appendix A.1.

The flow is assumed to be uni-directional along the $z-$direction with a constant inlet mass flux. The Blasius correlation in the turbulent regime is used to compute the friction factor $f_w$:

$$f_w = \frac{0.3164}{Re^{0.25}}. \tag{5.7}$$

where $Re$ is the Reynolds number of the fluid.

Thermal conduction of energy in the bulk fluid is assumed to be absent and energy is added to the fluid only at the wall surface (fluid-solid interface). The solid heat conduction model is basically a $x - z$ slab in Cartesian co-ordinate system which convects the heat generated to the bulk fluid. The exact solutions for the fuel and fluid temperatures, $T_{fuel}$ and $T_c$, respectively, are given below.

$$\begin{aligned}T_{fuel}(x,z,t) =& r_F\Big(1+\tanh(C_{tf}\,t)\Big)\left(\tfrac{1}{2}+\sin\left(\tfrac{\pi\,x}{2L_X}\right)\right)\\ &\left(1+\tanh\left(\tfrac{2w}{3}-\tfrac{wz}{L_Z}\right)\right)+T_{f0} \qquad (5.8)\\ T_c(z,t) =& r_T\Big(1+\tanh\left(C_{tc}\,t\right)\Big)\left(a+b\tanh\left(-cw+\tfrac{wz}{L_Z}\right)\right) \qquad (5.9)\\ \rho(z,t) =& \rho_c+f\left(1-\tfrac{T(z,t)}{T_{c0}}\right)+g\sqrt{1-\tfrac{T(z,t)}{T_{c0}}}. \qquad (5.10)\end{aligned}$$

The internal energy and total energies are given by

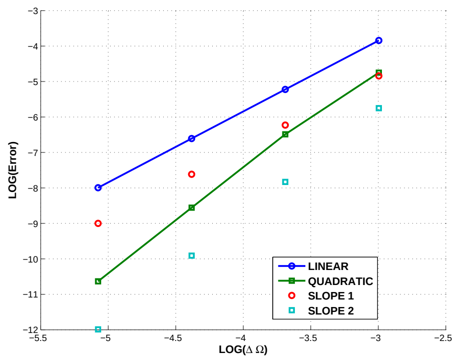
$$\begin{aligned}\rho e =& \rho C_v T_c \qquad (5.11)\\ \rho E =& \rho e+\tfrac{1}{2}\tfrac{G^2}{\rho}, \qquad (5.12)\end{aligned}$$

where $r_T, T_{f0}, C_{tf}, r_F, T_{c0}, C_{tc}, w, a, b, c, f, g$ are parameters to control the magnitude and time scales of the solution. $C_v$ is the specific heat at constant volume, $G$ is the mass flow rate.

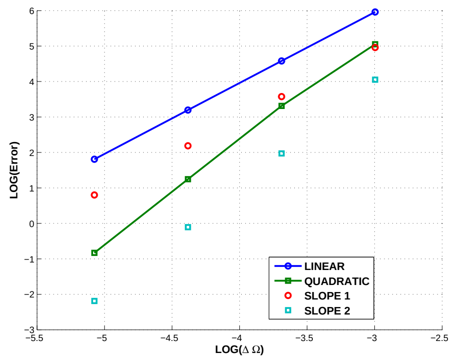
The Equation of State employed to close the system of equations is a linearized relation, dependent on density and temperature.

$$P = P_0+\alpha(\rho-\rho_0)+\beta\left(T_c-T_0\right), \qquad (5.13)$$

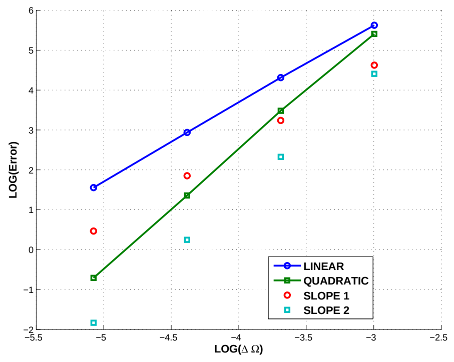
where $\alpha, \beta$ are the linearization constants. Note that $\alpha$ is actually related to the speed of sound in the flowing medium and provides a simple way to manually change the Mach number in the calculations, apart from varying the mass flux $G$ itself.



(a) Spatial accuracy in $\rho$



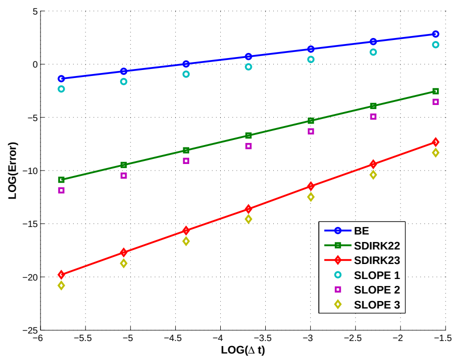
(b) Spatial accuracy in $\rho E$



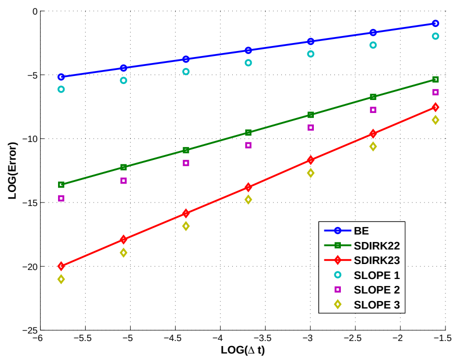
(c) Spatial accuracy in Fuel Temperature

Figure 5.4: Conjugate Heat Transfer Problem: Spatial Accuracy

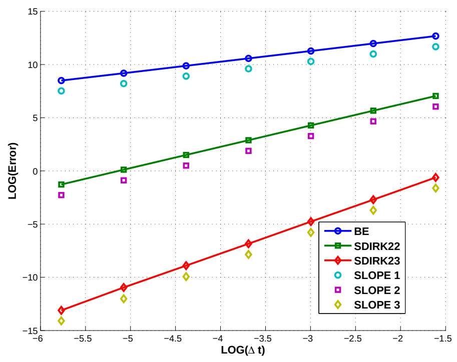
Based on this manufactured solution, the forcing functions have been generated and a convergence order study has been carried out for various levels of spatial and temporal discretizations. The numerical solutions approach the true solutions as the spatial and temporal meshes are refined, as expected. The convergence results, shown in Fig. 5.4 and Fig. 5.5, prove that the implementation of the physics is verified and demonstrate that higher-order accuracy can be obtained using the MFNK technique.



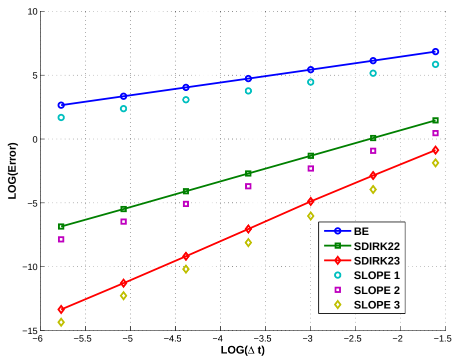
(a) Temporal accuracy in $\rho$



(b) Temporal accuracy in $\rho U$



(c) Temporal accuracy in $\rho E$



(d) Temporal accuracy in Fuel Temperature

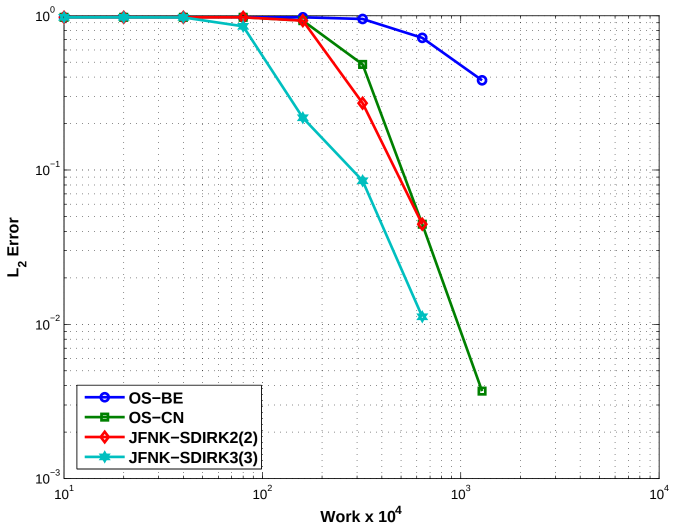
Figure 5.5: Conjugate Heat Transfer Problem: Temporal Accuracy

#### 5.3.1.2 Efficacy

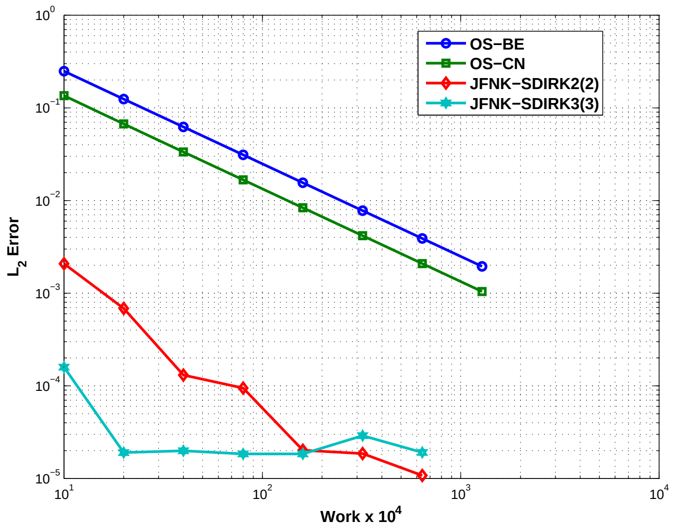
The conjugate heat transfer problem introduced earlier with MMS is used again to test the efficacy of the coupling methods. By varying the temporal scales of the exact solutions i.e., change the constants as $C_{tf} = 1000C_{tc}$. This forces the evolution of the fuel temperature to occur at a much faster rate than the transient in the fluid equations. In order to resolve this stiffness, we use the traditional OS coupling strategy and the MFNK method with the $L-$stable SDIRK2(2), SDIRK3(3) methods.

The accuracy results obtained for a given spatial mesh is plotted against the total computational time for the different coupling techniques in Fig. 5.6.

From the efficacy plot for the total energy field Fig. 5.6(b), it is evident that for any given user-specified tolerance, the total cost for the solution procedure using OS strategy with first order BE method is higher by several orders of magnitude. But since the temporal evolution of fuel temperature occurs at a much faster rate, resolving the physical time scales required reducing the time step sizes for all the methods. Once the asymptotic region was reached, the higher order MFNK methods quickly reduce the error in the solution. For the temperature variable, for the same computational cost, the loosely coupled scheme is less accurate than MFNK based tightly coupled solution by two orders of magnitude. This result indicates that for stiff problems, once the dynamical physical scales are resolved, the higher order temporal accuracy in the tightly coupled schemes do provide considerable computational gain. It is also interesting to note that using a second order temporal method (CN) with OS coupling saves computational due to the fact that there is only 1 implicit stage while the SDIRK2(2) method has two stages per time step. These results indicate that it is possible for OS schemes with high-order temporal methods to be feasible in terms of efficiency, as compared to tightly coupled methods, when the time step sizes are well below dynamical time scales of the individual physics.



(a) Efficacy for Temperature



(b) Efficacy for Total energy

Figure 5.6: Conjugate Heat Transfer Problem: Efficacy Study

### 5.3.2 Coupled Neutronics-Thermal Conduction Example

Neutronics and Heat conduction physics solution fields are strongly coupled to each other through temperature dependent cross-section values and heat generation from fission in the fuel. Since the coupling between these two physics models is strong, it is necessary to test the effectiveness of the coupling methodology to accurately resolve the varying time and length scales. In this section, the coupled neutronics and heat conduction physics models are verified by means of MMS. In the first case, the same spatial discretization are applied to both physics ; in the second case, different (non-embedded) spatial meshes are used to quantify the effect of non-conforming meshes.

#### 5.3.2.1 Verification: Identical Spatial Meshes

Making use of the MMS technique once again, a test problem is used to verify convergence of the method to exact solutions. Since the coupling between neutronics and conduction is non-linear and due to the fact that the conduction physics is non-linear by itself, obtaining manufactured solutions can become quite intricate. A Matlab script has been written to obtain the forcing functions based on the following assumptions.

Two delayed neutron precursors groups are considered along with two energy groups and one scalar temperature field. Hence, the total number of solution fields is five. We give below the exact solution for the fields for the

2D test case:

$$\phi_1(x,y,t) = A_\phi\big(1+\tanh(r_\phi t)\big)\sin(\pi x)\sin(\pi y)\,xy \tag{5.14}$$

$$\phi_2(x,y,t) = \phi_1(x,y,t) \times \tfrac{\Sigma_{s,1\to 2}}{(\Sigma_{rem,2}+D_2 B_g^2)} \tag{5.15}$$

$$C_i(x,y,t) = C_i(x,y,0)e^{-\lambda_i t} + \int_0^t ds\; e^{\lambda_i(s-t)} \sum_{g=1}^{g=2} \beta_{i,g}\nu\Sigma_{f,g}\phi_g(x,y,s) \tag{5.16}$$

$$T(x,y,t) = A_T\big(1+\tanh(r_T t)\big)\sin(\pi x)\sin(\pi y), \tag{5.17}$$

where $B_g^2 = (\frac{\pi}{L_X})^2 + (\frac{\pi}{L_Y})^2$ is the geometric buckling term, $L_X$ and $L_Y$ are the domain sizes in the $x$ and $y$ dimensions. $A_\phi$, $r_\phi$, $A_T$ and $r_T$ are constant parameters. Using the exact solutions for the fluxes, the exact solutions for the precursors, $C_i$, can easily be obtained and are given by

$$C_i(x,y,0) = \frac{1}{\lambda_i}\sum_{g=1}^{g=2} \beta_{i,g}\nu\Sigma_{f,g}\phi_g(x,y,0), \quad i = 1,2. \tag{5.18}$$

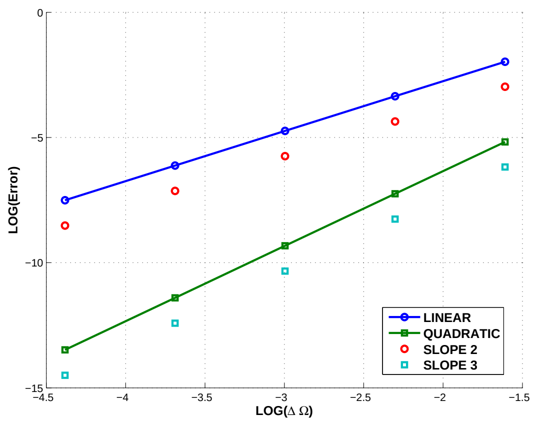
Doppler feedback is accounted in the neutronics model through the removal cross-section of group 1:

$$\Sigma_{rem,1}(T) = \Sigma_{rem,1}(T_0) + \left.\frac{\partial \Sigma_{rem}}{\partial\sqrt{T}}\right|_0 (\sqrt{T} - \sqrt{T_0}). \tag{5.19}$$

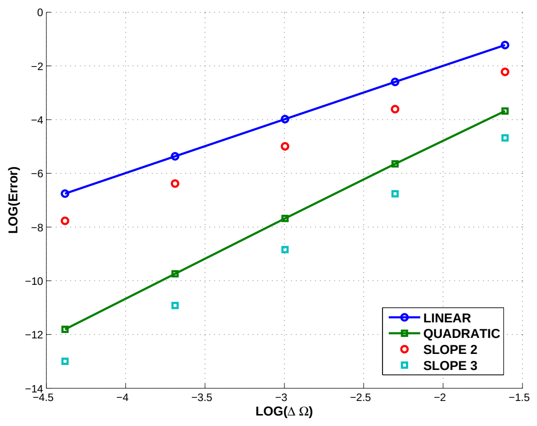
The following equation is employed to described the temperature-dependent conductivity:

$$k(T) = k_0 + \left.\frac{\partial k}{\partial T}\right|_0 (T - T_0). \tag{5.20}$$

The Matlab script used to obtained the forcing functions is given in Appendix A.2. Note that here only the fast energy group's removal cross-section is affected by temperature variations. Additional temperature dependencies can be included in the removal and fission cross-sections for both groups using the provided script. With the knowledge of the exact solution profiles and the corresponding forcing functions, a space/time convergence study is carried out. The convergence plots for this example are shown in Fig. 5.7 and Fig. 5.8.



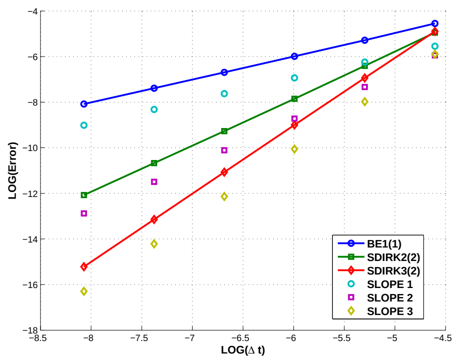
(a) Spatial accuracy - Fast Flux



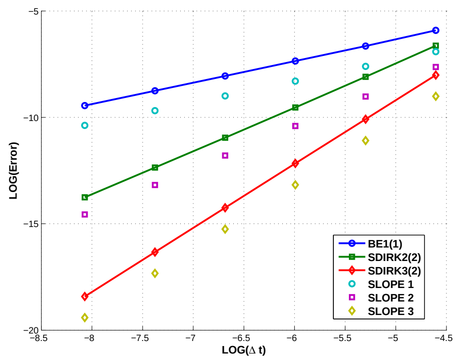
(b) Spatial accuracy - Fuel Temperature

Figure 5.7: Coupled Neutronics/Heat Conduction Problem: Spatial Accuracy

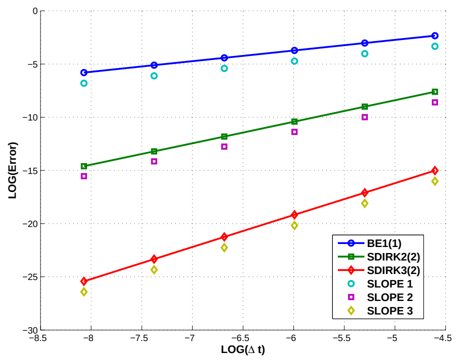
It is clear from the results that the solution fields for all physics components are high-order accurate in space and time and agree well with theoretical convergence rates. By varying the coupling coefficient $\frac{\partial \Sigma_{rem}}{\partial \sqrt{T}}$ in Eq. (5.19), and other free parameters such as $r_\phi, r_T, 1/v_1, 1/v_2$, stiffer transients were created and convergence to the true solution was observed still. The higher order temporal schemes are efficient and the stiffly-accurate SDIRK schemes prove to be superior than traditional low-order BE scheme.



(a) Temporal accuracy - Fast Flux



(b) Temporal accuracy - Thermal Flux

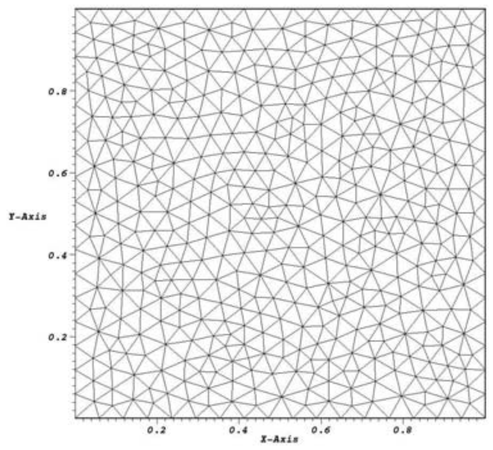


(c) Temporal accuracy - Fuel Temperature

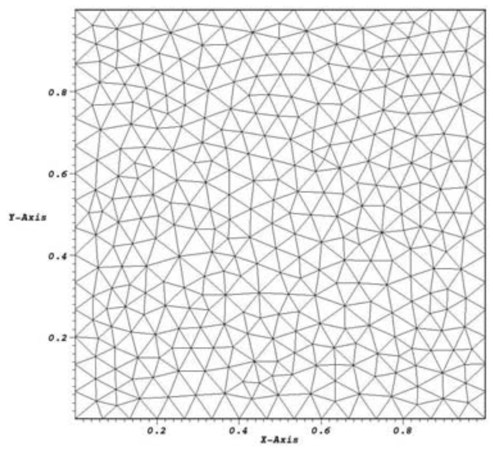
Figure 5.8: Coupled Neutronics/Heat Conduction Problem: Temporal Accuracy

#### 5.3.2.2 Verification: Non-embedded Spatial Meshes

Multi-physics applications often require that different physics components employ different spatial meshes, which resolve the spatial scales occurring in the specific physics. Even though a complete study of this question is not covered in this project and requires extensive work all by itself, an example is provided here for the coupled neutronics/heat conduction physics utilizing different spatial meshes; see Fig. 5.9. Note that the meshes are *not* embedded (when meshes are embedded, projection and interpolation operators are simpler to define).



(a) Neutronics Mesh

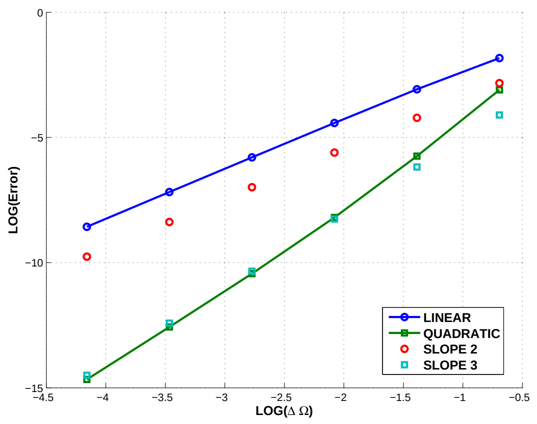


(b) Conduction Mesh

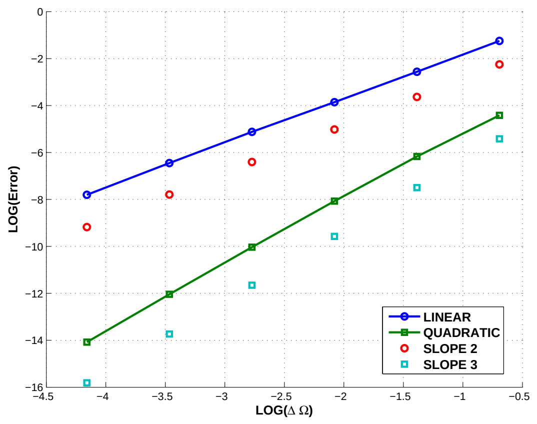
Figure 5.9: Coupled Neutronics/Heat Conduction Problem: Non-conforming Meshes

As detailed in Section 3.1.3, the spatial coupling error that may occur due to inadequate projection/interpolation in between source/target meshes can be avoided by employing a 'high-enough' numerical quadrature in the finite element setting. It is important to verify that mitigation of the interpolation errors is possible by using high order quadratures. In this example, the same test problem with MMS from Section 5.3.2.1 is considered but different spatial meshes are employed.

The exact solution profiles for the two physics components show that their spatial scales vary differently. Without the use of high-order quadratures, the interpolation error start to dominate and corrupt the numerical solution from reaching asymptotic spatial convergence orders. In order to eliminate this, high-order quadrature rules are employed to overkill the spatial errors due to solution interpolation and projection on a target mesh, as indicated in Section 3.1.3. Fig. 5.10 shows that high-order spatial convergence is recovered, as expected, for the MMS by using more quadrature points to resolve the variation in the coupled solution. This is only a preliminary study with multi-mesh for multi-physics software verification and further analysis is needed to test the efficiency of the technique employed in the project.



(a) Spatial accuracy - Fast Flux



(b) Spatial accuracy - Fuel Temperature

Figure 5.10: Coupled Neutronics/Heat Conduction Problem with Nonconforming Meshes: Spatial Accuracy

#### 5.3.2.3 Efficacy

A rod ejection benchmark problem from the ANL problem book [78] (Identification: 14-A1) is considered here to verify the coupling method described in this project. This is a super-prompt critical transient with adiabatic heating and Doppler feedback for a thermal reactor model. In contrast with the previous MMS test cases, the time scales change quite drastically over the transient and it is expected that higher order time stepping schemes would outperform low-order schemes. In this simulation, the total power level changes by over 10 orders of magnitude. The adiabatic heating assumption is also numerically challenging: since no dissipative terms are present, the errors introduced in computing the temperature field can grow undamped with inconsistent numerical treatment, thereby requiring accurate coupling and time stepping methods.

Based on the described geometry, the domain was discretized using second-order triangular Lagrange finite element, as shown in Fig. 5.11. The initial steady state eigenvalue obtained is $k_{\mathit{eff}} = 0.99636\,41531\,80855$ for the given spatial mesh and it closely matches the reference value. The transient calculation was performed using various combinations of coupling and temporal discretization schemes: OS-BE1(1), OS-IM2(1), MFNK-BE1(1), MFNK-IM2(1), MFNK-SDIRK3(2). The reference solution was calculated with a fine spatial mesh and a 3–rd order $L-$stable SDIRK scheme SDIRK3(3) with $\Delta t = 0.001$ secs. The results obtained from these the reference simulation for the fast and thermal flux distributions are shown in Fig. 5.12 and the transient evolution in Fig. 5.13. It is interesting to note that the profile peak power solution does not match the benchmark solution but considering that low order discretizations with coarse spatial meshes were used in the benchmark, it is possible that the true converged solution is the one given here. Ryosuke et al. [79] noticed similar temporal evolution of the solution fields and the code to code verification of KARMA with the PRONGHORN software gives confidence in the results obtained here.

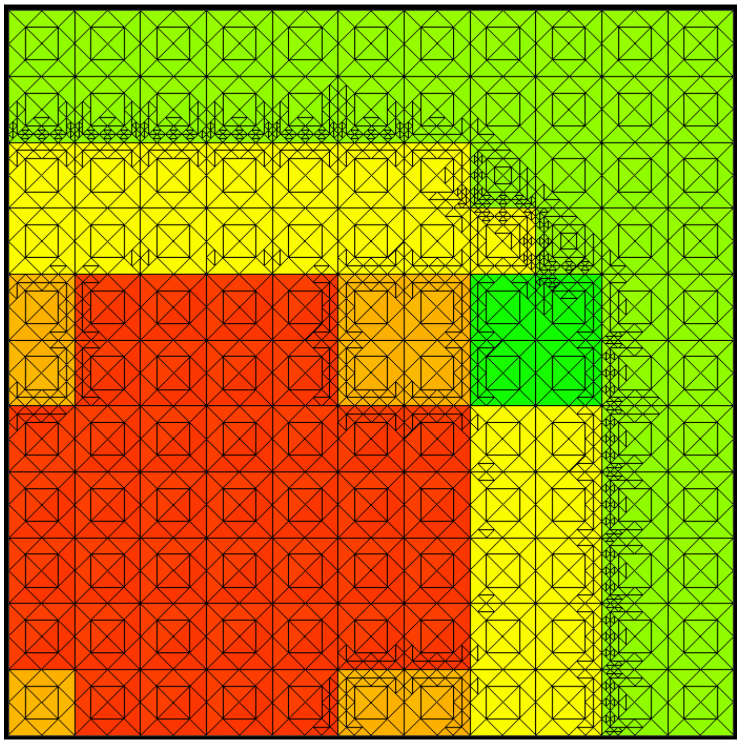


Figure 5.11: Super-prompt Critical Benchmark Problem: Geometry and Computational Mesh

All the tightly coupled simulations (MFNK-based solves) use $\Delta t = 0.004$ secs. The OS simulations had to employ a much smaller time step of $\Delta t = 10^{-4}$ secs in order to converge; indeed, the explicit linearization of the OS schemes led the solution to diverge with large time steps. Fig. 5.14 shows a closer view of the power peak due to the rod ejection transient.

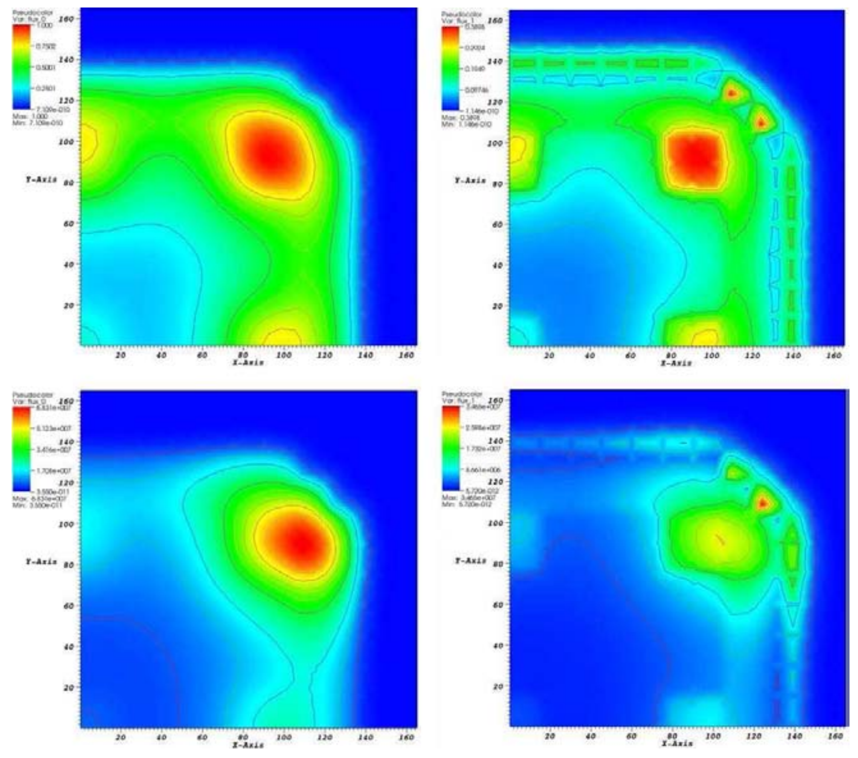


Figure 5.12: Super-prompt Critical Benchmark Problem: Fast(left) and Thermal(right) Flux Profiles at $t = 0$ (top) and $t = 3$ secs (bottom)

It is clear that the first-order BE1(1) scheme, even with tight coupling, i.e., MFNK-BE1(1), wrongly computes the magnitude and time of occurrence in the power peak. This trend with BE1(1) can also be seen with the OS coupled strategy, even with the much smaller time step used for OS schemes. The higher order schemes, IM2(1) and SDIRK3(2), approach the reference solution consistently as time step sizes are reduced and predict the occurrence of the peak power accurately.

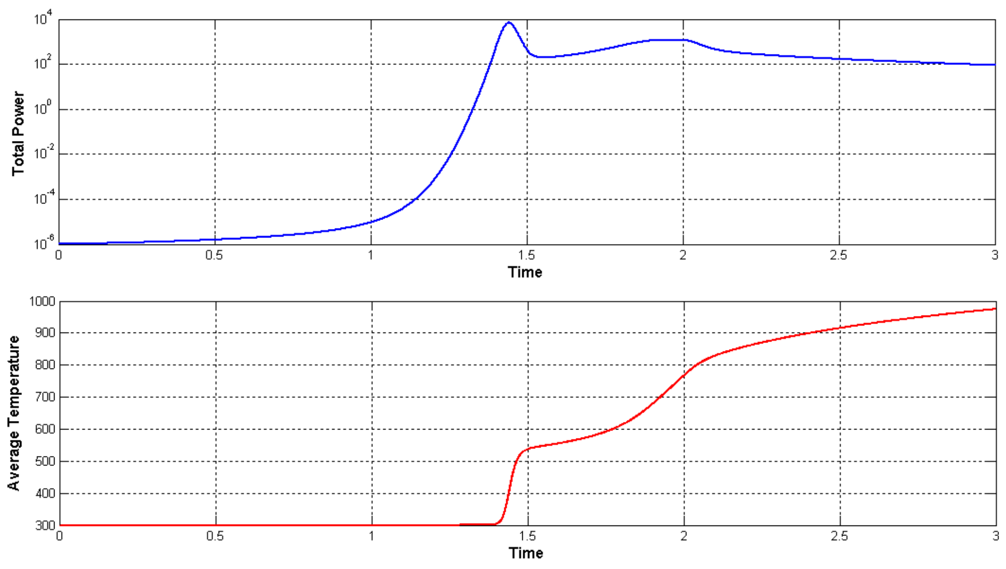


Figure 5.13: Super-prompt Critical Benchmark Problem: Power Transient

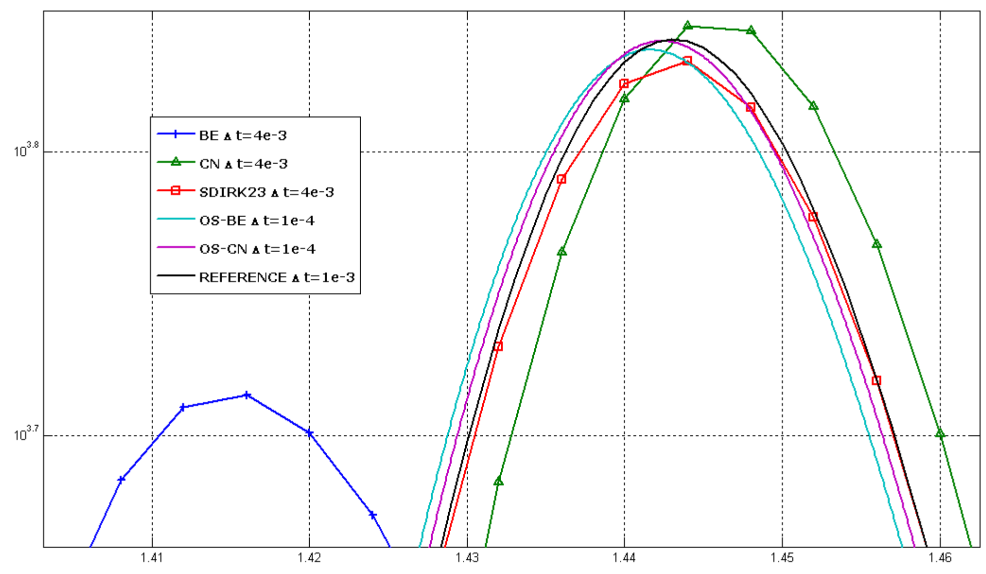


Figure 5.14: Super-prompt Critical Benchmark Problem: Power Transient with Different Numerical Schemes

These results emphasize the usefulness of employing higher-order temporal integration schemes to predict the behavior of stiff transients that can occur in nuclear reactors. The usage of a tight coupling strategy allows to use larger time steps than OS schemes, due to the fully resolved non-linearities at each time step, which can be used in combination with adaptive time-stepping techniques to reduce the total computational times.

## 5.4 Uncertainty Quantification for Multi-physics Problems

Uncertainty quantification is necessary even for single-physics simulations and it is imperative to perform this analyses for multi-physics problems in order to gain better understanding on weak versus strong coupling between the physics. Aleatory and Epistemic uncertainty contributions can occur from a variety of sources. They can stem from physical model errors, data uncertainty errors, numerical errors due to inaccurate geometry descriptions, resolution of spatial and temporal scales, numerical errors from coupling methodologies to name a few, apart from Epistemic errors that are inherent to the system being modeled. In the current work, uncertainty propagation in the solution specifically due to material properties which are usually obtained through experiments and/or higher fidelity models with necessary approximations and homogenizations, is analyzed. Since the properties used in numerical simulations depend heavily on closure relations and empirical correlations, the propagation of error and uncertainty in the solution fields need to be ascertained. It is also important to note that the uncertainities introduced in the solution obtained from simulations are irreducible, even with a fine resolution of spatial or temporal scales when there is an uncertainty in the data.

Uncertainty quantification in the solution due to these variables following certain distributions can then be performed using several mature methods. One popular and less intrusive option is the Generalized Polynomial Chaos (GPC) method [80]. GPC is in essence a decomposition method where an uncertain output variable of interest $Z$ is expressed in terms of a basis $\Phi$ of a stochastic space consisting of random variables $\zeta = \{\zeta_1 \ldots, \zeta_n\}$,

$$Z(t,\zeta) = \sum_n Z_n(t)\Phi_n(\zeta) \tag{5.21}$$

where $\zeta$ can be described by either a Gaussian, Gamma or Uniform distribution. For most problems, where the true distribution is not known, a Gaussian distribution can be chosen to represent the random variable. In this case, Hermite polynomials are used as basis functions $\Phi_i$ and are given by

$$H_n(x) = \Phi_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}(e^{-x^2}) \tag{5.22}$$

where $n$ is the order of the Hermite polynomial and $x$ is the random variable of interest. GPC methods have exponential convergence as the order of the polynomials $p$ is increased. Further information can be obtained from references [81, 82].

The application of a GPC method of order $p$, GPC($p$), to the coupled neutronics-conduction model problem is studied here. The introduction of the dependency on the randomness of the properties adds another dimension to the system of equations. Hence, the coupled physics solution fields are also functions of the stochastic variation in these random input variables. In other words, Eq. (5.21) can be rewritten in terms of random variables $\zeta$ as

$$Z(t,\zeta) = C_0 + \sum_j C_j(t)H_1(\zeta_j) + \sum_j C_j(t)\prod_k H_2(\zeta_k) + \ldots \tag{5.23}$$

where $C_j$ are the coefficients of the expansion and can be computed making use of the orthogonality property of Hermite polynomials. In Eq. (5.23), the coefficients $C_j$ are essentially the moments of the $j^{th}$ order Hermite polynomial. Note that the zero'th moment $C_0$ represents the mean value of the physics solution $Z$ when all the random variables $\zeta$ are at their mean values. In order to obtain the coefficients, Eq. (5.23) can be multiplied by a Hermite polynomial of certain order and integrated from $-\infty$ to $\infty$, and using orthogonality relation, we have

$$C_i = \int_{-\infty}^{\infty} d\zeta e^{-\frac{\sum_j \zeta_j^2}{2}} Z(t,\zeta)H_n(\zeta) \tag{5.24}$$

Note that in Eq. (5.24), the solution $Z(t,\zeta)$ is obtained from code results using a random sample for the input variable based on its distribution. The actual integral in Eq. (5.24) is computed using numerical quadrature rules, leading to non-intrusive uncertainty quantification methodology. Gauss-Hermite quadrature rules are used for the numerical integration and the order of the quadrature chosen should be high enough to compute the integral exactly. For instance, to compute the GPC(2) coefficients, a Gauss-Hermite quadrature rule that can exactly integrate a fourth order polynomial (three point quadrature rule, $qp$=3) is required. Hence the total number of simulations necessary, after optimizations to eliminate recurrent cross-term effects, is in the order of $n * p * qp$ where $n$ is the number of input random variables. Once the required simulations are computed, the coefficients $C_i$ can be computed and the output variables of interest can be found using the basis Eq. (5.23).

Alternately, Monte-Carlo based sampling procedure are efficient at providing accurate statistical information regarding the uncertainty in the models. But since Monte-Carlo based sampling methods require large number of samples (or model runs), it is prohibitive to directly perform the simulation for every sample. We can instead use the basis obtained through GPC($p$) polynomial and then use this as a surrogate model to obtain the required statistical data for each output variable of interest, along with their respective Probability Distribution Functions (PDF). Using such surrogate modeling methodology designed specifically for propagating uncertainty from model inputs to model outputs, a quantitative analysis can be performed and better confidence in simulation predictions can be gained.

A super prompt-critical transient problem is considered here, along the lines of the previous ANL benchmark. This 2-d, two energy-group, two delayed-group neutron diffusion problem with Doppler feedback has a variation in the thermal absorption cross-section defined by the relation:

$$\Sigma_{rem,2}(T) = \Sigma_{rem,2}(T_0) + \left.\frac{\partial \Sigma_{rem}}{\partial \sqrt{T}}\right|_0 (\sqrt{T} - \sqrt{T_0}) + \left.\frac{\partial \Sigma_{rem}}{\partial T}\right|_0 (T - T_0). \quad (5.25)$$

The geometry of the problem essentially consists of a rodded fuel element surrounded by unrodded fuel which is encompassed in a lattice of reflector assemblies. This is shown in Fig. 5.15. The material properties are obtained from MATPRO correlations for $UO_2$ fuel with 95% theoretical density [83]. The transient is initiated by changing the thermal fission cross-section for the rodded fuel element through a ramp duration of 0.1 seconds. The solution profiles for the fast and thermal fluxes at SS conditions is shown in Fig. 5.16.

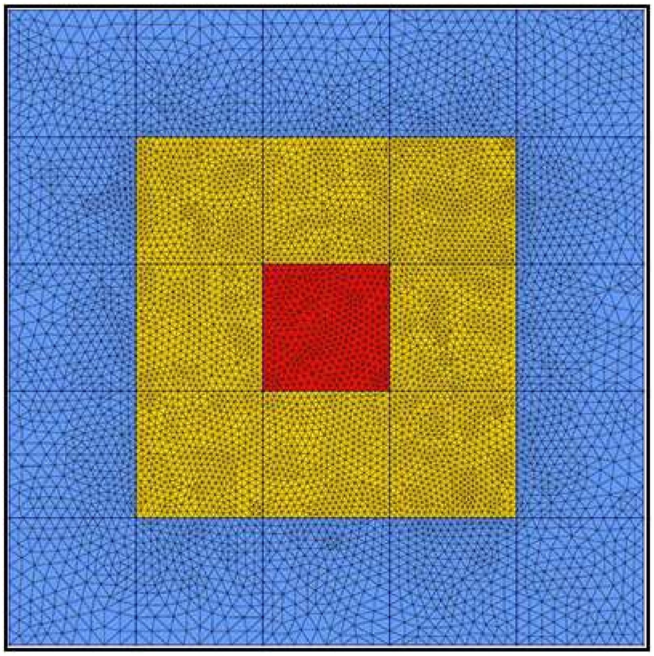


Figure 5.15: Uncertainty Quantification Test Problem: Geometry and Computational Mesh

Four random input variables are chosen with the following Gaussian distributions:

$$\Sigma_{rem,2}(T_0)|_{rod+fuel} = N(0.2231, 5\%) \tag{5.26}$$

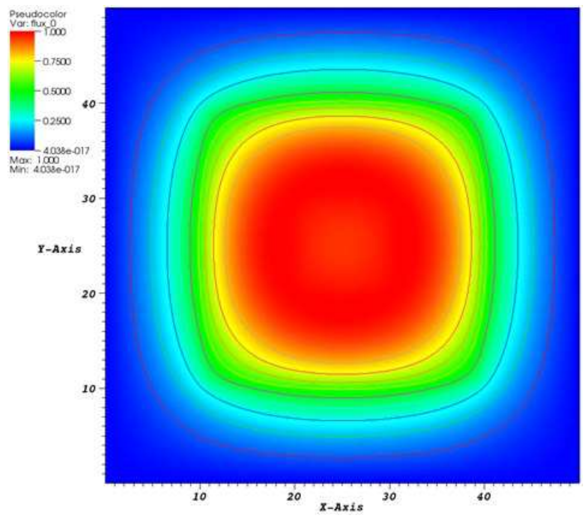
$$\Sigma_{rem,2}(T_0)|_{fuel} = N(0.09194, 5\%) \tag{5.27}$$

$$\frac{\partial \Sigma_{rem}}{\partial \sqrt{T}} = N(0.002043, 10\%) \tag{5.28}$$

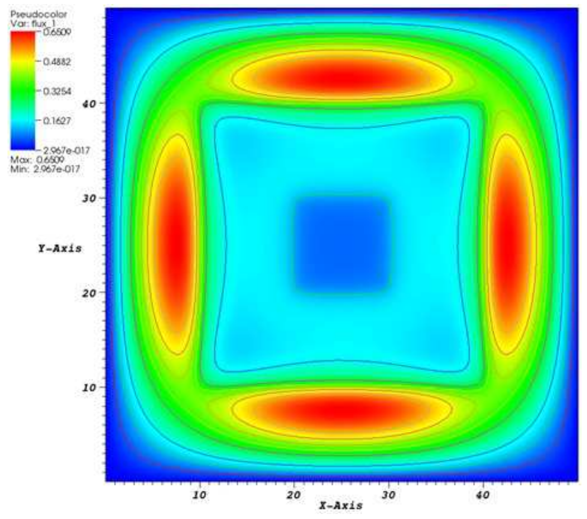
$$\frac{\partial \Sigma_{rem}}{\partial T} = N(5 \times 10^{-6}, 10\%) \tag{5.29}$$

and the uncertainty in three specific solution fields namely, $k_{eff}$, the total maximum power in the transient, and the peak fuel temperature, are sought after.

A second order GPC expansion ($p = 2$) is considered along with the possible cross-terms to account for propagation of errors due to complex interplay of different physics. In order to efficiently compute the analytical integrals arising from the GPC basis expansions to compute the coefficients, a high-order Gauss-Hermite quadrature rule is used to eliminate any integration errors.



(a) Fast flux



(b) Thermal flux

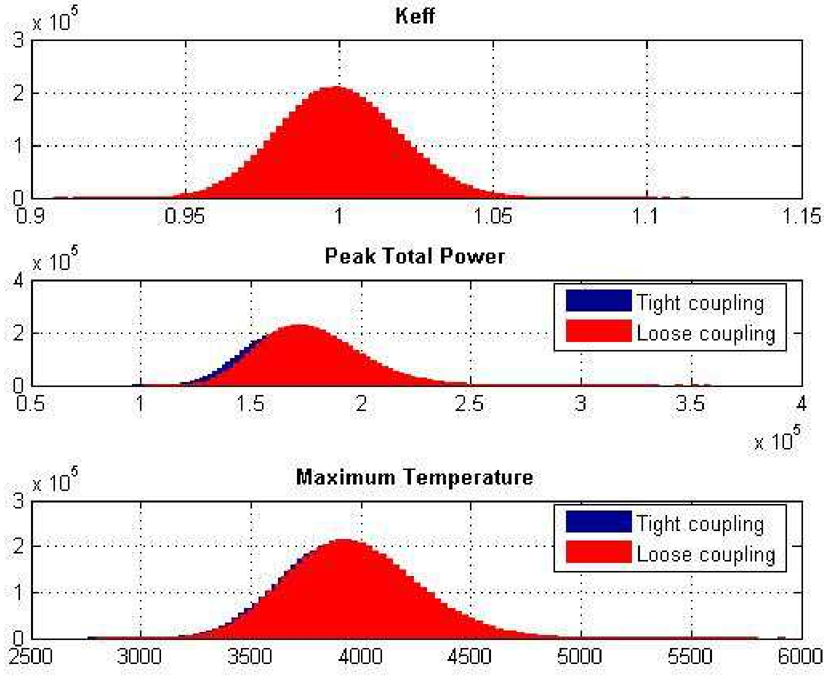
Figure 5.16: Uncertainty Quantification Test Problem: Initial Solution Profiles

Once a reference spatial mesh is determined, the mean solution fields with all input random variables at their probable mean values are found. Then, each random variable is perturbed by a factor of $10^{-6}$ sequentially and the corresponding change in the solution fields is found. This measure of the global sensitivity of the solution fields w.r.t these random variables is given in Table 5.1. Since the current test problem is at hot zero power conditions (initial fuel temperature = 300 C), the eigenvalue $k_{eff}$ is not dependent on the coupling with temperature in the fuel. But all the transient parameters namely, the peak power and maximum fuel temperature, are strongly sensitive to the doppler coefficient $\frac{\partial \Sigma_{rem}}{\partial \sqrt{T}}$. The values themselves in Table 5.1 are specific for the problem considered here and can change depending on the mean, the deviation and the chosen input distribution.

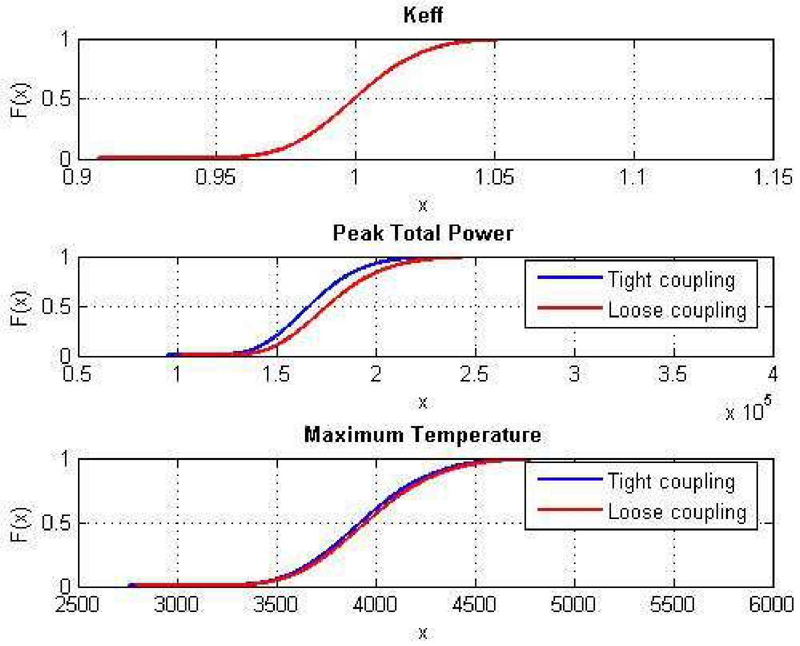
Table 5.1: Relative sensitivity values for the output variables depending on the input random variables

| Output variable | $\Sigma_{rem,2}(T_0)\|_{rod+fuel}$ | $\Sigma_{rem,2}(T_0)\|_{fuel}$ | $\frac{\partial \Sigma_{rem}}{\partial \sqrt{T}}$ | $\frac{\partial \Sigma_{rem}}{\partial T}$ |
|---|---|---|---|---|
| $k_{eff}$ | -0.3173834 | -8.4336155 | 0 | 0 |
| Peak Total Power | -12.156938 | 28.4867 | -551.53479 | -11.8566 |
| Peak Temperature | -6.74126 | 15.06638 | -1.046647E6 | -2.61168E4 |

For the transient simulated here, the stability limit for traditional OS coupling was found to be around $\Delta t = 0.02$ by experiment i.e., the OS strategy needed time step sizes smaller than $\Delta t = 0.02$ to obtain a stable numerical solution. We use both the tightly coupled and OS based loosely coupled methods for the same transient with this stability limit step size. Based on the simulation results, the output variable expansion coefficients were computed and a surrogate model based on the expansion in Hermite polynomial basis was created. This surrogate model was then used with Monte-Carlo sampling techniques with $5 \times 10^6$ samples to obtain the mean and deviation in the output variables given in Table 5.2 and the corresponding Probability/Cumulative Distribution Functions shown in Fig. 5.17 for both OS and tight coupling based solutions.



(a) Probability Distribution Function



(b) Cumulative Distribution Function

Figure 5.17: Uncertainty Quantification Test Problem: Distribution Functions for Output Variables

It is evident from the peak total power distribution plots that the loosely coupled schemes have a different distribution than the true solution distribution (tightly coupled). As the time step size is decreased, this disparity does vanish since the OS linearization for coupled physics terms becomes more accurate. It is possible to use Richardson extrapolation technique to compute the time step independent coefficients $C_j$ to eliminate the corruption of temporal errors in estimating the uncertainty due to input random variables, which provides the new possibility to obtain better prediction of the output uncertainty with OS coupling schemes.

Table 5.2: Mean and standard deviations for output variables

| Coupling | $k_{eff}$ | Peak Total Power | Peak Temperature |
|---|---|---|---|
| OS | N(0.9999867970, 0.019539) | N(177685.35,23178.69) | N(3968.08,295.6) |
| MFNK | N(0.9999867970, 0.019539) | N(167900.83,20936.29) | N(3939.17,290.3) |

Since traditional multi-physics codes use loose coupled schemes, the predicted mean and the standard deviation for the range of input distribution are conservative (mean power and temperature are higher). In reactor design and safety calculations, the margins calculated using these can then restrict the design and reduce the total thermal power output. Even though the problem posed here is not a full reactor core configuration, the results obtained suggest that improvements based on tightly coupled methods can lead to improved margin characterization. Further studies are needed using higher order GPC expansions and different time step sizes in order to reduce the total uncertainty in the output variables and to gain better intuition on uncertainty quantification of multi-physics problems.

# Chapter 6

# Application of KARMA to Alternate Problems

‘If you wish to advance into the infinite, explore the finite in all directions.’
– Johann Wolfgang von Goethe

In this section, the application of the KARMA framework to a different variety of non-linear multi-physics problems, other than those that occur in nuclear reactors transients, is explored. The flexibility of the framework and the versatility of the applications of the implemented code are demonstrated using these problems. First, the application of the Newton’s method for eigenproblems is shown followed by a stiff system of coupled equations occurring in radiation-diffusion problems. The theory and results obtained from these problems are shown here.

## 6.1 Criticality Eigenproblem and Modal Analysis

Large algebraic eigenvalue problems often arise in nuclear engineering applications. In reactor physics, criticality problems typically require the calculation of the fundamental eigenvalue and associated eigenmode to determine cycle lengths, reactivity margins and power distributions. For analyzing the unstable patterns encountered in Boiling Water Reactors (BWR) [84, 85], often modal analysis is performed that requires solutions to large eigenvalue problems. Reactor instabilities are local power variations occurring due to periodic flow oscillations and can sometimes grow undamped. In stability studies, not only the fundamental mode but also higher-order eigenmodes are needed in order to predict core behavior accurately.

Fast and accurate numerical methods to obtain the fundamental mode and to perform modal analysis for reactor instabilities are an ongoing topic of research; see, e.g., [86, 87, 88, 89, 90]. In the current section, we underscore the link between all eigenproblems with constrained non-linear optimization problems and take advantage of this reformulation to use the Newton's method for solving the eigenproblems. This lends the possibility to use the existing MFNK framework introduced earlier in Section. (3).

Now let us consider the eigenproblems encountered in reactor physics based on the Multi-group Neutron Diffusion (MGND) equation given below:

$$\begin{aligned}-\vec{\nabla}\cdot D^g(\vec{r})\vec{\nabla}\Phi^g(\vec{r}) + \Sigma_r^g(\vec{r})\Phi^g(\vec{r}) - \sum_{g'\neq g}\Sigma_s^{g'\to g}(\vec{r})\Phi^{g'}(\vec{r}) \\ = \frac{1}{\lambda}\chi^g\sum_{g'=1}^{G}\nu\Sigma_f^{g'}(\vec{r})\Phi^{g'}(\vec{r}) \qquad \forall g = 1,\ldots,G, \quad \text{for } \vec{r}\in\mathcal{D} \qquad (6.1)\end{aligned}$$

In addition to Eq. (6.1), which is defined in the reactor domain $\Omega$, boundary conditions are supplied on the domain's boundary $\partial\Omega$. The boundary conditions are typically of homogeneous Dirichlet-type (zero flux) or of homogeneous Neumann-type (symmetry lines).

After spatial discretization using cG FEM, the above multigroup equations can be recast in a discrete operator form

$$L\phi_m = \frac{1}{\lambda_m} F\phi_m, \tag{6.2}$$

where $L$ is the multigroup loss operator containing leakage, absorption and scattering terms, $F$ is the multigroup production operator containing the fission terms, $\lambda_m$ is the $m^{th}$ eigenvalue and $\phi_m$ is the $m^{th}$ (multigroup) eigenmode associated with the $\lambda_m$ eigenvalue. The pair $(\lambda_m,\, \phi_m)$ will be denoted hereafter as the $m^{th}$ eigenpair. The block structure of the $L$ discrete operator is

$$L = \begin{bmatrix} L_1 & 0 & \dots & & & & 0 \\ -L_{2,1} & L_2 & 0 & \dots & & & 0 \\ \vdots & \vdots & \ddots & 0 & \dots & & 0 \\ -L_{t,1} & -L_{t,2} & \dots & L_t & -L_{t,t+1} & \dots & -L_{t,G} \\ \vdots & & & & \ddots & \dots & \vdots \\ -L_{G1} & \dots & & & & -L_{G,G-1} & L_G \end{bmatrix} \tag{6.3}$$

where the row index $t$ denotes the first thermal energy group, i.e., the first group for which up-scattering is present. The diagonal blocks $L_i$ represent the discrete form of $-\vec{\nabla}\cdot D^i\vec{\nabla} + \Sigma_r^i$ whereas the off-diagonal blocks $L_{i,j}$ represent down-/up-scattering operator $\Sigma_s^{j\to i}$. Similarly, blocks $F_{i,j}$ are the discrete representation of $\chi^i\nu\Sigma_f^j$. Each block in $L$ and $F$ is a real and sparse matrix of size $n\times n$ with $n$ denoting the number of spatial unknowns. It is also important to note that matrix $L$ is not symmetric due to the non-symmetric scattering processes in between energy groups. The problem dimension $N =$

$n \times G$ is large (for moderately coarse 3D 2-group diffusion calculations, $N$ is of the order of, at least, 50,000 unknowns.) The $m$-th eigenfunction is a multigroup vector: $\phi_m = [\phi_m^1, \ldots, \phi_m^G]^T$, where each component $\phi_m^g$ is a vector of length $n$.

The following ordering of eigenvalues is chosen: $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_{N-1}| \geq |\lambda_N|$, where we have indicated the uniqueness of the largest eigenvalue in neutronics applications [91]. Out of the $N$ eigenmodes, the first (fundamental) mode predicts the critical core configuration (in neutronics, it is customarily to let $\lambda_1 = k_{\mathit{eff}}$, where $k_{\mathit{eff}}$ is the effective multiplication factor) and higher modes dictate the behavior of the system during a transient. Since higher modes decay away in comparatively shorter time periods as the mode index number increases, only a subset of the higher-order modes are needed in practical applications.

Eq. (6.2) can be symbolically transformed by left multiplication with $L^{-1}$ to obtain a standard unsymmetric eigenvalue problem of the form

$$M\phi_n = \lambda_n \phi_n, \tag{6.4}$$

with $M = L^{-1}F$.

The generalized eigenvalue problem of Eq. (6.2) is only reformulated as Eq. (6.4) for notational simplicity and $M$ is never directly computed and stored due to the large memory requirements (i.e., $L^{-1}$ is never computed explicitly.) To simplify notations, the standard eigenvalue problem, Eq. (6.4), will be used hereafter, although it is important to note that each matrix-vector (matvec) product of the form $z = Mx$ requires (i) one matvec operation $y = Fx$ followed by (ii) a linear solve $Lz = y$ for $z$. Traditionally, the solve of the loss operator $L$ is performed by recursively sweeping through the energy groups in a Block-Gauss-Seidel (BGS) fashion until convergence.

Hence, we will consider one such BGS sweep as the primary work unit and performance parameter for the different algorithms tested in computing several eigenmodes.

In most existing reactor analysis codes, the Power Iteration (PI) [92] method has been used to obtain the largest eigenvalue. By using deflation techniques or spectral shifting techniques [92] with the PI method, higher eigenmodes can be obtained, albeit with decreasing accuracy. This is due to the fact that performing deflation very accurately is computationally expensive and any inaccuracy introduced in this process, i.e., in the previous modes, can result in the corruption of the deflated system for higher-modes, thereby making it difficult to recover the true eigenmodes.

Recently, Krylov-subspace iteration methods have been proposed as alternatives to the PI method to compute several dominant eigenmodes, e.g., the subspace projection method with locking of converged eigenvectors [93], the explicit Arnoldi iteration [73, 90], the implicit restarted Arnoldi method (IRAM) [94, 95, 96, 97, 98], the Jacobi-Davidson method [99, 88, 100]. Subspace methods for eigenproblems have been described and reviewed by several authors in the mathematics and numerical analysis communities; see, for instance, [101, 102].

The approach employed in the current work aims to employ the existing framework based on MFNK technique in order to find multiple eigenmodes accurately. This methodology is based on hybridizing the traditional techniques for solving eigenvalue problems with a variant of inexact Newton's method. In order to underscore the link between eigenproblems with optimization problems tackled using Newton's method, consider the following Rayleigh quotient functional (a non-linear functional) for a given Hermitian matrix $A$, defined as:

$$R(\phi) = \frac{\phi^T A\phi}{\phi^T \phi} \qquad \text{for } \phi \neq 0. \tag{6.5}$$

The minimization of the Rayleigh quotient requires satisfying the optimality condition

$$\nabla R(\phi) = 0 = \frac{2}{\phi^T \phi}\left(A\phi - R(\phi)\phi\right). \tag{6.6}$$

This optimality condition is solved using Newton's method

$$[\nabla^2 R(\phi^l)]\delta\phi = -\nabla R(\phi^l) \tag{6.7}$$

$$\phi^{l+1} = \phi^l + \delta\phi, \tag{6.8}$$

where $\nabla^2 R$ is the Hessian matrix of the Rayleigh quotient functional, or equivalently the Jacobian matrix of the optimality condition, Eq. (6.6). These simple lines of algebra may facilitate establishing the relationship between eigenproblems and Newton's method, which will be discussed in section 6.1.2, where we propose a hybrid scheme that combines Newton's method and subspace iteration eigensolvers in order to compute, with high accuracy, a *large* number of eigenmodes for nuclear reactor analysis problems. Furthermore, the non-linear solves of Newton's method will be performed in a matrix-free fashion to avoid computing the possible expensive matrix $\nabla^2 Q$. Finally, recall that matrix $M$ is never formed, and only the action of the linear system on a vector is realized. Hence for improved efficiency, preconditioned versions of the matrix-free Newton's method are applied.

### 6.1.1 Review on Existing Schemes to Compute Multiple Eigenmodes

Various subspace iteration eigensolvers applying a Rayleigh-Ritz projection have become popular over the recent years to determine several dominant eigenmodes in nuclear reactor applications; several references of such work were given in the introduction. In their simplest forms, they generate Krylov subspaces by repeatedly multiplying matrix $M$ with a basis vector, and in that sense can be thought of generalizations of the power iteration method. To reduce the size of the Krylov space needed to compute several eigenvalues near a portion of the eigenspectrum, some appropriate shifting strategies can be employed. Similar shifting strategies can also be used in the simple PI method to obtain eigenvalues with a faster convergence. A brief review of the PI and Krylov-subspace family of methods are given below.

#### 6.1.1.1 Standard Power Iteration

In most reactor design and analysis codes currently in use, some form of modified/accelerated Power Iteration (PI) technique is employed to find the fundamental eigenvalue ($\lambda_1$) and the associated eigenmode. Such a procedure is known to be slowly convergent when the dominance ratio ($\frac{\lambda_2}{\lambda_1}$ ) is close to 1, resulting in a large number of operator applications, requiring many inversions of the multigroup loss operator $L$.

Common schemes to accelerate the power iteration technique are the Chebyshev acceleration [103] and the Wiedlandt shift [92].

#### 6.1.1.2 The Shifted Inverse-Power Iteration

The Shifted Inverse-Power Iteration (SIPI) [101] is a popular and improved variant of the PI method where a guess (shift) of the eigenvalue is known reasonably well.

In the SIPI method,the eigenspectrum is shifted by a constant value $\sigma$ while preserving the corresponding eigenvectors. Hence, when a reasonable guess for the eigenvalue is known a-priori, the SIPI iteration converges to the eigenpair efficiently. Alternately, the Rayleigh quotient can be used as an improved shift at each iteration, leading to a locally quadratic asymptotic convergence rate for unsymmetric problems [104]. It has also been shown by Geltner that a good initial guess of the eigenvector is necessary for local convergence of an eigenmode, particularly when the loss matrix is not positive definite. The Rayleigh shift resembles the Rayleigh quotient shown in Eq. (6.5) and is given by

$$\sigma_i = \frac{\phi_i^T M \phi_i}{\phi_i^T \phi_i}, \tag{6.9}$$

and at each SIPI iteration the following linear system needs to be solved

$$(M - \sigma_i I)\phi_{i+1} = \phi_i. \tag{6.10}$$

It should be noted that, as the iteration converges to the solution eigenpair, the shifted matrix operator $(M - \sigma_i I)$ becomes ill-conditioned and singular. This is one of the primary disadvantages in using the SIPI method.

The linear system $(M - \sigma_i I)$ is usually solved iteratively (inner iterations) to a given linear tolerance. It is possible to set this linear tolerance adaptively based on the norm of the residual of the eigenvalue problem, leading to an inexact-SIPI method, akin to the inexact-Newton method in philosophy, which is computationally less expensive than the exact SIPI variant. The motivation behind this is that the linear solves need not to be extremely accurate, while the eigenvalue residual is still large.

The SIPI procedure is suitable to find one eigenpair. When more than one eigenpair is sought, e.g., when performing modal analysis, additional work needs to be performed to remove the contribution of converged eigenmodes. In the work by Demaziere [90], the traditional PI method is used to calculate several eigenmodes with Wielandt deflation but this procedure involves the necessity for calculating the forward and adjoint eigenvector for each mode. The accuracy of the subsequent modes may be affected if the computed modes have a large residual error that can propagate in the computation of subsequent higher modes and corrupts the deflated linear system (in [90], a tolerance close to machine precision was used).

Alternately, a suitable locking technique through orthogonalization can be performed in order to remove the contribution of an already computed mode from the linear system [105]. In the our implementation of Rayleight quotient iteration , this procedure is utilized for computing several dominant eigenmodes.

#### 6.1.1.3 Krylov Subspace Iteration Methods

The basic philosophy of the PI method is to build a subspace by the repeated action of operator $M$ on a starting vector $v$, yielding a $\ell$-dimensional subspace whose span is $V = \{v, Mv, M^2v, ...M^{\ell-1}v\}$. As the size $\ell$ of the subspace becomes large, the power series spanned by the subspace does not form an appropriate basis to extract the eigenvalue and eigenvector approximations [102]. The columns of this subspace matrix $V$ are not orthogonal. In Krylov subspace iteration, an orthogonal basis is sought and the resulting Krylov matrix $\hat{V}$ provides approximations to the eigenvectors corresponding to $\ell$ dominant eigenvalues of $M$. Some advanced Krylov subspace methods for non-symmetric system of equations include the explicit Arnoldi [73] and Implicitly Restarted Arnoldi [95], the Jacobi-Davidson [99] and Krylov-Schur [73] methods.

The original Arnoldi algorithm [101] is a powerful extension of the subspace iteration in that it builds an orthonormal basis of the Krylov subspace and factorizes the matrix $M$ into an upper Hessenberg matrix. The central idea behind the Arnoldi factorization is to construct eigenpairs of the original matrix from eigenpairs of the factorized, much smaller, Hessenberg matrix. Sorensen [106] improved the Arnoldi method by introducing several shifted QR iterations on the Hessenberg matrix; this reduces the overall number of required matvec operations. Several other variations and optimizations have been added to this Arnoldi method and have been successfully implemented in the library ARPACK [95]. The Implicitly Restarted Arnoldi Method (IRAM) [106] is considered to be one of the state-of-the-art schemes to compute several dominant eigenpairs for large, sparse linear systems. Several researchers [97, 107, 98, 90] have considered the use of an Arnoldi eigensolver for $k$-eigenvalue problems and found it to be a promising alternative to obtain the fundamental mode.

### 6.1.2 Newton Iteration Based Hybrid Algorithm

Instead of approaching the eigenvalue problem with traditional iterative techniques, we recast the eigenproblem as a non-linear problem to be solved by means of Newton's method. Peters and Wilkinson [108] proved that the inverse iterations are equivalent to a Newton's iterative scheme. Saad [101] also considered the suitability of using non-linear Newton type iteration schemes for large symmetric eigenvalue problems and proposed several variations for the definition of the non-linear residual function.

Let us consider the eigenvalue problem in Eq. (6.4) and formulate it as a $(N+1)$-dimensional unconstrained optimization problem (recall that (i) $M = L^{-1}F$ is a matrix of size $N$ by $N$ and that (ii) it is never explicitly formed).

Based on a formulation proposed by Wu et al. [102] for the standard eigenvalue problem, the non-linear residual function can be written as follows

$$\mathcal{F}(y) = \begin{bmatrix} (M - \lambda I)\,\phi \\ -\frac{1}{2}\phi^T\phi + \frac{1}{2} \end{bmatrix} = 0 \tag{6.11}$$

where $\mathcal{F}(y)$ is the non-linear Newton residual vector, of size $N+1$. The first $N$ components of $\mathcal{F}$ represent the linear system for the eigenproblem in Eq. (6.4) and the last component of $\mathcal{F}$ is simply a 2-norm normalization of the eigenvector. The $N+1$ dimensional solution vector $y$ contains the eigenmode and the eigenvalue and is given by

$$y = \begin{bmatrix} \phi \\ \lambda \end{bmatrix} \tag{6.12}$$

The eigenmode and the eigenvalue solution are then obtained using Newton's method

$$\begin{aligned} \delta y^l &= -\left[J(y^l)\right]^{-1}\mathcal{F}(y^l) && (6.13) \\ \text{and } y^{l+1} &= y^l + \delta y^l && (6.14) \end{aligned}$$

where the Jacobian matrix is given by

$$J(y) = \begin{bmatrix} M - \lambda I & -\phi \\ -\phi^T & 0 \end{bmatrix} \tag{6.15}$$

Peters [108] proved that this augmented matrix is non-singular even when $(\lambda, \phi)$ is the true eigenpair being sought. Saad [101] confirmed the proof for the symmetric case and extended it to the non-symmetric matrix case which is considered here.

The choice of the optimization problem given in Eq. (6.11) avoids nearly singular, ill-conditioned Jacobian matrices and hence is convergent locally to an eigenpair without the numerical difficulties (e.g., large condition numbers) often observed in techniques such as the SIPI method near an eigenpair solution [109]. The quadratic convergence of the Newton scheme can be utilized to create a robust eigensolver as long as a suitable initial guess for the eigenmode is available. It is also known that the above Newton-type procedure for eigenvalue problems is equivalent to the Rayleigh Quotient iteration, whose convergence behavior is well understood [102, 109]. Nonetheless, a few points have to be addressed in order to use Newton's method to compute the eigenmodes:

1. the need for an appropriate initial guess close to the desired eigenpair;

2. the cost of computing the Jacobian matrix $J$ and memory requirements for storing it are prohibitive for large problems;

3. the need of efficient preconditioning technique to reduce Krylov subspace required for inner linear solves.

The next paragraphs address these. First, Newton's initial guesses to compute the dominant eigenmodes will be obtained from either a few SIPI iterations or a Krylov subspace iteration technique, with coarse tolerances. The Newton iteration acts as the eigensolver once it has been "bootstrapped" using a standard eigensolver used to provide an initial guess to focus Newton's search eigenspace around one or several desired modes. Second, we will rely on matrix-free approaches to avoid the computation of the Jacobian matrix. Finally, preconditioning techniques will be employed. Details regarding the proposed algorithm are discussed below.

#### 6.1.2.1 Starting the Newton Iteration

It is well known that Newton's method converges quadratically to the solution as long as the initial condition is inside the "ball of convergence". If we are to obtain a certain number of dominant modes, then proper initial guesses must be provided. For our purpose, we are not interested in just any eigenmode and a procedure to enrich the Newton's search directions for the dominant eigenmodes is necessary in order to focus the Newton's search space.

One option is to perform first a few SIPI iterations (with coarse tolerance) to obtain an approximation to the fundamental eigenmode, thus providing a starting vector for Newton's method. We refer to this scheme as the "SIPI-Newton" hybrid scheme. Note that if the shift parameter is set to 0, the standard PI is then used to focus the search space.

A second option is to employ a subspace iteration method (we have chosen the IRAM scheme) to bootstrap the eigenpair search with Newton's method. This option is more appealing than the "SIPI-Newton" scheme because the Arnoldi iteration readily provides approximations to several dominant eigenpairs at once. They can be obtained with a coarse tolerance *without* having recourse to explicit deflations or the need for an initial eigenvalue estimate as in the SIPI method. Additionally, the subsequent Newton iterations to solve for the different modes are completely independent and this stage could be performed in parallel. Hereafter, we denote this scheme, where coarse-tolerance estimates from the IRAM technique are employed as initial iterates for the Newton solves, as the "Arnoldi-Newton" hybrid scheme.

Finally, we note that a fine tolerance is not necessary at the start of Newton solve in Eq. (6.13) when the search direction is far away from true solution. Then, the Jacobian solve can be performed 'inexactly' in the sense that the linear iteration tolerance can be made to depend on the non-linear function residual Eq. (6.13). This Inexact-Newton iteration method results in the following convergence criteria for the linear solve.

$$\left|\left|J(y^l)\delta y^l + F(y^l)\right|\right| \leq c^l \left|\left|F(y^l)\right|\right|, \tag{6.16}$$

where $c^l$ is a parameter chosen to tighten the linear solve convergence as the non-linear residual is reduced. For more information on Inexact-Newton schemes and the forcing factor $c^l$, we refer the reader to [59].

#### 6.1.2.2 Matrix-free Technique

In Jacobian-free variants of Newton's method, the explicit computation of the Jacobian matrix $J$ is not required, which is particularly useful in our case since the Jacobian matrix contains matrix $M = L^{-1}F$, which we do not want to compute explicitly nor store. Since the Jacobian-free method is at the core of the tightly coupled solution algorithm introduced for solving multi-physics problems, much of the infrastructure for implementing the hybrid eigenvalue solver is in place.

Let the $(N+1)$-dimensional Krylov vector $v$ be written as $(\varphi, \mu)^T$ where $\varphi$ is the portion of the Krylov vector corresponding to the eigenfunction and $\mu$ is a scalar corresponding to the eigenvalue. Then, the action of the Jacobian matrix on a Krylov vector $v$ can be obtained as

$$Jv \simeq \frac{F(\mathbf{y} + \epsilon v) - F(\mathbf{y})}{\epsilon} = \begin{bmatrix} M\varphi - \lambda\varphi - \mu(\phi + \epsilon\varphi) \\ -\phi^T\varphi - \frac{1}{2}\epsilon\varphi^T\varphi \end{bmatrix} \tag{6.17}$$

where it should be remembered that $z = M\varphi$ is actually the following linear solve: $Lz = F\varphi$. Eq. (6.17) is based on a finite difference approximation, which can be avoided here: since the definition of Jacobian is exactly available, albeit not explicitly, a matrix-free solve using GMRes, with no memory allocated for the Jacobian matrix itself, can be carried out and the (exact) matrix-vector operator is given by

$$Jv = \begin{bmatrix} M\varphi - \lambda\varphi - \mu\phi) \\ -\phi^T\varphi \end{bmatrix} \tag{6.18}$$

In Eq. (6.18), the effect of the perturbation $\epsilon$ has vanished. Both Eq. (3.76) and Eq. (6.18) are Jacobian-free approaches (the Jacobian matrix is not formed) and both forms require the same linear solve $Lz = F\varphi$. In our implementation, we have chosen the exact matrix-vector operation, i.e., Eq. (6.18), over the finite difference approximation of Eq. (3.76).

#### 6.1.2.3 Preconditioners for the JFNK Technique

In order to limit the size of the Krylov subspace in the linear solve at each Newton iteration, effective preconditioning techniques are necessary. The form of outer (Newton) – inner (Krylov) iteration for eigenvalue problems resembles the inverse iteration subspace family of methods and also is closely related the Jacobi-Davidson scheme [108, 109] when the secondary equation being used as a preconditioner is Point-Jacobi. The power of the MFNK scheme is that any number of preconditioners can be employed as long as (i) the cost of computing the preconditioner itself is cheaper than the cost of computing the true operator ($M$) and (ii) the preconditioner collapses the eigenspectrum so that the spread in eigenvalues is reduced. Note that every preconditioner solve can also be performed in a matrix-free fashion rather than actually forming the preconditioning matrix $P$, if memory requirements necessitate that.

Some details on the preconditioning methods for the simple eigenproblem in Equation (6.4) are now discussed.

#### 6.1.2.4 Block Gauss-Seidel Preconditioner

The Block Gauss-Seidel (BGS) form of the preconditioner, where a block is defined as one energy group, is a natural choice of preconditioner to invert $L$. In traditional reactor analysis codes using PI method, the BGS method is typically used to invert the loss operator solve. In the current JFNK context, we use a single BSG sweep over all groups as a preconditioner. Since each diagonal block $L_i$ is symmetric (when discretized using a standard finite element or finite difference method), efficient Krylov linear solvers such as Conjugate Gradient can be used to solve each one-group equation. Here, the individual blocks are themselves preconditioned with a level-0 Incomplete Cholesky decomposition, IC(0). Therefore, the preconditioner $P$ is given by

$$P = \begin{bmatrix} \tilde{L}^{-1}F - \lambda I & -\phi \\ -\phi^T & 0 \end{bmatrix} \tag{6.19}$$

where $\tilde{L}$ is the lower triangular block of $L$, hence discarding any upscattering terms. A preconditioner solve requires the following solves:

$$L_i z_i = (F\varphi)_i - \sum_{j<i} L_{i,j} z_j \quad \text{for } i = 1, \ldots, G \tag{6.20}$$

Other preconditioners such as multigrid, multilevel methods could be utilized instead of the BGS scheme. Another idea would be to use traditional PI technique as a preconditioner for the Newton solve, as suggested by Knoll [110].

### 6.1.3 Implementation

The methods presented here have been implemented in the KARMA code framework which interfaces to external libraries such as PETSc [68], SLEPc [73] and ARPACK [95] to make use of existing eigenvalue solvers for comparison purposes and to bootstrap the Newton iteration by providing appropriate initial guess.

The PETSc-based Scalable Library for Eigenvalue Problem Computations (SLEPc) library has several standard eigenvalue solvers such as PI, SIPI, Explicitly restarted Arnoldi method and Krylov-Schur methods. Apart from the built-in solvers, SLEPc also provides interfaces to the ARPACK eigensolver package. The IRAM algorithm employed as bootstrapping in our hybrid method was taken from ARPACK.

For the requested number dominant eigenmodes, ($nev$), the computational cost of IRAM behaves as $N\ell^2$ where $N$ is the problem rank of matrix $M$ and $\ell$ is the size of the subspace. Hence for fine tolerances and large problem sizes ($N$), a bigger span of Krylov space may be needed, increasing the total memory cost. Instead, the proposed hybrid Arnoldi-Newton scheme can be implemented in a completely matrix-free fashion using JFNK, with IRAM to deliver the initial eigenmodes to coarse tolerances (thus requiring only a smaller subspace) and Newton's method to drive the residual of the eigenproblem to a tight precision. This scheme can compute several dominant eigenpairs (in parallel) with low memory overhead and high accuracy with existing frameworks of PI implementation.

The hybrid SIPI-Newton scheme can also be used to converge to the dominant eigenpairs but usually it is expected to be inferior in terms of performance in comparison to advanced Krylov iteration techniques like Arnoldi to find multiple eigenvalues. This is due to the fact that the choice of the shift parameter affects the convergence of this scheme significantly and eigenpairs can only be found one at a time, i.e., sequentially. Nevertheless, the low memory requirements and the relative ease in implementation of this scheme as compared to the Arnoldi-Krylov method may make it attractive, especially since traditional reactor analysis codes already have most of the necessary framework in place. In order to make the results shown here independent of a user-provided shift, 20 iterations of power iteration are performed to obtain an eigenvalue estimate to 1e-2 tolerance. This then is used as the initial shift for all SIPI runs thereby making this hybrid scheme automated to some extent.

### 6.1.4 Results

Numerical results using the different eigensolvers introduced in the previous sections are presented here for three typical eigenvalue problems found in nuclear reactor analysis. The first case is a 2-D IAEA benchmark problem that is used to analyze the convergence of the implemented scheme for the first few eigenmodes. The next problem considered is a homogenous 2-group, 2-d problem to compare the traditional PI, state-of-art IRAM and the current hybrid algorithms in terms of efficiency in computing the fundamental mode, as a function of the dominance ratio.

#### 6.1.4.1 Case 1: IAEA 2-D Benchmark Problem

For the purpose of determining the accuracy and efficiency of the proposed numerical scheme, the well known IAEA 2-D benchmark problem (ANL, 1977) with modifications to the cross-section to account for z-leakage is used as a test problem. This problem is a two group model of a PWR quarter core with reflective boundary conditions on left and top sides and vacuum boundary on the remaining sides (zero incoming current). Details on the used cross-sections for different materials and the lattice representation is given in Appendix (B.1).

The 2-D domain is discretized using triangular mesh elements. The spatial discretization of the generalized eigenvalue problem is performed using piecewise quadratic, Lagrange elements on triangles. The reference results presented below for the eigenvalue computation were from a discretization with 81940 elements and 123937 unknowns/group.

The thermal flux profiles for the first five modes of the benchmark problem obtained using the hybrid Arnoldi-Newton iteration are shown in Fig. 6.1 - Fig. 6.5.

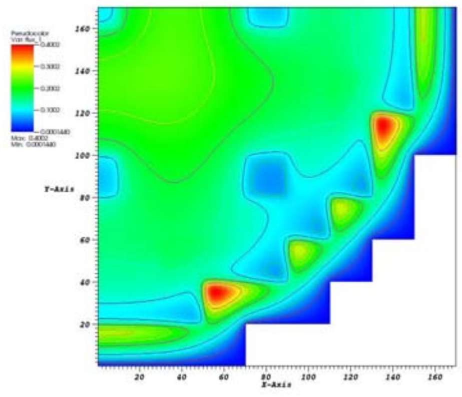


Figure 6.1: IAEA 2D Benchmark Problem: First Eigenmode for Thermal Flux.

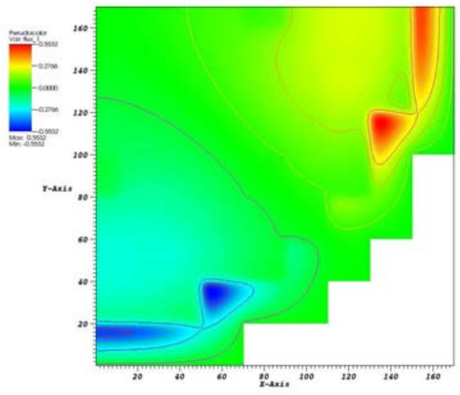


Figure 6.2: IAEA 2D Benchmark Problem: Second Eigenmode for Thermal Flux.

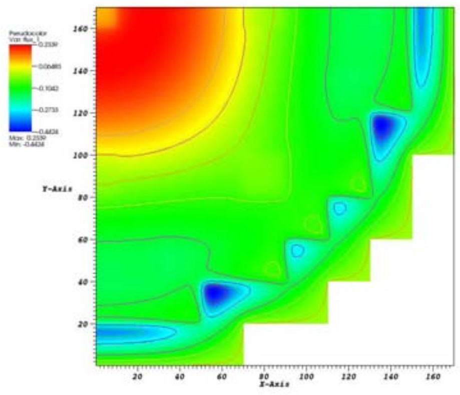


Figure 6.3: IAEA 2D Benchmark Problem: Third Eigenmode for Thermal Flux.

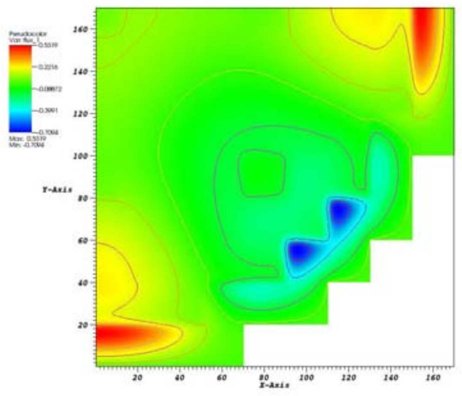


Figure 6.4: IAEA 2D Benchmark Problem: Fourth Eigenmode for Thermal Flux.

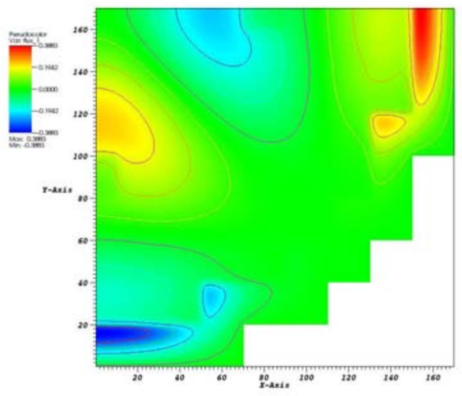


Figure 6.5: IAEA 2D Benchmark Problem: Fifth Eigenmode for Thermal Flux.

In Table (6.1), the eigenvalues for the first 10 modes are shown for the benchmark problem along with the L2 norm of the residual error $M\phi_i$-$\lambda_i\phi_i$. The eigenvalues from the IRAM-Newton hybrid scheme were compared to the solution from a fine tolerance (1E-14) IRAM run that is implemented in ARPACK, interfaced through SLEPc. The eigenvalues computed from both schemes match exactly and hence this test case proves the convergence of the hybrid scheme to all the desired modes. In the above run, an IRAM iteration provides a reasonable guess of the eigenmodes to a coarse tolerance of only 1E-3 with maximum subspace size of 15 to bootstrap the inexact Newton iteration.

Table 6.1: Eigenvalues for several modes computed using IRAM-Newton iteration scheme

| **Mode** | **IRAM-Newton** | **Residual Error** |
|---|---|---|
| 1 | 1.02958492118978 | 3.925623517e-14 |
| 2 | 1.00262113845152 | 1.665425359e-14 |
| 3 | 0.99162639339383 | 4.511323367e-14 |
| 4 | 0.93909498902443 | 3.756085782e-14 |
| 5 | 0.91382020547476 | 4.287420904e-13 |
| 6 | 0.90139826550461 | 1.052507002e-13 |
| 7 | 0.89045692528088 | 3.088887872e-14 |
| 8 | 0.82718775002535 | 1.762787579e-13 |
| 9 | 0.82499156741152 | 3.749614193e-14 |
| 10 | 0.81562246518003 | 4.042513635e-14 |

The results shown in Table (6.1) indicate that the new numerical scheme computes the eigenpairs with machine precision accuracy for all the requested modes (nev=10) of the model problem. Hence the new convergence of the

Newton based hybrid scheme to the true numerical eigenmode is guaranteed as long as the initial guess provided is in the 'ball of convergence'.

Next, the same problem is solved using different eigen-methods and the cost of computation in terms of number of BGS operator applications is listed in Table (6.2). Also the size of the subspace indicates the memory cost requirements needed for solving the problem. The results indicate that the SIPI based schemes are unfeasible in terms of the total cost due to the chosen method for determining the initial shift. If a better algorithm is available for this, the higher eigenmodes can be computed with lesser cost. On the other hand, IRAM and IRAM-Newton hybrid scheme show very efficient performance and the hybrid scheme only requires half the memory requirement as compared to pure IRAM run. The results show the strength in terms of convergence with lower memory cost of the hybrid scheme based on Arnoldi iteration.

Table 6.2: Eigenvalues for 10 eigenmodes using different iteration schemes

| **Method** | **Number of operators** | **Subspace size** |
|---|---|---|
| SIPI | 8115 | 1 |
| SIPI-Newton | 6507 | 1 |
| IRAM | 2180 | 20 |
| IRAM-Newton | 2116 | 10 |

The preliminary results show that there is an optimal subspace-size for IRAM which minimizes the total number of BGS operators. But a-priori, this size is not known and can only be determined by experiments on the problem of interest. Also, as the subspace-size increases, the memory requirements increase linearly and can become prohibitive unless distributed systems are utilized. The IRAM-Newton scheme alternately uses IRAM only to start the iteration and hence could gain immensely from the usage of a much lesser subspace size to achieve coarse tolerances. And if auxillary systems representing the Jacobian matrix can be used as preconditioners, the total cost of the linear Krylov solve can also be reduced considerably.

#### 6.1.4.2 Case 2: Homogenous Infinite Medium, 2-Group Problem with Up-scattering

Consider a 2-D, two energy group problem with non-zero, fast-to-thermal and thermal-to-fast scattering cross-section. Hence, the Loss matrix $L$ is a full block matrix, unlike the IAEA 2D case which was in block-lower triangular form. The cross-section data used for the problem are given in Appendix (B.2). The implementation was verified by checking the spatial convergence of the eigenvalue to the exact eigenvalue, as the mesh is refined.

In the infinite medium limit, the dominance ratio for the discretized, generalized MGND eigenvalue problem can be made to approach unity. This translates to a large number of iterations that is needed in order to converge to the fundamental eigenpair. If several dominant eigenpairs are desired in this context, traditional power iteration techniques are practically infeasible.

Better Preconditioners such as multigrid, multilevel methods for such elliptic systems will significantly improve the performance of the linear solves and hence reduce the number of operator applications for the Inexact-Newton based schemes. Such possibilities are left for future investigation and currently only the BGS preconditioner is used currently to resolve the neutronic system with upscattering.

In this setting, the problem is solved using power iteration, IRAM, SIPI-

Newton hybrid iteration and Arnoldi-Newton hybrid iteration for various dominance ratios. The results shown below arise from the discretization of the homogenous problem using piecewise quadratic Lagrange basis functions on a structured grid with 400 QUAD8 elements and 1281 dofs/group. The cost results obtained from various test runs with a tolerance of 1E-10 are shown in Table (6.3). All the columns list the total number of BGS operator applications since the total cost is a function of this parameter.

It is quite evident that if the subspace size is increased for the Arnoldi iteration, the number of operator count will decrease until the optimal size is used. This was shown by [90]. Since it is hard to calculate this optimal subspace size a-priori and that larger memory requirements prohibit storing many vectors, the situation where limited subspace size is prescribed has been considered here.

Table 6.3: Number of operator applications needed for different schemes as a function of Dominance Ratio (DR)

| **Length** | **DR** | **Power** | **CSIPI** | **SIPI-Newton** | **IRAM** | **IRAM-Newton** |
|---|---|---|---|---|---|---|
| 12.255 | 0.3500 | 83 | 75 | 68 | 47 | 62 |
| 23.127 | 0.5000 | 147 | 121 | 96 | 73 | 89 |
| 85.95 | 0.9001 | 980 | 285 | 216 | 180 | 153 |
| 126.38 | 0.9500 | 1916 | 443 | 394 | 276 | 188 |
| 291.1 | 0.9900 | 7317 | 1133 | 563 | 353 | 284 |
| 926.5 | 0.9990 | 74270 | 5396 | 786 | 1488 | 692 |
| 2931 | 0.9999 | 604400 | 8478 | 1069 | 1899 | 782 |

The shift parameter for SIPI is found by performing several power iterations to coarse tolerance. The hybrid SIPI-Newton and Arnoldi-Newton schemes use a coarse tolerance of 1E-3 to obtain the initial guess and the eigenpair is converged to the user specified tolerance by the inexact Newton algorithm. For all calculations involving Arnoldi based schemes, the size of the subspace has been set at a fixed value of 10 for the current problem.

As expected, the results prove that as the dominance ratio approaches unity, the number of power iterations increase exponentially. The Arnoldi iteration converges to the dominant pair in much lesser iterations but the true power of this scheme, finding multiple eigenpairs simultaneously, is not utilized here. On the other hand, the hybrid SIPI-Newton scheme performs the quite well unlike the IAEA 2-D benchmark problem and this depends on the methodology to choose the initial shift parameter. Hence, when a guess for the dominant eigenvalue is available, the SIPI-Newton scheme can be the optimal method of choice and but otherwise, the hybrid Arnoldi-Newton scheme provides a gain of atleast factor of 2 as compared to just using IRAM scheme.

Clearly, results in Table (6.3) show the better convergence of the hybrid schemes for higher dominance ratio problems. This test result proves the feasibility of such schemes to resolve the fundamental eigenmodes for even strongly coupled (in terms of energy groups) problems. Further studies are necessary to improve the bootstrapping procedure and implementation of more efficient preconditioners for these block-symmetric system of equations.

### 6.1.5 Closing Remarks

The hybrid technique proposed using inexact Newton iteration is proven to be quite effective for the problems considered and delivers performance and convergence on-par to the state-of-art IRAM scheme. Future work is necessary to gauge the applicability of various kinds of preconditioners, other than the ones shown in this section, to improve computational efficiency. The flexibility of the MFNK framework to include these ideas apart from solving tightly-coupled multi-physics simulations, affirms that KARMA framework code can tackle linear, non-linear, eigenvalue and transient problems with different preconditioning approaches and coupling methods under "one roof".

## 6.2 Non-equilibrium Radiation Diffusion Physics Problem

The Boltzmann transport equation describes the behavior of radiation traveling through a material. It is generally considered as the most accurate model for the statistical average density of particles in a system, with very few assumptions. Non-equilibrium radiation transport physics deals specifically with the transport of photon energy and its coupling with a background material. It occurs at the heart of stars and several high energy physics systems [111], where there are steep energy and temperature gradients.

Several approximations that can be applied to the Boltzmann equation since solving the transport equation is computationally quite expensive. In the current work, we use the gray diffusion approximation with flux limiters [112]. The system of equations describing the radiation and material energy fields used here are given as

$$\frac{\partial E}{\partial t} - \vec{\nabla}\cdot(D_r\vec{\nabla}E) = \sigma_a(T^4 - E), \tag{6.21}$$

$$\frac{\partial T}{\partial t} - \vec{\nabla}\cdot(D_t\vec{\nabla}T) = \sigma_a(E - T^4), \tag{6.22}$$

where $E, T$ are the radiation energy and material temperatures respectively, $D_r, D_t$ are the radiation and temperature diffusion coefficients and $\sigma_a$ is the photon absorption coefficient.

The definitions for these material properties [113] are

$$\sigma_a = \frac{z^3}{T^3}, \tag{6.23}$$

$$D_r(T) = \frac{1}{3\sigma_a + (1/E)|\frac{\partial E}{\partial x}|}, \tag{6.24}$$

$$D_t(T) = kT^{5/2}, \tag{6.25}$$

with $z$ being the material atomic number and $k$ is a constant.

Recently, the use of tightly coupled methods for non-equilibrium radiation diffusion with Jacobian-Free Newton Krylov methods has been analyzed [27, 20, 2] and its superiority over Operator-Split loosely coupled methods are provided. Based on these ideas, the KARMA framework is used in the current work to solve problems with different discretizations and coupling methods in a single code system.

Since radiation diffusion problems in general have strong gradients, traditional cG FEM spatial discretization of the elliptic operator in Eq. (6.22) is unstable without additional stability preserving terms added to the weak form. These family of methods, generally referred to as Stabilized Finite Element Methods (SFEM) [114] have enjoyed much success in advection dominated problems. The above set of equations can also be discretized with a dG(0) formulation or with the traditional Finite Volume Method (FVM) to conserve the radiation energy solution field throughout the transient. Preliminary studies with both these spatial treatments have shown that SFEM necessitates the selection of an optimal parameter in order to avoid unphysical solutions and is not guaranteed to be absolutely stable while FVM requires the usage of fine mesh resolution in order to gain better solution accuracy. Hence, all results shown in the current section will use FVM for spatial discretization with high number of elements to capture the energy and temperature profiles accurately.

Once the spatial scales are resolved by an appropriate spatial treatment, the temporal integration can be performed using $L-$stable schemes such as SDIRK2(2) and SDIRK3(3). It is also possible to use adaptive time-stepping strategies to choose a time step based on the dominating dynamical time scale of the problem. In order to correctly resolve the solution fields from both the radiation and material temperature accurately, a minimum of the time scales for the solution evolution is computed and used based on the following controller.

*Dynamical time scale controller:*

$$\tau_{dyn}^{n} = \frac{S}{\left|\frac{1}{\phi}\frac{\partial\phi}{\partial t}\right|_{n}} \approx \Delta t_n \frac{S}{\left|\frac{\phi_{n+1}-\phi_n}{\phi_{n+1}}\right|} \tag{6.26}$$

where $\phi$ is the solution field for current physics and $S$ is a safety factor.

### 6.2.1 One-dimensional Problem

In order to analyze the different space-time discretization and coupling methods, we consider a one-dimensional model problem that consists of a unit radiation flux impinging on an initially cold slab of unit depth. This results in with Robin boundary conditions for the radiation equations at $x = 0$ and $x = 1$. These conditions are

$$\frac{1}{4}E - \frac{1}{6\sigma_a}\frac{\partial E}{\partial x} = 1, x = 0, \tag{6.27}$$

$$\frac{1}{4}E + \frac{1}{6\sigma_a}\frac{\partial E}{\partial x} = 0, x = 1. \tag{6.28}$$

The material temperature has homogenous Neumann conditions imposed at the boundaries. The initial solution fields are

$$E(x,0) = 10^{-5}, \forall x = [0,1], \tag{6.29}$$

$$T(x,0) = E(x,0)^{1/4} \approx 0.0562. \tag{6.30}$$

The material atomic number $z$ for the homogenous medium is taken to be 1.0. Taking $k = 0.1$, we replicate the results provided by Mousseau et al. [113].

### 6.2.2 Results

The problem shown above was discretized with FVM using 800 elements and SDIRK3(3) method in both space and time, respectively, and solved using both weakly and strongly coupled methods. The solution profiles at a final time of T=2.5 seconds obtained using an adaptive time step controller are shown in Fig. 6.6.

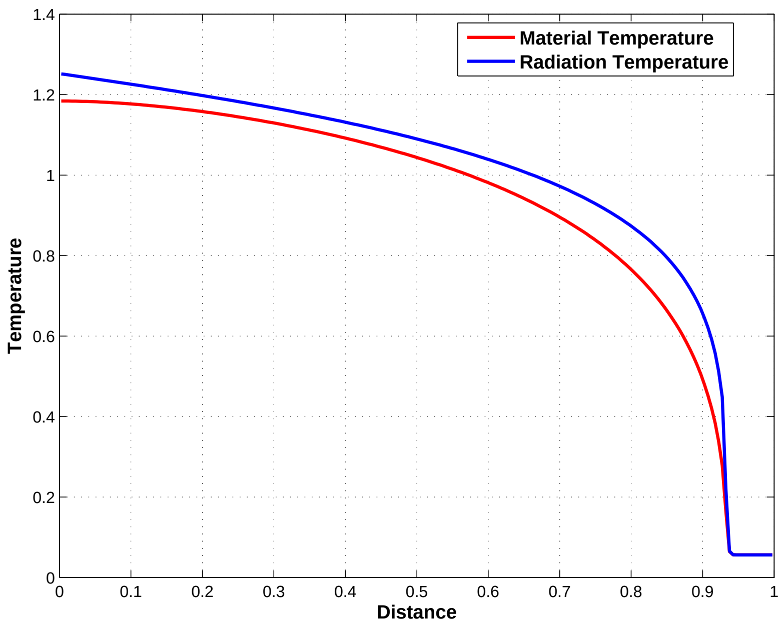


Figure 6.6: Non-equilibrium Radiation Diffusion Test Problem: Radiation and Material Temperature Profiles.

Note that the results given by Mousseau in [113] for the same test problem do not match the solutions shown here. The wave propagation speed of the energy source due to the left boundary condition is resolved differently in Fig. 6.6 but it converges consistently to a reference solution as the safety factor $S$ is reduced.

Next, we will look at the different tight coupling strategies (Picard versus Newton), based on the linearized physics-based Jacobian that is lagged at the previous step along with ILU(0) as an algebraic preconditioner. The results for the computational cost of the tightly coupled methods with constant time stepping using different step sizes are shown below.

Table 6.4: CPU time for one-dimensional problem using Picard vs Newton iteration

| **Δ t** | **Picard-Krylov** | **Newton-Krylov** |
|---|---|---|
| $4 \times 10^{-3}$ | – | 78.3 |
| $2 \times 10^{-3}$ | 236.624 | 127.143 |
| $1 \times 10^{-3}$ | 354.773 | 144.921 |

From Table 6.4, we can infer that the cost of the Newton-Krylov solution procedure increases sub-linearly as the number of steps is increased. Also due to the usage of the linearized Jacobian as only a preconditioner, higher time step sizes can be used to solve the system of equations. This is not the case for Picard iterations using the linearized Jacobian as the operator with ILU(0) as its preconditioner, since the solution starts to diverge for large time step sizes.

### 6.2.3 Closing Remarks

The flexibility of using the KARMA multi-physics code system is evident from the experiments conducted for the problems involving radiation diffusion physics using different spatial and temporal discretization schemes, along with a variety of consistent coupling methods. Also, using high-order methods with time adaptivity enables the ability to accurately capture the solution evolution based on its dynamical time scales. Further investigation is necessary to ascertain the reasons for the disparity in the results obtained here compared to Mousseau.

# Chapter 7

# Conclusions

'Computers are useless. They can only give you answers.'
– Pablo Picasso

Numerical simulation of multi-physics problems are difficult due to the need to resolve the stiff variations in spatial and temporal scales. This project used tight coupling methods for multi-physics problems in order to retain high-order spatio-temporal accuracy in the computed solution fields for stiff transients occurring in nuclear reactors and other fields (radiative transfer). The lessons learnt from using tight coupling methods for these problems are summarized below along with envisioned extensions to the KARMA framework that would provide further intuition for the develop efficient methods for coupled multi-physics problems.

## 7.1 Lessons Learned

1. Existing coupling methods using Operator-Splitting (OS) strategies were analyzed and the deficiencies in these methods including condi-

tional stability and degradation of the higher order temporal accuracy were shown. In contrast, tightly coupled methods with either Picard or Newton iterations do restore the accuracy in the numerical solution, thereby reducing the total computation time where stability limits do not dominate the dynamical physical scales.

2. The tightly coupled methods can be unified under a single Matrix-free Nonlinear-Krylov (MFNK) framework that is based on a finite-differenced expression to obtain the action of the Jacobian matrix representing the coupled system matrix on a vector. Existing OS strategies offer intuitive knowledge regarding the important length and time scales to be tackled which can be used to create specialized schemes that serve as good preconditioners to reduce the total computational time in the MFNK technique. Note that usually such preconditioners are used as solvers themselves if the coupling strength between the physics terms is weak.

3. The efficacy of the tightly coupled schemes are superior as compared to the loosely coupled OS schemes. Slight modifications on existing coupling strategies by introducing Picard-like iterations or performing fewer Newton iteration can improve the stability regions of the simple OS strategies.

4. The variations in time scales during the course of a multi-physics transient problem are often not dominated by a single physics component alone. The complex interplay of these temporal scales requires adaptive techniques in order to resolve the solution fields accurately in amenable wall-clock times. Tightly coupled schemes with adaptive time step-

ping procedures are excellent candidates to attain better efficacy using higher order, $L-$stable, temporal methods.

5. Method of Manufactured Solutions (MMS) and analytical methods are effective tools for the verification of non-linear multi-physics problems. Even though high fidelity physical models were not used in this work, the application of these techniques to even complicated cases is possible using symbolic mathematical toolboxes.

## 7.2 Future Work

The non-linear solution methods for multi-physics problems given in this work offers tremendous scope for future extensions. Few of the research areas that need to be focussed in the near future are listed below.

1. Create a multi-channel analysis code within the existing KARMA framework in order to apply the code system to problems in design and safety analysis of nuclear reactors.

2. Use the ability to create an array of varying fidelity physics models with the KARMA framework in order to verify the application of efficient, fidelity-independent coupling strategies.

3. Use spatial adaptive capabilities from hp-FEM ideas and adjoint based error estimators. Similar principles apply to time adaptivity using adjoint based temporal adaptivity.

4. If rigorous a-priori estimates for determining the strength of the coupling terms can be obtained based of $Jv$ and $\tilde{J}v$, where $\tilde{J}$ is the OS Jacobian matrix, an adaptive solution procedure selection can be made

to choose from using either OS coupled strategy as the solver itself or as a preconditioner for the tightly coupled solution technique. This has tremendous potential to reduce the total runtimes in regions of the transient when the dynamical time scales are themselves well below the stability limits for OS coupling. Such algorithms can create computationally efficient, high accuracy schemes while making use of the unified MFNK framework proposed here.

5. The KARMA framework has extensive parallel capabilities since it is based on PETSc for all the relevant data-structures. Preliminary results for single-physics non-linear diffusion problems and coupled neutronics/thermal-conduction problems have shown linear speedup for up to 32 processors. Further studies need to be performed to measure the scalability of tightly coupled methods and for finding efficient parallel solve/preconditioning techniques for problems of interest in reactor analysis.

6. Finally, the KARMA framework is implemented with a flexible API. With only definitions of the semi-discrete non-linear residual that is consistent with the actual PDE for the physics model, the MFNK can be used to solve the non-linear, time-dependent problem. Improved efficiency can also be obtained if an external code can provide a suitable preconditioner to reduce the total Krylov iterations. This lends the possibility to use existing single-physics codes in order to attain tightly coupled solutions, when necessary, thereby preserving numerous man-years of development efforts on these codes.

# Bibliography

[1] R. B. Lowrie, A comparison of implicit time integration methods for nonlinear relaxation and diffusion, Journal of Computational Physics 196 (2) (2004) 566–590.

[2] P. N. Brown, D. E. Shumaker, C. S. Woodward, Fully implicit solution of large-scale non-equilibrium radiation diffusion with high order time integration, Journal of Computational Physics 204 (2) (2005) 760–783.

[3] W. Woodruff, A kinetics and thermalhydraulics capability for the analysis of research reactors, Nucl. Technol. 64 (2) (1984) 196–206.

[4] C. Housiadas, Lumped parameters analysis of coupled kinetics and thermal-hydraulics for small reactors, Annals of Nuclear Energy 29 (11) (2002) 1315–1325.

[5] C. A. Felippa, K. C. Park, Staggered transient analysis procedures for coupled mechanical systems: Formulation, Computer Methods in Applied Mechanics and Engineering 24 (1) (1980) 61–111.

[6] J.-F. Gerbeau, M. Vidrascu, P. Frey, Fluid-structure interaction in blood flows on geometries based on medical imaging, Computers & Structures 83 (2-3) (2005) 155–165.

[7] T. Gratsch, K.-J. Bathe, Goal-oriented error estimation in the analysis of fluid flows with structural interactions, Computer Methods in Applied Mechanics and Engineering 195 (41-43) (2006) 5673–5684.

[8] P. Wriggers, C. Miehe, Contact constraints within coupled thermomechanical analysis–A finite element model, Computer Methods in Applied Mechanics and Engineering 113 (3-4) (1994) 301–319.

[9] C. Newman, G. Hansen, D. Gaston, Three dimensional coupled simulation of thermomechanics, heat, and oxygen diffusion in UO2 nuclear fuel rods, Journal of Nuclear Materials 392 (1) (2009) 6–15.

[10] R. Klein, S. Doebling, F. Graziani, M. Pilch, T. Trucano, ASC Predictive Science Academic Alliance Program Verification and Validation Whitepaper, Tech. Rep. UCRL-TR-220711, Lawrence Livermore National Laboratory (LLNL) (2006).

[11] V. S. Mahadevan, Nonlinearly consistent schemes for coupled problems in reactor analysis, M.S thesis, Texas A&M University, College Station, TX (2006).

[12] J. C. Ragusa, V. S. Mahadevan, Consistent and accurate schemes for coupled neutronics thermal-hydraulics reactor analysis, Nuclear Engineering and Design 239 (3) (2009) 566–579.

[13] A. Kavenoky, J. Lautard, The neutron kinetics and thermal-hydraulic transient computational module of the neptune system: CRONOS, in: Proc. Topical Meeting on Advances in Reactor Physics and Core Thermal Hydraulics, Kiamesha Lake, NY, 22-24 September, 1982, 781–792.

[14] I. Toumi, A. Bergeron, D. Gallo, E. Royer, D. Caruge, FLICA-4: a three-dimensional two-phase flow computer code with advanced numerical methods for nuclear applications, Nuclear Engineering and Design 200 (1-2) (2000) 139–155.

[15] Y. Xu, T. Downar, R. Walls, K. Ivanov, J. Staudenmeier, J. March-Lueba, Application of TRACE/PARCS to BWR stability analysis, Annals of Nuclear Energy 36 (3) (2009) 317–323.

[16] E. Uspuras, A. Kaliatka, E. Bubelis, Validation of coupled neutronic/thermal-hydraulic code RELAP5-3D for RBMK-1500 reactor analysis application, Annals of Nuclear Energy 31 (15) (2004) 1667–1708.

[17] G. Strang, On the construction and comparison of difference schemes, SIAM Journal on Numerical Analysis 5 (3) (1968) 506–517.

[18] G. I. Marchuk, On the theory of the splitting-up method, Vol. II of Numerical Solution of Partial Differential Equations, Academic Press, New York, 1971.

[19] D. L. Ropp, J. N. Shadid, C. C. Ober, Studies of the accuracy of time integration methods for reaction-diffusion equations, Journal of Computational Physics 194 (2) (2004) 544–574.

[20] C. C. Ober, J. N. Shadid, Studies on the accuracy of time-integration methods for the radiation-diffusion equations, Journal of Computational Physics 195 (2) (2004) 743–772.

[21] E. J. Weniger, Prediction properties of Aitken's iterated $\Delta^2$ process, of Wynn's epsilon algorithm, and of Brezinski's iterated theta algorithm,

Journal of Computational and Applied Mathematics 122 (1-2) (2000) 329–356.

[22] P. N. Brown, Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, SIAM J. Sci. Stat. Comput. 11 (3) (1990) 450–481.

[23] D. A. Knoll, D. E. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, Journal of Computational Physics 193 (2) (2004) 357–397.

[24] D. A. Knoll, L. Chacon, L. G. Margolin, V. A. Mousseau, On balanced approximations for time integration of multiple time scale systems, Journal of Computational Physics 185 (2) (2003) 583–611.

[25] V. A. Mousseau, Implicitly balanced solution of the two-phase flow equations coupled to nonlinear heat conduction, Journal of Computational Physics 200 (1) (2004) 104–132.

[26] V. A. Mousseau, M. A. Pope, Accuracy and efficiency of a coupled neutronics and thermal-hydraulics model, Nuclear Engineering and Technology 41 (2009) 885–893.

[27] V. A. Mousseau, D. A. Knoll, New physics-based preconditioning of implicit methods for non-equilibrium radiation diffusion, Journal of Computational Physics 190 (1) (2003) 42–51.

[28] V. S. Mahadevan, J. C. Ragusa, Coupling schemes for multiphysics reactor simulation, Tech. Rep. INL/EXT-07-13490, Idaho National Laboratory, Idaho Falls, ID (2007).

[29] J. J. Duderstadt, L. J. Hamilton, Nuclear Reactor Analysis, John Wiley & Sons, New York, 1976.

[30] C.-D. Munz, M. Dumbser, S. Roller, Linearized acoustic perturbation equations for low Mach number flow with variable density and temperature, Journal of Computational Physics 224 (1) (2007) 352–364.

[31] F. H. Harlow, A. A. Amsden, A numerical fluid dynamics calculation method for all flow speeds, Journal of Computational Physics 8 (2) (1971) 197–213.

[32] R. C. Martineau, R. A. Berry, The pressure-corrected ICE finite element method for compressible flows on unstructured meshes, Journal of Computational Physics 198 (2) (2004) 659–685.

[33] B. Muller, Low-Mach-Number Asymptotics of the Navier-Stokes equations, Journal of Engineering Mathematics 34 (1998) 97–109.

[34] P. Solin, K. Segeth, I. Dolezel, Higher-order Finite Element Methods, Czech Republic Series: Studies in Advanced Mathematics, Chapman and Hall/CRC, Boca Raton, FL, 2003.

[35] J. S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods, Vol. 54 of Algorithms, Analysis and Applications Series: Texts in Applied Mathematics, Springer, New York, 2008.

[36] G. Kesserwani, R. Ghostine, J. Vazquez, A. Ghenaim, R. Mose, Riemann Solvers with Runge–Kutta discontinuous Galerkin schemes for the 1D shallow water equations, Journal of Hydraulic Engineering 134 (2) (2008) 243–255.

[37] P. D. Lax, Weak solutions of nonlinear hyperbolic equations and their numerical computation, Communications on Pure and Applied Mathematics 7 (1) (1954) 159–193.

[38] W. J. Rider, R. B. Lowrie, The use of classical Lax-Friedrichs Riemann solvers with discontinuous Galerkin methods, International Journal for Numerical Methods in Fluids 40 (3-4) (2002) 479–486.

[39] R. Hartmann, The role of the Jacobian in the adaptive discontinuous Galerkin method for the compressible Euler equations, in: G. Warnecke (Ed.), Analysis and Numerics for Conservation Laws, Springer, 2005, 301–316.

[40] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, J. Comput. Phys. 183 (2) (2002) 508–532.

[41] A. de Boer, A. H. van Zuijlen, H. Bijl, Review of coupling methods for non-matching meshes, Computer Methods in Applied Mechanics and Engineering 196 (8) (2007) 1515–1525.

[42] J. Grandy, Conservative remapping and region overlays by intersecting arbitrary polyhedra, J. Comput. Phys. 148 (2) (1999) 433–466.

[43] X. Jiao, M. T. Heath, Common-refinement-based data transfer between non-matching meshes in multiphysics simulations, International Journal for Numerical Methods in Engineering 61 (14) (2004) 2402–2427.

[44] T. J. Tautges, A. Caceres, Scalable parallel solution coupling for multiphysics reactor simulation, Journal of Physics: Conference Series 180 (2009) 12–17.

[45] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, Vol. 14 of Computational Mathematics, Springer Series, New York, 1996.

[46] E. Hairer, S. Norsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, 2nd Edition, Vol. 8 of Computational Mathematics, Springer Series, New York, 1993.

[47] G. G. Dahlquist, A special stability problem for linear multistep methods, BIT Numerical Mathematics 3 (1) (1963) 27–43.

[48] R. K. Alexander, Stability of Runge-Kutta methods for stiff ordinary differential equations, SIAM J. Numer. Anal 31 (1994) 1147–1168.

[49] K. Eriksson, C. Johnson, A. Logg, Explicit time-stepping for stiff ODEs, SIAM J. Sci. Comput. 25 (4) (2003) 1142–1157.

[50] K. Gustafsson, M. Lundh, G. Soderlind, A PI stepsize control for the numerical solution of ordinary differential equations, BIT Numerical Mathematics 28 (2) (1988) 270–287.

[51] I. Fried, D. S. Malkus, Finite element mass matrix lumping by numerical integration with no convergence rate loss, International Journal of Solids and Structures 11 (4) (1975) 461–466.

[52] A. Kvaerno, Singly diagonally implicit Runge-Kutta methods with an explicit first stage, BIT Numerical Mathematics 44 (3) (2004) 489–502.

[53] C. A. Kennedy, M. H. Carpenter, Additive Runge-Kutta schemes for convection-diffusion-reaction equations, Applied Numerical Mathematics 44 (1-2) (2003) 139–181.

[54] O. Axelsson, A class of A-stable methods, BIT Numerical Mathematics 9 (3) (1969) 185–199.

[55] Y. Nievergelt, Aitken's and Steffensen's accelerations in several variables, Numerische Mathematik 59 (1) (1991) 295–310.

[56] D. E. Roberts, The vector epsilon algorithm: a residual approach, Numerical Algorithms 29 (1) (2002) 209–227.

[57] Y. Xu, A matrix free Newton/Krylov method for coupling complex multi-physics subsystems, Ph.D. dissertation, Purdue University, West Lafayette, IN (2004).

[58] S. C. Eisenstat, H. F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1) (1996) 16–32.

[59] R. Dembo, S. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (1982) 400–408.

[60] P. McHugh, D. Knoll, Inexact Newton's method solution to the incompressible Navier-Stokes and energy equations using standard and matrix-free implementations, in: AIAA Computational Fluid Dynamics Conference, 11th, Orlando, FL, 1993.

[61] Y. Saad, M. H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.

[62] M. J. Grote, T. Huckle, Parallel preconditioning with sparse approximate inverses, SIAM J. Sci. Comput. 18 (3) (1997) 838–853.

[63] A. Krechel, K. Stuben, Parallel algebraic multigrid based on subdomain blocking, Parallel Computing 27 (8) (2001) 1009 – 1031.

[64] S. F. Ashby, P. N. Brown, M. R. Dorr, A. C. Hindmarsh, A linear algebraic analysis of Diffusion Synthetic Acceleration for the Boltzmann transport equation, SIAM Journal on Numerical Analysis 32 (1) (1995) 128–178.

[65] M. Benzi, Preconditioning techniques for large linear systems: a survey, Journal of Computational Physics 182 (2) (2002) 418 – 477.

[66] V. S. Mahadevan, J. C. Ragusa, High-order spatio-temporal schemes for coupled, multi-physics reactor simulations, Tech. Rep. INL/EXT-08-14884, Idaho National Laboratory, Idaho Falls, ID (2008).

[67] D. Gaston, C. Newman, G. Hansen, D. Lebrun-Grandie, MOOSE: A parallel computational framework for coupled systems of nonlinear equations, Nuclear Engineering and Design 239 (10) (2009) 1768–1778.

[68] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Efficient management of parallelism in object-oriented numerical software libraries, in: E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhauser Press, Cambridge, MA, 1997, 163–202.

[69] B. S. Kirk, J. W. Peterson, R. H. Stogner, G. F. Carey, `libMesh`: a c++ library for parallel adaptive mesh refinement/coarsening simulations, Engineering with Computers 22 (3–4) (2006) 237–254.

[70] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, International

Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331.

[71] G. K. Kumar, V. Kumar, Vipin, A parallel algorithm for multilevel graph partitioning and sparse matrix ordering, Journal of Parallel and Distributed Computing 48 (1998) 71–95.

[72] J. Ahrens, C. Law, W. Schroeder, K. Martin, K. Inc, M. Papka, A parallel approach for efficiently visualizing extremely large, time-varying datasets, Tech. Rep., Los Alamos Laboratory, Los Alamos, NM (2000).

[73] V. Hernandez, J. E. Roman, V. Vidal, SLEPc: A scalable and flexible toolkit for the solution of Eigenvalue problems, ACM Trans. Math. Softw. 31 (3) (2005) 351–362.

[74] L. Thomason, A simple, small, minimal, c++ xml parser, `http://sourceforge.net/projects/tinyxml/` (May 2007).

[75] P. J. Roache, Code verification by the Method of Manufactured Solutions, Journal of Fluids Engineering 124 (1) (2002) 4–10.

[76] B. Ganapol, Analytical benchmarks for nuclear engineering applications: case studies in neutron transport theory, Tech. Rep., NEA, Paris (2008).

[77] P. Knupp, K. Salari, Verification of Computer Codes in Computational Science and Engineering (Discrete Mathematics and Its Applications), Chapman and Hall/CRC, Boca Raton, FL, 2002.

[78] Argonne Code Center: Benchmark Problem Book, Vol. ANL-7416, Suppl. 2, Argonne National Laboratory, Argonne, IL, 1977.

[79] R. Park, D. Gaston, S. Kadioglu, D. Knoll, D. Lebrun-Grandie, R. Martineau, W. Taitano, Tightly coupled multiphysics simulation for pebble bed reactors, in: International Conference on Mathematics, Computaional Methods & reactor physics (M&C 2009), Saratoga Springs, New York, 2009.

[80] N. Wiener, The homogeneous chaos, American Journal of Mathematics 60 (4) (1938) 897–936.

[81] G. Em Karniadakis, J. Glimm, Preface: uncertainty quantification in simulation science, J. Comput. Phys. 217 (1) (2006) 1–4.

[82] H. N. Najm, Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics, Annual Review of Fluid Mechanics 41 (1) (2009) 35–52.

[83] D. Hagrman, G. Reyman, MATPRO-Version 11. A handbook of materials properties for use in the analysis of Light Water Reactor fuel rod behavior, Tech. Rep. NUREG/CR-0497, TREE-1280, Rev. 3 (1979).

[84] J. A. Boure, A. E. Bergles, L. S. Tong, Review of two-phase flow instability, Nuclear Engineering and Design 25 (2) (1973) 165–192.

[85] J. March-Leuba, J. Rey, Coupled thermohydraulic-neutronic instabilities in boiling water nuclear reactors: a review of the state of the art, Nuclear Engineering and Design 145 (1-2) (1993) 97–111.

[86] A. Hotta, Y. Suzawa, H. Takeuchi, Development of BWR regional instability model and verification based on ringhals 1 test, Annals of Nuclear Energy 24 (17) (1997) 1403–1427.

[87] V. Vidal, G. Verdu, D. Ginestar, J. L. Munoz-Cobo, Variational acceleration for subspace iteration method. Application to nuclear power reactors, International Journal for Numerical Methods in Engineering 41 (3) (1998) 391–407.

[88] G. Verdu, D. Ginestar, R. Miro, V. Vidal, Using the Jacobi-Davidson method to obtain the dominant lambda modes of a nuclear power reactor, Annals of Nuclear Energy 32 (11) (2005) 1274–1296.

[89] D. Ginestar, R. Miro, G. Verdu, D. Hennig, A transient modal analysis of a BWR instability event, Journal of Nuclear Science and Technology 39 (5) (2002) 554–563.

[90] F. Zinzani, C. Demaziere, C. Sunde, Calculation of the Eigenfunctions of the two-group neutron diffusion equation and application to modal decomposition of BWR instabilities, Annals of Nuclear Energy 35 (11) (2008) 2109 – 2125.

[91] M. G. Krein, M. A. Rutman, Linear operators leaving invariant a cone in a Banach space, Amer. Math. Soc. Translation 10 (3) (1962) 199–325.

[92] E. Bodewig, Matrix Calculus, 1st Edition, Interscience, New York, 1956.

[93] Z. Bai, G. W. Stewart, SRRIT—a FORTRAN subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix, Tech. Rep., University of Maryland at College Park, College Park, MD (1992).

[94] R. Lehoucq, D. C. Sorensen, Deflation techniques for an implicitly re-started Arnoldi iteration, SIAM J. Matrix Anal. Appl 17 (1996) 789–821.

[95] R. B. Lehoucq, D. C. Sorensen, C. Yang, ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Re-started Arnoldi Methods, Society for Industrial & Applied Math., Philadelphia, PA, 1998.

[96] D. C. Sorensen, Deflation for Implicitly Restarted Arnoldi Methods, Tech. Rep., SIAM J. Matrix Anal. Appl (1998).

[97] G. Verdu, R. Miro, D. Ginestar, V. Vidal, The implicit restarted Arnoldi method, an efficient alternative to solve the neutron diffusion equation, Annals of Nuclear Energy 26 (7) (1999) 579–593.

[98] J. S. Warsa, T. A. Wareing, J. E. Morel, J. M. McGhee, R. B. Lehoucq, Krylov subspace iterations for deterministic k-eigenvalue calculations, Nuclear Science and Engineering 147 (2004) 26–42.

[99] E. R. Davidson, The iterative calculation of a few of the lowest Eigenvalues and corresponding Eigenvectors of large real-symmetric matrices, Journal of Computational Physics 17 (1) (1975) 87–94.

[100] H. Voss, A new justification of the Jacobi-Davidson method for large eigenproblems, Linear Algebra and Its Applications 424 (2-3) (2007) 448–455.

[101] Y. Saad, Numerical Methods for Large Eigenvalue Problems, 1st Edition, Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, Manchester, UK, 1992.

[102] K. Wu, Preconditioned techniques for large eigenvalue problems, Ph.D. dissertation, Minneapolis, MN (1997).

[103] Y. Saad, Chebyshev acceleration techniques for solving nonsymmetric Eigenvalue problems, Mathematics of Computation 42 (166) (1984) 567–588.

[104] P. B. Geltner, General Rayleigh quotient iteration, SIAM Journal on Numerical Analysis 18 (5) (1981) 839–843.

[105] A. Dax, The orthogonal Rayleigh quotient iteration (ORQI) method, Linear Algebra and Its Applications 358 (1-3) (2003) 23–43.

[106] D. C. Sorensen, Implicit application of polynomial filters in a k-step Arnoldi method, SIAM J. Matrix Anal. Appl. 13 (1) (1992) 357–385.

[107] R. Miro, D. Ginestar, G. Verdu, D. Hennig, A nodal modal method for the neutron diffusion equation. Application to BWR instabilities analysis, Annals of Nuclear Energy 29 (10) (2002) 1171–1194.

[108] G. Peters, J. H. Wilkinson, Inverse iteration, Ill-conditioned equations and Newton's method, SIAM Review 21 (3) (1979) 339–360.

[109] Y. Zhou, Studies on Jacobi-Davidson, Rayleigh quotient iteration, inverse iteration generalized Davidson and Newton updates, Numerical Linear Algebra with Applications 13 (8) (2006) 621–642.

[110] D. Knoll, R. Park, K. Smith, Application of the Jacobian-free Newton-Krylov method in computational reactor physics, in: Computations and Supercomputing in Nuclear Applications (M&C 2009) International Conference, Saratoga Springs, New York, 2009.

[111] J. J. Duderstadt, G. A. Moses, Inertial Confinement Fusion, Vol. XII, John Wiley & Sons, New York, 1982.

[112] G. L. Olson, L. H. Auer, M. L. Hall, Diffusion, P1, and other approximate forms of radiation transport, Journal of Quantitative Spectroscopy and Radiative Transfer 64 (6) (2000) 619–634.

[113] V. A. Mousseau, D. A. Knoll, W. J. Rider, Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion, Journal of Computational Physics 160 (2) (2000) 743–765.

[114] D. N. Arnold, F. Brezzi, M. Fortin, A stable finite element for the Stokes equations, Calcolo 21 (1984) 337–344.

[115] Argonne Code Center: Benchmark Problem Book, Vol. ANL-7416, Suppl. 2, Argonne National Laboratory, Argonne, IL, 1977.

Appendix A

# MATLAB SCRIPTS FOR MANUFACTURED SOLUTIONS

## A.1 MMS Script for Coupled Conduction/Fluid Problem

```
clear all;  clc;

%          |=========|  |   yMAX
%    |     |   Tf    |  |
%    y     |   2-D   |  Tc (1-D: y)
%    |     |  (x,y)  |  |
%          |=========|  |   0
%          0  --x--  xMAX

syms x y t rF rT Ctf Ctc Tf Tc rho rhoU rhoE mu hc Dh
syms xMAX yMAXw del GAMMA Cp Cv RCONST Qconst G
syms dpdrho dpde kfluid Fw a b c p0 rho0 Tf0 Nu0 mu0

% Heat conduction PDE

% Ctf and Ctc are time constants -> Basically, you could
% make Ctf >> Ctc and this will introduce fast time scales
% due to heat conduction and slower scales due to heat
% removed by fluid.

% Fuel temperature exact solution
Tf = Tf0 + (1+tanh(Ctf*t)) * rF * ((0.5+ sin(pi/2*y/yMAX) )’
      * (1+tanh(w*2/3-w*x/xMAX))) ;

% Coolant temperature exact solution
Tc = (1+tanh(Ctc*t)) * rT * (a-b*tanh(c*w-w*y/yMAX)) ;

% viscosity variation with temperature
mu0 = 1.2075e-006 * subs(subs(Tc, t, 0), y, 0) ;

% Density profile
rhoc = 219 ; f = 275.32 ; g = 511.58 ; Tc0 = 2503.7 ;
rho = rhoc + f*(1-Tc/Tc0) + g*(1-Tc/Tc0).^0.5 ;

inte = Cv * Tc ;

% Velocity constant in space-time
v = G./rho ;

% momentum = mass flux
m = rho .* v ;

% Total energy
e = rho.*inte + 0.5*m.*m./rho ;
```

```
% linearized EOS
p = p0 + dpdrho * (rho-rho0) +
  dpde * (Tc-subs(subs(Tc, t, 0), x, 0)) ;

M = v ./ sqrt(dpdrho) ;

% viscosity variation with temperature
mu = 1.2075e-006 * Tc ;

% http://www.cheresources.com/convection.shtml
Nu = Nu0 * (mu./mu0).^0.14 ;

hc = Nu * kfluid / Dh ; % Heat Transfer Coefficient

Re = G * Dh / mu ; % Reynolds number
Fw = 0.3164 / Re^0.25 ;  % Blasius friction factor

% Thermal conductivity for fuel.
k = 6400.0./Tf^2 * exp(-16.35/Tf) ;

% % % % % % % FORCING FUNCTIONS % % % % % % % % % % %

STf = diff(Tf,t) - diff(k*diff(Tf,x), x)
      - diff(k*diff(Tf,y), y)
      + hc * (subs(Tf,x,xMAX)-Tc) ;

Scont = diff(rho, t) + diff(m, y);

Smom = diff(m, t) + diff(m*v, y) + diff(p, y)
      + Fw * v * abs(v) ;

Sener = diff(e, t) + diff((e+p)*v, y)
```

```
      - hc * (subs(Tf,x,xMAX)-Tc) ;


disp('STf');  ccode(STf)
disp('Scont');  ccode(Scont)
disp('Smom'); ccode(Smom)
disp('Sener');  ccode(Sener)
```

Listing A.1: MMS Script for Coupled Conduction/Fluid Problem

## A.2 MMS Script for Coupled Neutronics/Conduction Problem

```
syms x y z LZ t s T egroups dgroups k avgnu Bg2
syms LX LY CT CF rT rF PI normalization_const
syms DEFAULT_FUEL_TEMP doppler_coeff k0 k1 rho cp
syms beta_tot PHI1 PHI2 xsrem1 xsrem2 xsfiss1 xsfiss2
syms xsrem01 xsrem02 xsnufiss1 xsnufiss2
syms xsdiff1 xsdiff2 energy_per_fission1 energy_per_fission2
syms invvel1 invvel2 xsscatt12 xsscatt21

sx = sin(x/LX*PI) ;
sy = sin(y/LY*PI) ;

T = CT*(1+tanh(rT*t))*sx*sy ;
k = k0 + k1*(T - DEFAULT_FUEL_TEMP) ;

egroups = 2 ;   % energy groups
dgroups = 2 ;   % delayed precursor groups

xsnufiss1 = avgnu * xsfiss1 ;
xsnufiss2 = avgnu * xsfiss2 ;

xsrem1 = xsrem01 + doppler_coeff*(
        sqrt(T) - sqrt(DEFAULT_FUEL_TEMP) ) ;
xsrem2 = xsrem02 ;

beta(1) = sym('beta[1]') ;
beta(2) = sym('beta[2]') ;
lambda(1) = sym('lambda[1]') ;
lambda(2) = sym('lambda[2]') ;
beta_tot = beta(1) + beta(2) ;
```

```
% 2-D geometric buckling
Bg2 = PI*PI*(1/LX^2+1/LY^2) ;
PHI1 = CF*(1+exp(rF*t))*sx*sy*(x/LX*y/LY) ;
PHI2 = PHI1 * xsscatt12 / (xsrem2 + xsdiff2 * Bg2) ;

for dg = 1 : dgroups
  % Initial precursor concentration
    PREC_0(dg) = beta(dg)*(subs(xsnufiss1*PHI1+
              xsnufiss2*PHI2, t, 0))/lambda(dg) ;
  % Analytically solve the precursor ODE
    PREC(dg) = ( PREC_0(dg) + beta(dg)*int(
         subs(
          (xsnufiss1*PHI1+
            xsnufiss2*PHI2)*exp(lambda(dg)*s), t, s),
          s, 0, t) ) * exp(-lambda(dg)*t) ;
end

% FORCING FUNCTIONS
% Fuel Temperature
srcT = rho*cp*diff(T, t) - diff(k*diff(T,x),x) -
    diff(k*diff(T,y),y) - normalization_const*
    (energy_per_fission1*xsfiss1*PHI1+
      energy_per_fission2*xsfiss2*PHI2) ;

% Fast Flux
srcFLX1 = invvel1*diff(PHI1,t) - diff(xsdiff1*diff(PHI1,x),x)
      - diff(xsdiff1*diff(PHI1,y),y) + xsrem1*PHI1
      - (1-beta_tot)*(xsnufiss1*PHI1+xsnufiss2*PHI2)
      - xsscatt21*PHI2 ;
if dgroups > 0
    for dg = 1 : dgroups
         srcFLX1 = srcFLX1 - lambda(dg)*PREC(dg) ;
```

```
          srcPREC(dg) =  diff(PREC(dg),t) -
      beta(dg)*(xsnufiss1*PHI1+xsnufiss2*PHI2)
      + lambda(dg)*PREC(dg) ;
    end
end
% Thermal Flux
srcFLX2 = invvel2*diff(PHI2,t) - diff(xsdiff2*diff(PHI2,x),x)
      -diff(xsdiff2*diff(PHI2,y),y) + xsrem2*PHI2
      - xsscatt12*PHI1 ;

Power = normalization_const*(energy_per_fission1*xsfiss1*PHI1
      + energy_per_fission2*xsfiss2*PHI2) ;

% Total power in the domain as a function of time
totPower = int (int (Power, y, 0, LY), x, 0, LX) ;

codeT = ccode(T)
codeFLX1 = ccode(PHI1)
codeFLX2 = ccode(PHI2)
for dg = 1 : dgroups
    fprintf('\ncodePREC{%d} = \n\n', dg) ;
    disp(ccode(PREC(dg)))
end

codeST = ccode(srcT)
codeSFLX1 = ccode(srcFLX1)
codeSFLX2 = ccode(srcFLX2)

for dg = 1 : dgroups
    codeSPREC{dg} = ccode(srcPREC(dg)) ;
    fprintf('\ncodeSPREC{%d} = \n\n', dg) ;
    disp(codeSPREC{dg})
end
```

```
codePower = ccode(totPower)
```

Listing A.2: MMS Script for Coupled Neutronics/Conduction Problem

# Appendix B

# CROSS-SECTION DATA FOR EIGENVALUE PROBLEMS

## B.1 2-D Two-group IAEA Benchmark Problem

This problem is based on the benchmark introduced in ANL [115] and contains 4 materials. The cross-sections are obtained from the 3D data by accounting for leakage in z-direction with a $Bg_z^2 = 8 \times 10^{-5}$.

The lattice configuration is of the form given below with each assembly having dimensions = [10, 10] cms.

3 1 1 1 1 1 1 3 3 1 1 1 1 2 2 4 4

1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 4 4

1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 4 4

1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 4 4

1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 4 4

1 1 1 1 1 1 1 1 1 1 1 2 2 4 4 4 4

1 1 1 1 1 1 1 1 1 1 1 2 2 4 4 4 4

```
3 1 1 1 1 1 1 3 3 2 2 2 2 2 4 4 0 0
3 1 1 1 1 1 1 3 3 2 2 2 2 2 4 4 0 0
1 1 1 1 1 1 1 2 2 2 2 2 4 4 4 4 0 0
1 1 1 1 1 1 1 2 2 2 2 2 4 4 4 4 0 0
1 1 1 2 2 2 2 2 2 2 4 4 4 4 0 0 0 0
1 1 1 2 2 2 2 2 2 2 4 4 4 4 0 0 0 0
2 2 2 2 2 4 4 4 4 4 4 4 0 0 0 0 0 0
2 2 2 2 2 4 4 4 4 4 4 4 0 0 0 0 0 0
4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0
```

## B.2 Cross-section Data for 2-D Two-group Homogenous Medium Problem

$D_1 = 1.0, D_2 = 0.4$

$\Sigma_{r,1} = 0.04, \Sigma_{r,2} = 0.08$

$\nu\Sigma_{f,1} = 0.01, \nu\Sigma_{f,2} = 0.13$

$\Sigma_{s,1\longrightarrow 2} = 0.02, \Sigma_{s,2\longrightarrow 1} = 0.001$

# Part II

# University of Chicago/ Argonne National Laboratory's contribution

# Enabling High-Fidelity Neutron Transport Simulations on Petascale Architectures

**SUMMARY**
The UNIC code is being developed as part of the DOE's Nuclear Energy Advanced Modeling and Simulation (NEAMS) program. UNIC is an unstructured, deterministic neutron transport code that allows a highly detailed description of a nuclear reactor. The primary goal of our simulation efforts is to reduce the uncertainties and biases in reactor design calculations by progressively replacing existing multilevel averaging (homogenization) techniques with more direct solution methods based on first principles. Since the neutron transport equation is seven dimensional (three in space, two in angle, one in energy, and one in time), these simulations are among the most memory and computationally intensive in all of computational science. In order to model the complex physics of a reactor core, billions of spatial elements, hundreds of angles, and thousands of energy groups are necessary, leading to problem sizes with petascale degrees of freedom. Therefore, these calculations exhaust memory resources on current and even next-generation architectures. In this paper, we present UNIC simulation results for two important representative problems in reactor design and analysis—PHENIX and ZPR-6. In each case, UNIC shows good weak scalability on up to 163,840 cores of Blue Gene/P (Argonne) and 122,800 cores of XT5 (Oak Ridge). While our current per processor performance is less than ideal, we demonstrate a clear ability to effectively utilize the leadership computing platforms. Over the coming months, we aim to improve the per processor performance while maintaining the high parallel efficiency by employing better algorithms such as spatial *p*- and *h*-multigrid preconditioners, optimized matrix-tensor operations, and weighted partitioning for better load balancing. Combining these additional algorithmic improvements with the availability of larger parallel machines should allow us to realize our long-term goal of explicit geometry coupled multiphysics reactor simulations. In the long run, these high-fidelity simulations will be able to replace expensive mockup experiments and reduce the uncertainty in crucial reactor design and operational parameters.

**Keywords**
Neutron transport, discrete ordinates, nuclear reactors, parallel scalability, memory bandwidth.

## 1. INTRODUCTION

Nuclear engineering has a rich history of simulation-based design following sound economical and safety-driven principles. However, many of the modern reactor modeling codes were developed in the 1970s and 1980s and targeted serial platforms using homogenization techniques because of the high computational costs of explicit geometry approximations. In this paper, we describe the development of a new reactor analysis code that bridges the gap between the approximation-based legacy tools and a first-principles approach. The code we discuss in this paper is specifically targeted at applications for which the legacy tools are least reliable, and its development is possible only on the large scale parallel machines.

The performance of nuclear power reactors is governed by the fission rate of the nuclear fuel. A predictive analysis capability generally is required to optimize the safety characteristics of the reactor and minimize the costs associated with operating the reactor. This analysis capability is derived from the solution of a Boltzmann integro-differential transport equation for the neutron density. This equation is widely used in atmospheric modeling, astrophysical research, nuclear weapons research, medical physics, and industrial applications such as mineral assaying and oil-well logging. Among these fields, the most significant parallelization efforts to date have been applied to structured geometry solvers for the thermal radiative transport equation (gamma and x-rays) used in weapons related research, and researchers have utilized several "top" supercomputers to perform simulations. Unfortunately, many of the modeling challenges that arise in the thermal radiative and the neutron transport equations are sufficiently different that direct technology transferability between the codes is impractical.

The primary unknown in the Boltzmann transport equation is the neutron density or, in nuclear engineering vernacular, the neutron "flux" (density multiplied by velocity). The equation has seven independent variables: three in space, two in angle, one in energy, and one in time. Because an accurate, first-principles discretization of these variables is untenable, legacy solvers are typically based on approximations that reduce this dimensionality. The neutron transport equation can also be shown to asymptotically limit to the canonical hyperbolic, elliptic, and parabolic partial differential equation forms under simple changes in material properties that may occur in a nuclear reactor. In thick, highly-scattering regions, the transport equation limits to a (parabolic) time-dependent diffusion equation, which, in steady state, is elliptic. In "free-streaming" regions (a standard characteristic of research reactors), the limiting behavior is hyperbolic. Thus, the large dimensionality and many-faceted solution behaviors for this equation present the greatest challenges to the code developer.

Our targeted research is the immediate improvement to areas where legacy solvers are insufficient: nuclear reactor dynamics. These problems require the solution of the time-dependent Boltzmann transport equation and the simultaneous solution of the thermal-hydraulic and structural-mechanics equations [1]. Two years ago we started a multiyear development project to create a dynamics solver capability using the open-science high performance computing resources at Argonne National Laboratory (IBM Blue Gene/P) and Oak Ridge National Laboratory (Cray XT5) [2-4]. The initial condition for this formulation requires the solution of a time-independent k-eigenvalue equation [1] that is the focus of this manuscript. We note that with a linear implicit time formulation, all subsequent solutions at the end of each time step exhibit very similar requirements to that needed to solve the initial k-eigenvalue problem. In this paper, we focus on the recent success of the SN2ND solver from UNIC [5], which solves the steady state second-order even-parity formulation of the neutron transport equation.

## 2. NEUTRON TRANSPORT SIMULATION COMPLEXITY

The primary issue in nuclear reactor analysis is the shear scale of the problem to be solved. We have thus far limited our dynamics solver development to fast reactor designs, since these reactors have been proposed as an alternative to reduce the volume of spent fuel disposition (i.e. fission the high actinides rather than store them indefinitely) and the nuclear industry has

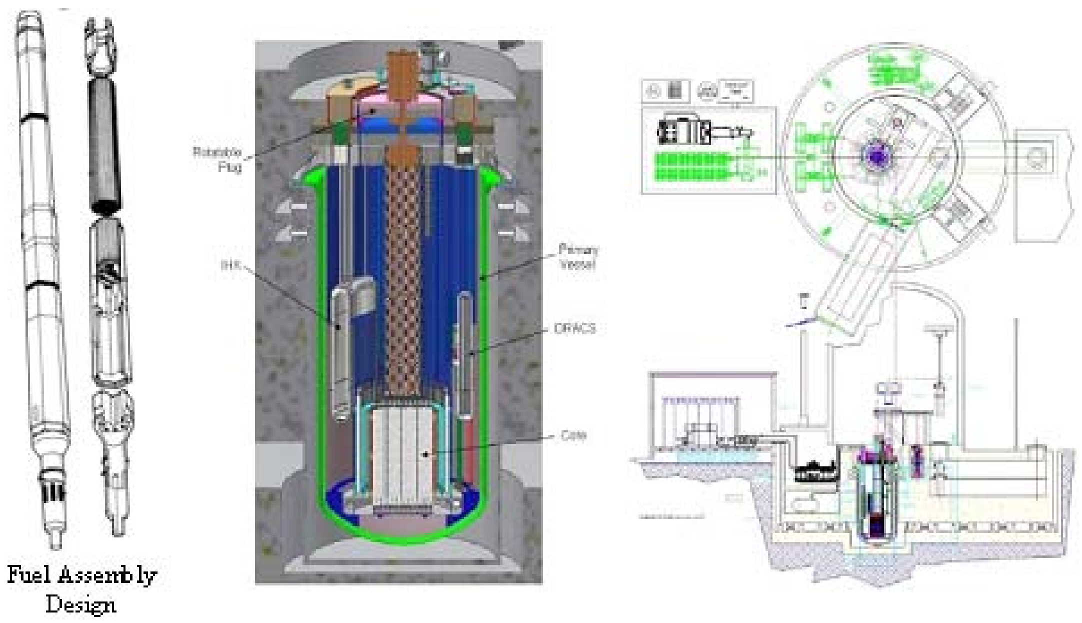


**Figure 1. Fuel Assembly, Reactor, and Plant Schematics for a Sample Fast Reactor**

insufficient engineering experience. With time, we will also apply our reactor analysis tools to more prevalent light water reactors (LWRs), very high temperature gas cooled reactors (VHTRs), and the Canadian deuterium uranium (CANDU) reactors [6] to help address the smaller pool of unanswered questions that these larger, thermal reactor designs pose.

We begin with the spatial domain of a typical sodium-cooled fast reactor, some examples of which are shown in Figure 1. For neutron transport, we may limit our focus to just the “core” of the reactor shown in the center picture of Figure 1, the scale of which, relative to the plant, can be inferred from the rightmost picture in Figure 1. The core typically comprises about 200-500 ducted fuel assemblies similar to those depicted in the leftmost picture of Figure 1.

The assemblies are composed of many (60-300) fuel pins depending on the fuel cycle performance targets. Radially, the core is built of fuel assemblies that form a rough cylinder, leading to a total modeling diameter of two to six meters and a height of three to five meters. This core size and the spatial heterogeneity of the fuel assemblies require approximately half a billion to a billion finite elements to accurately represent the geometry and the associated spatial gradients in the neutron density.

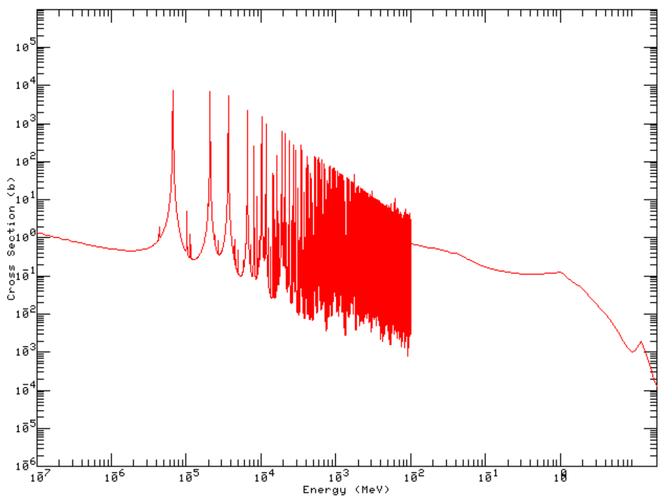
Next we consider the energy and angular requirements because they are tightly coupled. Neutrons lose fractions of their energy by scattering with materials, and the amount of energy loss per scattering depends on the scattering material and collision angles. Figure 2 shows the highly varying “cross section” data (roughly, the probabilities of interaction versus neutron energy) for the Uranium-238 and Iron-56 isotopes that constitute two of the largest components of a sodium cooled fast reactor. Note that these are on a log-log scale. Most other isotopes present in a nuclear reactor have cross section representations of similar complexity.

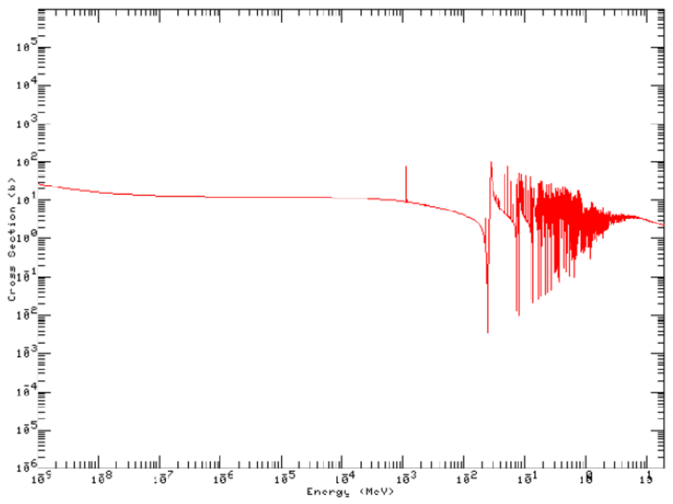
The large amount of material heterogeneity in the geometry, combined with the severity of the energy dependence in the cross section data, leads to flux distributions of comparable complexity to the space-energy distribution of the cross section data. Hence, it is impractical to use a smooth polynomial functional representation in energy, and all historical and modern energy discretizations employ a “multigroup” (zeroth-order finite element in energy) flux representation [1] by utilizing “effective” multigroup constants. Additionally, because neutron scattering couples the energy and angle terms, with the rapid changes in the energy dependence of the cross sections seen in Figure 2 come rapid variations in the magnitude of the flux in the angular variable. We estimate that a first-principles approach will require 100,000 energy groups and 1,000 angles (collocation or zeroth-order finite elements on the sphere), which leads to approximately $10^{17}$ degrees of freedom in space, energy, and angle for each time step. Thus, even on today’s supercomputers, some form of approximation is necessary to obtain solutions.

## 3. NEUTRON TRANSPORT FOR REACTOR ANALYSIS

Fortunately, most engineering analyses permit simplifying approximations of the explicitly discretized equation. The first and most important simplification reduces the demands of the energy representation. This requires several multilevel modeling and simplification steps, the details of which are beyond the scope of this paper, whose purpose is to produce a set of coarse group cross section data that preserves key neutron reaction rates in each energy group [1]. These approximations rely upon substantial experience on particular reactor systems and experimental validations of the predictive abilities of the legacy tools.

As we develop our code, the significance of these approximations will diminish relative to the legacy approaches, since we will enable a better matching of the “reference” configuration used to generate the coarse group cross section data to the system at hand. The end result is that we reduce the need for 100,000 groups to much less than 2,000 groups, with a general ability to use fewer than 100 groups for most analyses (2 groups are typically used in most industry reactor analysis codes). In this work, we use a 33-group approximation that pushes our current solver to the limits of the available memory on Blue Gene/P.





**Capture Cross Section for U-238** **Total Cross Section for Fe-56**

**Figure 2. Sample Cross Section Data for U-238 and Fe-56**

We next consider the time discretization. The average speed and multiplication time of neutrons in a reactor core are such that time steps on the order of milliseconds are needed for rapid transient scenarios such as a control rod ejection, making the neutronics component the “stiff” part of the overall multi-physics system. The duration of the simulated transient varies from hours to days, which makes the time spent in the neutron transport solver the limiting simulation factor. Most modern legacy tools for fast reactor analysis avoid this problem by using a point kinetics (space-angle-energy independent) model or a few energy-group diffusion theory methodology on a structured geometry grid [1]. With the improvements in the energy approximation and the use of transport rather than diffusion theory, we expect to significantly improve the accuracy and fidelity of the safety analysis modeling for these simulations.

Unstructured mesh deterministic methods use the multigroup approximation in energy combined with either a hybrid finite element or a continuous, finite element decomposition in space. Historical angle discretization schemes include spherical harmonic (polynomial) expansions, finite element, or angle collocation (zeroth order finite element, also known as discrete ordinates) [1]. To date, most parallelization efforts in neutron transport have focused on improvement of structured geometry discrete ordinates solvers [7-9] with moderate to good success on small to medium-range parallel machines, although it is difficult to find performance data for these tools on more than 2,000 processors. Unfortunately, these tools are not useful for simulating coupled multiphysics phenomena inside the complex reactor geometries. While some experts might argue that we can impose pin-cell level or the often termed “lego landing” homogenization approaches to capture the spatial heterogeneity [18], these approaches can quickly become unmanageable in terms of computational effort and really just substitute one problematic legacy approach with another (albeit better) one. Even if we were to take such an approach, it would seem wiser to just use the legacy tools based on assembly homogenization [10], since those tools can easily execute on serial platforms and provide comparable if not superior solutions. In addition to our own work, there has been substantial research on unstructured methodologies [11-13], but these codes are not obtainable, not set up for the specific needs of reactor analysis, or not demonstrated to work for large, heterogeneous geometry applications and scale well on contemporary parallel platforms (soon expected to have millions of cores).

We note that parallelization of the Monte Carlo method for has been particularly effective because of its "embarrassingly parallel" characteristics. However, the dynamics problems that we are targeting include massive memory requirements that prevent each processor from accessing the full space-energy representation of the problem in the Monte Carlo method. The standard proposed remedy is to use domain decomposition in the Monte Carlo algorithm, but this severely impacts its scalability. Moreover, Monte Carlo solutions can also contain stochastic uncertainties on the order of the expected perturbations from the thermal-structural feedback effects, which makes the Monte Carlo methods even more impractical. Thus, a massively parallel deterministic solver for dynamics problems truly fills a gap in the available predictive capabilities of modern neutron transport tools.

## 4. UNIC: MODERNIZATION AND DEVELOPMENT OF REACTOR ANALYSIS TOOLS

In this section we present an overview of our solution algorithm to the time-independent (the initial condition) neutron transport equation. We also discuss algorithmic choices we have made to reduce execution times by cutting down on extraneous floating point work while maintaining good parallel scalability.

### 4.1 Neutron Transport Equation

The multigroup form of the neutron transport equation consists of *G* equations with 1<*g*<*G*:

$$\hat{\Omega}\cdot\vec{\nabla}\psi_g(\vec{r},\hat{\Omega})+\Sigma_{t,g}(\vec{r})\psi_g(\vec{r},\hat{\Omega})=S_g(\vec{r},\hat{\Omega}) \ . \qquad (1)$$

Here, $\psi_g(\vec{r},\hat{\Omega})$ is the group neutron angular flux and $\Sigma_{t,g}(\vec{r})$ is the total cross section (sum of all reaction probabilities). Thus, the first term is a streaming/leakage term, and the second is a collision removal term. The system of equations is coupled via the source $S(\vec{r},\hat{\Omega})$ which we expand in terms of group-to-group scattering and fission as

$$\begin{aligned} S_g(\vec{r},\hat{\Omega}) &= \sum_{g'=1}^{G}\int \Sigma_{s,g'\to g}(\vec{r},\hat{\Omega}\cdot\hat{\Omega}')\psi_{g'}(\vec{r},\hat{\Omega}')d\hat{\Omega}' \\ &+\frac{1}{k}\chi_g\sum_{g'=1}^{G}\nu_{g'}(\vec{r})\Sigma_{f,g'}(\vec{r})\int\psi_{g'}(\vec{r},\hat{\Omega}')d\hat{\Omega}' \end{aligned} \ . \qquad (2)$$

Here, *k* is the dominant steady state system eigenvalue, also known as its effective multiplication factor, $k_{eff}$. The scattering source in Eq. (2) redistributes neutron energies and angles in an anisotropic way, while the fission source redistributes neutrons into the isotropic fission energy spectrum $\chi$.

Based on the parallelization successes of other authors with the Poisson equation, we focused part of our initial development on second-order methodologies that implement continuous, spatial finite element approximations such that we can take advantage of parallel conjugate gradient methods. To obtain the second-order discrete ordinates formulation used in SN2ND, we expand the angular flux in Eq. (1) into even-parity [$\psi_g^+(\vec{r},\hat{\Omega})$] and odd-parity [$\psi_g^-(\vec{r},\hat{\Omega})$] components:

$$\psi_g(\vec{r},\hat{\Omega})=\psi_g^+(\vec{r},\hat{\Omega})+\psi_g^-(\vec{r},\hat{\Omega}) \ . \qquad (3)$$

We then rewrite Eq. (1) using Eq. (3) to obtain coupled, first-order even-parity and odd-parity equations:

$$\hat{\Omega}\cdot\vec{\nabla}\psi_g^{\pm}(\vec{r},\hat{\Omega})+\Sigma_{t,g}(\vec{r})\psi_g^{\mp}(\vec{r},\hat{\Omega})=S_g^{\pm}(\vec{r},\hat{\Omega}) \ . \qquad (4)$$

We next solve for the odd-parity flux in terms of the even-parity flux and substitute it into the even-parity equation to obtain the second-order even-parity transport equation.

$$\begin{aligned}&-\hat{\Omega}\cdot\vec{\nabla}\frac{1}{\Sigma_{t,g}(\vec{r})}\hat{\Omega}\cdot\vec{\nabla}\psi_g^{+}(\vec{r},\hat{\Omega})+\Sigma_{t,g}(\vec{r})\psi_g^{+}(\vec{r},\hat{\Omega})=\\ &S_g^{+}(\vec{r},\hat{\Omega})-\hat{\Omega}\cdot\vec{\nabla}\frac{1}{\Sigma_{t,g}(\vec{r})}S_g^{-}(\vec{r},\hat{\Omega})\end{aligned}\qquad(5)$$

We weight Eq. (5) with a set of spatial trial functions $f(\vec{r})$, integrate over volume, and apply the divergence theorem to the first term such that we obtain the natural vacuum boundary condition term on the surfaces of the domain boundary.

$$\begin{aligned}&\int dV\,\hat{\Omega}\cdot\vec{\nabla}f(\vec{r})\frac{1}{\Sigma_{t,g}(\vec{r})}\hat{\Omega}\cdot\vec{\nabla}\psi_g^{+}(\vec{r},\hat{\Omega})\\ &+\int dV\;f(\vec{r})\Sigma_{t,g}(\vec{r})\psi_g^{+}(\vec{r},\hat{\Omega})\\ &+\int d\Gamma\;f(\vec{r}\in\Gamma)\left|\hat{\Omega}\cdot\hat{n}\right|\psi_g^{+}(\vec{r}\in\Gamma,\hat{\Omega})=\int dV\;f(\vec{r})Q_g^{+}(\vec{r},\hat{\Omega})\end{aligned}\qquad(6)$$

Finally, we implement a continuous finite element formulation for the even-parity flux (allowing for a discontinuous odd-parity flux as a secondary unknown). The coefficient matrix produced from the terms on the left side of Eq. (6) can be shown to be symmetric positive definite and thus suitable for the conjugate gradient (CG) methodology, provided that the terms on the right side of Eq. (6) are "lagged" in an iterative approach to be discussed below.

### 4.2 SN2ND Solver Implementation

The spatial approximation is treated by using a standard continuous finite element method, and we employ classic domain decomposition in which weights are applied to the vertices to balance the local work with the communication costs required to connect the domain. For the angular variable, we chose the discrete ordinates approximation, which requires us to define a set of directions on the unit sphere. With regard to parallelization, we employ the generic decomposition of (S)pace, (A)ngle, and (G)roup shown in Figure 3.

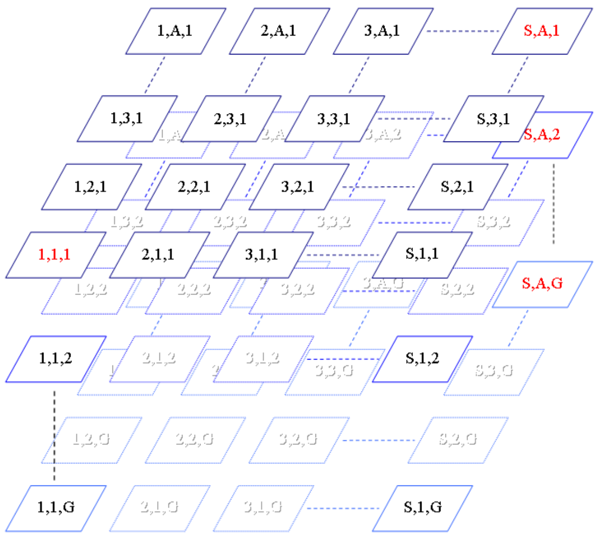


**Figure 3. Space, Angle, Group Decomposition for a Parallel Machine.**

In this approach, each MPI process sees four communicators: space, angle, group, and the global communicator. The advantage of this approach is that the group and angle communication does not overlap with respect to space, and thus the communication in these two directions can be done simultaneously on the parallel machine.

When a discrete ordinates approximation is applied to Eq. (6), we find that, for each group, the set of angular equations are coupled only via the "within-group" source term on the right side of Eq. (6). This "within-group" equation is typically solved by using Richardson iteration, termed as "scattering iteration" in the literature, where the within-group scattering source is lagged in iteration [1,2]. The iterations are accelerated by solving a synthetic diffusion equation for the angle-integrated (scalar) flux, which is essentially a multigrid preconditioner in angle [5]. Thus, a "scattering iteration" of SN2ND involves solving 100 diffusion-like equations (assuming 100 directions in the angular cubature) simultaneously to obtain the angular discrete ordinates flux for each group. These equations are currently solved by using a parallel SSOR-preconditioned CG methodology available in PETSc [17]. Although we are developing a customized multigrid preconditioner for this solver, we note that SSOR is the only preconditioner that has worked reliably for our problems and has least memory overhead. As stated earlier, SN2ND code is solving problem sizes that are at the limit of memory available on the largest contemporary machines. Therefore, SSOR is a careful choice after systematic evaluation of several preconditioners. In our custom multigrid implementation, SSOR will likely be used at the coarse level. To update the source (or perform a synthetic acceleration step) on the right side of Eq. (6), we collect the information on the angular communicator of each process. This requires a global reduce operation for the locally visible spatial mesh partition for each group (simultaneous communication on group and space communicators if fully partitioned in energy).

In our current implementation, we do not consider parallelization by group because we can already saturate the available parallel machines with our space-angle parallelization scheme. However, this approach means our memory requirements are linear with respect to the number of energy groups, which can be problematic on low-memory machines like Blue Gene/P [3]. In our current solver, we can distribute any number of angles on a given process and generally have found that two to three angles per process works best. With regard to solving the synthetic acceleration equation, we have currently assigned the first process on each angular communicator to again utilize the parallel SSOR-preconditioned CG algorithm in PETSc, which introduces a load imbalance by angle parallelization. With time we expect to redistribute this work on a subset of the processors on the angle communicator.

The steady state transport equation shown in Eqs. (1) and (2) requires an eigenvalue search procedure to obtain *k* and the associated flux vector. The gold standard for all modern neutron transport codes is to use inverse power iteration [1] because it minimizes the effort required to find the dominant eigenvalue. Assembling all of the group and direction equations derived from Eq. (6), we write

$$\begin{aligned} A\psi^{+} = B\psi^{+} + \frac{1}{k}F\psi^{+} \quad &\rightarrow \quad \psi^{+} = \frac{1}{k}T^{-1}F\psi^{+} \\ &\rightarrow \quad F\psi^{+} = \theta = \frac{1}{k}FT^{-1}\theta \end{aligned} \quad , (7)$$

where *A* is the coefficient matrix, *B* is the scattering source operator, and *F* is the fission source operator. The power (or outer) iteration methodology finds the dominant *k* eigenvalue using the following recurrence relationships:

$$\theta^{(i)} = \frac{1}{k^{(i-1)}} F\, T^{-1} \theta^{(i-1)}, \qquad k^{(i)} = k^{(i-1)} \frac{\left\langle w, \theta^{(i)} \right\rangle}{\left\langle w, \theta^{(i-1)} \right\rangle}. \tag{8}$$

In SN2ND, we currently use the Gauss-Seidel method to iteratively invert *T* during each outer iteration in Eq. (8), because for fast reactors only a single iteration is required for convergence (neutrons lose energy only during scattering events over the energy range of interest in fast reactors, the energy coupling is lower triangular). For time-dependent problems and thermal reactor calculations, a Gauss-Seidel scheme is less efficient, and we intend to use a more general Krylov method with our current Gauss-Seidel scheme as a preconditioner. We note that a Krylov solver will also assist in making the above methodology scalable in energy for time-dependent problems.

Fundamentally, this approach does not require *T* to be exactly inverted during each outer iteration. Instead, we require only that the error in the flux vector in Eq. (7) be slightly lower than the error in the fission source vector. We implemented an optimized scheme to account for this behavior and combined it with conventional Tchebychev acceleration [1,2]. Together, these approaches have allowed us to significantly reduce the overall time to solution. Figure 4 shows the impact of making these optimization changes on the C5G7 benchmark [18] where the outer iteration eigenvalue, fission source, and flux vector are plotted in addition to the within-group flux error for each energy group. In Figure 4, the un-accelerated approach takes roughly twice the number of outer iterations as the Tchebychev accelerated one. More important, the effort spent on solving the within-group flux equations for each outer iteration is substantially reduced in the optimized version (the targeted flux error obtained for each group flux at each iteration is relaxed). In practice, the effort spent on each outer iteration is nearly constant, although the dynamic error adjustments can introduce variability in the total solution time from problem to problem.

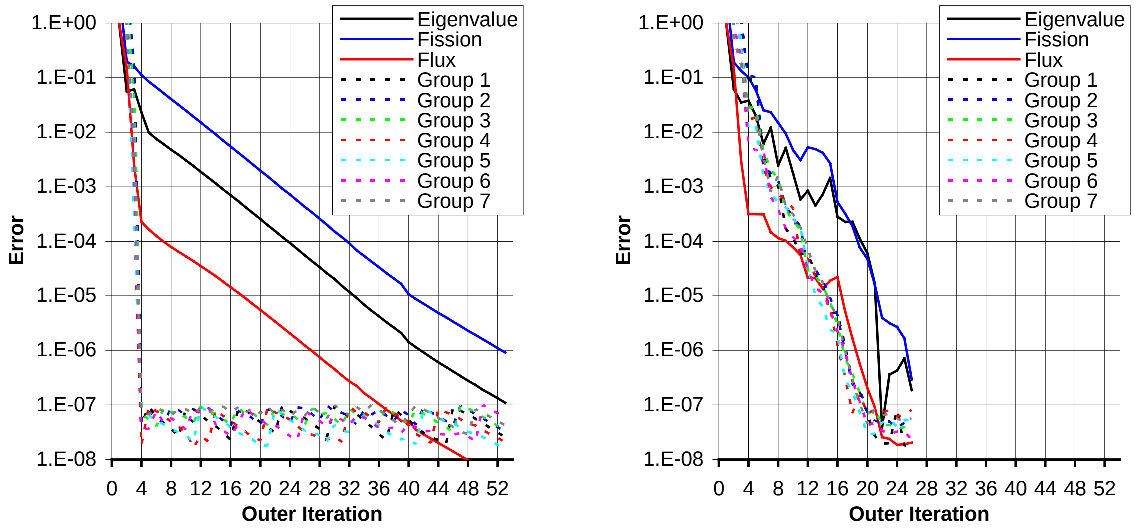


**Figure 4.Optimization and Tchebychev Acceleration Impact on the C5G7 Benchmark.**

## 5. PROBLEMS CHOSEN FOR STUDY

Two reactor problems have been chosen to demonstrate the performance of the SN2ND solver. Both problems consider the steady state eigenvalue solution, the initial condition for the time-dependent problems that we will be studying in the near future. Also, both problems cannot be well solved by using existing homogenization methodologies hence our desire to use UNIC.

## 5.1 PHENIX End-of-Life Experiments

The first problem is taken from the end-of-life experiments of the PHENIX reactor [19]. A solution using UNIC is desired because the legacy solvers (based on conventional homogeneous assembly approaches) have difficulty in representing the control rod configurations accurately. In this benchmark, only the control rod assemblies are represented heterogeneously. This mixed spatial representation is relevant in that our initial time-dependent calculations will also focus on representing only part of the geometry heterogeneously. Figure 5 depicts a slice of the PHENIX core center along with a typical unstructured mesh (prisms) and the flux solution at two important energy groups in the lower part of the control rod assembly created using VISIT [20]. We note that the solutions obtained with SN2ND are, to the best of our knowledge, the most reliable means of obtaining the correct solution compared with all other modern deterministic solution methods.

The benchmark authors requested numerous calculations regarding the control rod positioning. However, there is insufficient information in the benchmark's geometry description to create an accurate heterogeneous representation of the control rods and other reactor components, and we have limited access to more detailed information (proprietary). Consequently, we have constructed a "best-guess" model for which we must obtain our own reference eigenvalue solutions by continuous energy Monte Carlo calculations. Unfortunately we do not expect to complete these calculations until September 2009. Despite the difficulties encountered in using the PHENIX reactor as a verification problem, the performance data obtained in April 2009 is relevant in the context of this manuscript. Timing constraints have prevented us from getting the performance data from the latest version of our code (which has several algorithmic enhancements detailed in Figure 6).

For our previous calculations we used a 33-group cross section set with a $P_3$ expansion of the scattering kernel, which is generally accurate for homogenous problems. Using CUBIT [21], we created meshes considering different degrees of radial mesh refinement (three levels) and axial mesh refinement (three levels), leading to a total of nine meshes. Our simulations demonstrated that the medium-level approach for both the radial and axial directions was sufficiently accurate. This mesh contains 284,682 quadratic Lagrangian prismatic elements and 1,741,833 spatial vertices.

In Table 1, we present the weak scaling results we achieved using SN2ND on Blue Gene/P. Using MeTiS [22], we partitioned the mesh into 2,048 pieces leading to ~850 vertices per process, which is near the minimum that we can use with the parallel SSOR-preconditioned CG algorithm in PETSc (below this, communication overhead increases substantially on both machines). As we increase the number of angles (note that the even-parity formulation requires only the half-sphere set of angles or $2\pi$), we make a corresponding increase in the number of processors. As can be seen, the eigenvalue rapidly converges as the number of directions is increased, which is expected given that a majority of the domain is homogenized. An initial glance indicates a drop in weak scaling to 75% on the entire machine; however, the number of "fission" (outer) iterations needed to solve Eq. (8) is correspondingly seen to increase as well.

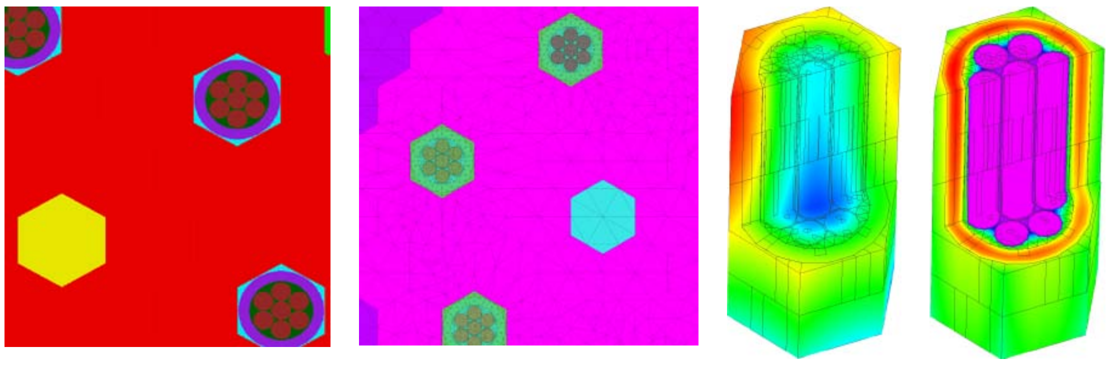


**Figure 5. Planar Configuration of PHENIX Geometry Model and Flux Solution**

**Table 1. Weak Scaling Study by Angle for the PHENIX Problem on Blue Gene/P**

| Cores | $4\pi$ Angles | $k_{eff}$ | Fission Iters. / Time | Total Time (sec) | Source Update (sec) | Weak Scaling |
|---|---|---|---|---|---|---|
| 32,768 | 32 | 0.96006 | 23 / 152 | 3493 | 2934 | 100% |
| 49,152 | 48 | 0.96004 | 23 / 152 | 3510 | 2933 | 100% |
| 65,536 | 64 | 0.96007 | 23 / 153 | 3526 | 2934 | 99% |
| 73,728 | 72 | 0.96015 | 23 / 156 | 3593 | 2934 | 97% |
| 131,072 | 128 | 0.96019 | 27 / 156 | 4209 | 3437 | 83%* |
| 163,840 | 160 | 0.96019 | 27 / 173 | 4676 | 3436 | 75%* |

Although the number of fission source iterations can vary depending on the space-angle-energy discretization and the dynamic error scheme shown in Figure 4, these “top-level” aspects of the code are not part of the parallelization scheme and should not be impacted significantly by changing the angular or spatial approximation. Upon further investigation we identified a problem in the within-group scattering source iteration algorithm, corrected it, and thus removed the superfluous outer iterations. To gauge the impact of our recent algorithmic improvements to SN2ND, we revisited the 65,536 processor calculation on Blue Gene/P from Table 1. We now obtain the same solution in a total time of 1366 seconds with 631 seconds in the source update routine (or 39% of the time reported in Table 1) using 24 outer iterations. Figure 6 shows the more detailed impact of the improvements made to SN2ND on another example problem (discussed next). These reductions in computational effort are not derived from machine-specific optimization or from implementing multigrid (which we will not have results for until September); they are due solely to our efforts to improve those parts of the algorithm that were highlighted as poorly implemented in April 2009. With time we expect many more changes – in addition to the proposed *p*- and *h*-multigrid preconditioner schemes – all of which should further reduce the time to solution such that the multi-physics dynamics calculations we proposed will become feasible.

## 5.2 Zero Power Reactor 6 Experiment 6A

The other problems we chose to simulate for our project this year are the Zero Power Reactor (ZPR) experiments 6a and 7 [23]. The ZPR-6 experiments were performed to acquire fundamental data on nuclear reactor designs of interest. These particular experiments focused on uranium-fueled (ZPR-6/6a) and plutonium-fueled (ZPR-6/7) sodium-cooled fast reactor systems.

The SN2ND simulations of these experiments will be used to help validate the code and to better ascertain the approximation errors in legacy approaches by enabling direct comparisons of computed results. While the experiment ZPR-6/7 has more data and is our preference, we have begun with the 6a experiment this year because of its simplicity. Figure 7 provides two pictures of the explicit geometry model (left, center). The gray color in Figure 7 is used for the matrix tube and drawer fronts that are loaded into each tube position. The solid green squares are two-inch depleted uranium metal blocks directly loaded into the tubes surrounding the main core and act as a neutron blanket. We separated the matrix assemblies, withdrew one of the drawers from the front matrix assembly, and pulled a section of the plates out to give a better perspective on the overall geometry. We also provided a plot of the enriched uranium plate power (the other plate power contribution is minor) on the right of Figure 7, where we have again separated the matrix halves.

Our initial calculations focused on scoping studies to identify any immediate areas within SN2ND and the meshing tools that had to be fixed before attempting the fully explicit calculation. Accordingly, we reduced the number of unique fuel drawer types and introduced simple asymmetries into the geometry to investigate local flux heterogeneities (note we removed several fuel drawers reducing $k_{eff}$ below 1). While our more recent work has focused on solving the explicit experiment, we encountered severe meshing difficulties within CUBIT that cannot be overcome at this point.

In general, the exact geometry of a ZPR-6 experiment is difficult to solve with either SN2ND or a legacy structured geometry solver because of the large number of material boundaries, as indicated by the left picture in Figure 8. Furthermore, for even-parity methods such as SN2ND, the extremely small voids separating the plates and various other components can not be explicitly incorporated, and we must make some type of geometric simplification (homogenization), as shown on the right of Figure 8.

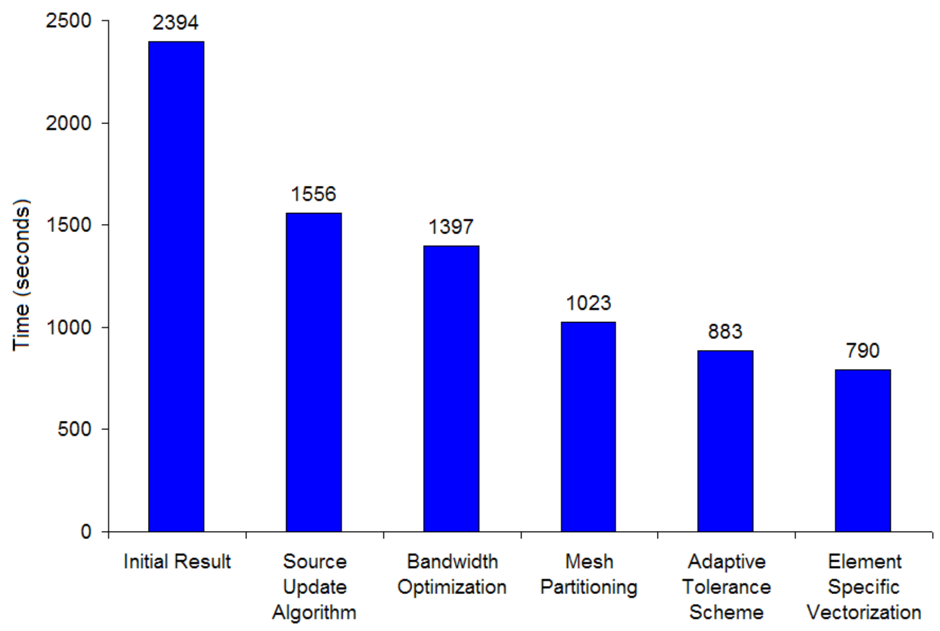


**Figure 6. Impact of Various Algorithmic Improvements to SN2ND Since the Original Version of This Paper (April 2009) for the ZPR 6/6a Problem in Table 3 on 16,384 Cores of Blue Gene/P.**

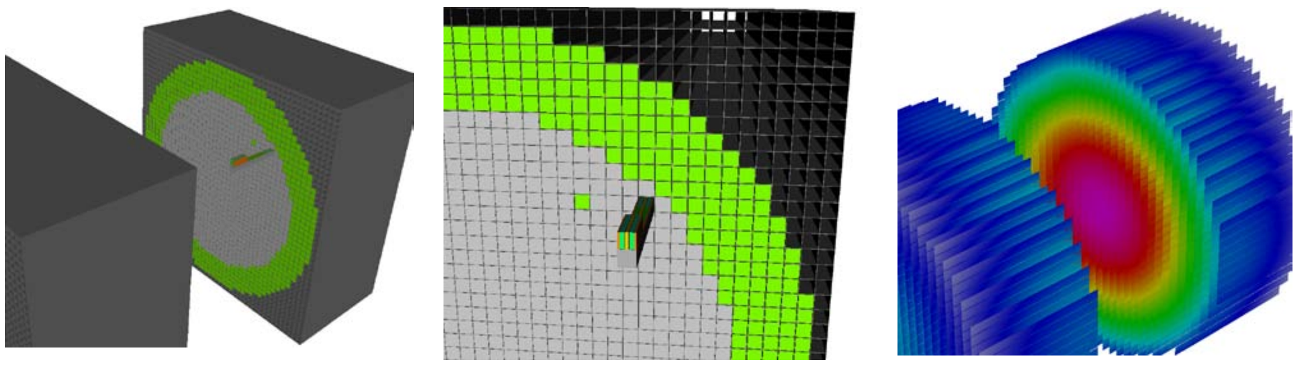
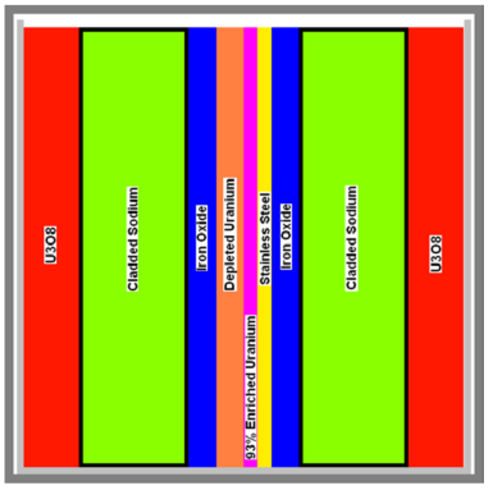
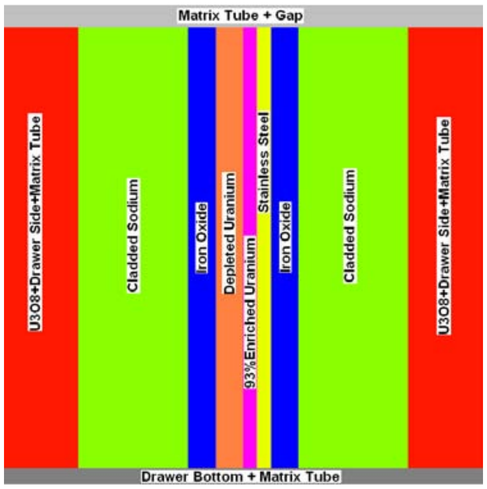


**Figure 7. ZPR-6 Experiment 6A Geometry and U-235 Plate Power Solution.**





**Explicit Geometry** **Simplified (Homogenization) Scheme**

**Figure 8. Fuel Drawer Model for Initial ZPR-6 Assembly 6A Benchmark**

While one can question the impact of homogenizing the various void regions, our experience indicates that the homogenization (where the original density of the matrix tube is reduced and spread into the void region) introduces a negligible error in the leakage. The reason is that the void region can impact only a small fraction of the solid angle with regard to streaming from the dominant neutron-producing plates (uranium and steel). Additionally, the truncation errors resulting from using too few energy groups, two few angles in the angular cubature, and too coarse a spatial mesh overwhelm the error introduced by this spatial homogenization. Even using the geometric simplifications and the homogenization scheme shown on the right of Figure 8, we generate quadratic finite element meshes with 15 million vertices and cubic order meshes with 50 million vertices (note that the empty matrix tube was removed for these calculations because of its relative lack of importance to the eigenvalue). Using the 33-group energy representation with our current preconditioner in space (SSOR-preconditioned CG in PETSc) quickly forces SN2ND to the memory limit of Blue Gene/P.

Figure 9 shows a snapshot of the space-energy distribution obtained for two selected drawers from the front matrix assembly that can be identified in the leftmost picture. Note that drawer "A" is at the exact center of the geometry, while drawer B is chosen significantly away from the core center, and that we took the solution slices very near the axial center. The typical cross section generation process (lattice calculation) assumes a flux solution similar to that observed at the core center (i.e. drawer A) and thus the global flux gradients are not directly accounted for throughout the remaining domain. The differences between the drawers illustrate the importance of the global flux gradient on the flux solution within each drawer.

From the 33-group structure, we plot the spatial gradients for each drawer at 1.1 MeV, 243 keV, and 221 eV. To begin, we note that in the highest energy region, the spatial distributions in the A and B drawers are similar. Further inspection shows the flux gradients to be ~10% between the two drawers (difference in flux magnitude between the drawer slices) and ~40% over each individual drawer (difference in flux magnitude within each drawer slice). At 243 keV, above the resolved resonance range and at the bottom end of the fission source, we see much more pronounced gradients in the flux that are associated with the scattering events occurring primarily in the steel and uranium materials. In drawer B we see evidence of the global flux gradient predominately in the $U_3O_8$ plates at the edge of the drawer. This gradient is not only left to right on the figure but also bottom to top. Overall, this amounts to only a ~10% gradient between the drawers and a ~12% gradient within each drawer. At 221 eV we again see a strong gradient around the $U_3O_8$ plates attributable to the scattering source. We also note the lower magnitude of neutrons in the enriched uranium plates which is an artifact of the higher number of neutrons produced in the enriched uranium plates at higher energies that are then scattered down into this group in the surrounding materials. Overall, we see a ~9% gradient between the drawers and a ~44% gradient within each drawer at this energy.

In general we can state that the reference calculation used to generate the cross sections is sufficient for the test phase. It also seems likely that generating plate wise cross sections using an infinite lattice of the core centered drawer as a reference system would be acceptable, although previous experience indicates we would need more energy groups in the global calculation to capture the space-energy coupling. We emphasize that one can obtain a reference solution to many of the reaction rates within each plate using a continuous energy Monte Carlo approach, but no existing Monte Carlo code can obtain spatial gradients such as those observed in Figure 9 or the gradients observed in the plates between the different drawers. In summary, these results highlight the new ability to visualize and identify the differences between the space-energy flux distributions in each drawer. This data can then be used to devise a better cross section generation methodology in which the reference configurations can more accurately represent the total core configuration.

Continuing with our assessment on parallel performance, we consider the strong scaling performance of SN2ND on Blue Gene/P in Table 2 where a mesh with 1,822,176 quadratic finite elements and 14,845,369 vertices was used. In order to fit within the memory requirements on Blue Gene/P, a 9-group $P_3$ cross section data library was used. We used 16 angles on the half-sphere and 4 angles per process; thus the total number of processors (column 1) is 4 times the number of spatial processors in Table 2. As can be seen, the scalability on Blue Gene/P is excellent in this range of processor counts. We note that strong spatial scaling up to the full machine is not relevant for the workloads encountered in our scoping studies to achieve spatial and angular convergence. The more common workload we are targeting is represented by weak scaling where we grow the number of processors as we increase the number of angles while keeping the spatial part partitioned among a fixed set of processors. We also point out that the SSOR preconditioner is known to require more iterations, and thus more communication, as ever-finer partitions of the spatial domain are used. At a certain load point we can expect the SSOR preconditioner to become completely ineffective and the iteration count to rise dramatically. While we have almost always found this point to be about 800 vertices per process, that experience was for homogenized assembly problems, and the heterogeneous results indicate that the performance drop-off is now just below the level observed in Table 2. Given that the ZPR benchmark requires both a large number of vertices and angles, our emphasis on weak scaling is valid.

We next consider the weak angle scaling performance of SN2ND on Blue Gene/P and XT5. Using the strong scaling information, we target at least 2,500 vertices per process on Blue Gene/P and 3,500 vertices per process on XT5. For Blue Gene/P, we used the 9-group $P_3$ data with a mesh consisting of 698,720 quadratic serendipity hexahedral elements and 2,927,567 vertices. While we would generally prefer to use a finer mesh and energy structure, the memory constraints on Blue Gene/P limit our calculations on the low end of the weak scaling study. Table 3 gives the weak scaling performance on Blue Gene/P for SN2ND where we partitioned the mesh over 1,024 cores (2,858 vertices per core) and increase the order of the Legendre-Tchebychev cubature in correspondence with the processor count.

From Table 3, the weak scaling on Blue Gene/P drops off consistently as we increase the number of directions and reaches 74% at the full machine capacity of 163,840 cores.

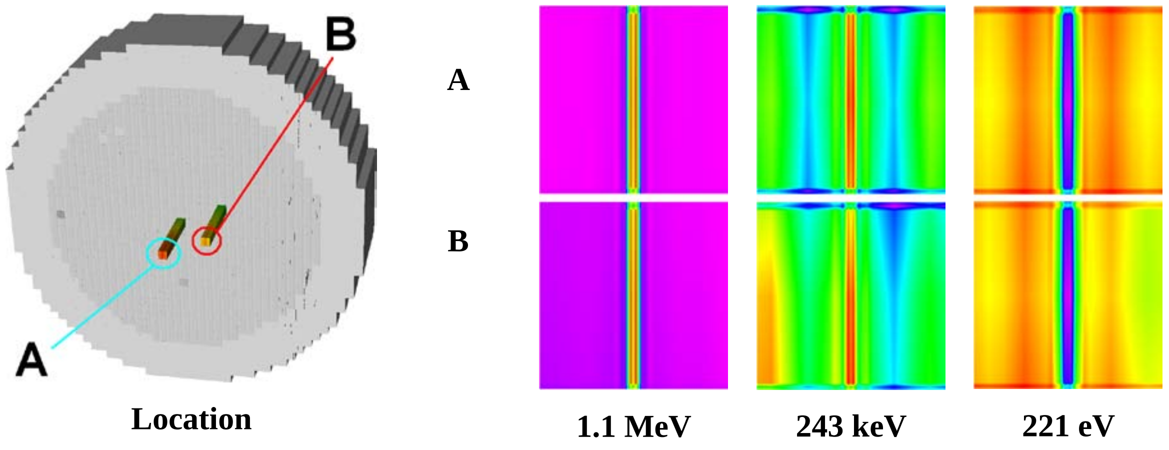


**Figure 9. Flux Distributions in Selected Drawers of ZPR 6-6a**

**Table 2. Spatial Strong Scaling Results for Blue Gene/P**

| Total Cores | Spatial Cores | Vertices/ Process | Total Time (seconds) | Parallel Efficiency |
|---|---|---|---|---|
| 8,192 | 2,048 | 7324 | 2402 | 100% |
| 16,384 | 4,096 | 3662 | 1312 | 92% |
| 24,576 | 6,144 | 2441 | 873 | 92% |
| 32,768 | 8,192 | 1831 | 637 | 94% |

**Table 3. Weak Scaling by Angle for the ZPR 6/6a Problem on Blue Gene/P (2.9 million vertices and 9 energy groups)**

| Cores | $4\pi$ Angles | $k_{eff}$ | Fission Iters. / Time | Total Time (seconds) | Time in Angular System Solver (seconds) | Load Imbalance in Angular System Solver | Source Update (seconds) | Weak Scaling |
|---|---|---|---|---|---|---|---|---|
| 16,384 | 32 | 0.98952 | 21 / 38 | 790 | 364 | 2.9 | 319 | 100% |
| 36,864 | 72 | 0.98960 | 21 / 41 | 855 | 390 | 4.2 | 319 | 92% |
| 49,152 | 96 | 0.98967 | 21 / 41 | 865 | 397 | 5.1 | 319 | 91% |
| 65,536 | 128 | 0.98997 | 21 / 42 | 874 | 420 | 5.3 | 320 | 90% |
| 102,400 | 200 | 0.99032 | 20 / 45 | 894 | 420 | 6.1 | 306 | 88% |
| 131,072 | 256 | 0.99010 | 21 / 45 | 935 | 441 | 7.4 | 321 | 84% |
| 163,840 | 320 | 0.99040 | 21 / 51 | 1066 | 457 | 7.8 | 321 | 74% |

**Table 4. Weak Scaling by Angle for the ZPR 6/6a Problem on XT5 (14.8 million vertices and 33 energy groups)**

| Cores | $4\pi$ Angles | keff | Fission Iters. / Time | Total Time (seconds) | Time in Angular System Solver (seconds) | Load Imbalance in Angular System Solver | Source Update (seconds) | Weak Scaling |
|---|---|---|---|---|---|---|---|---|
| 36,864 | 72 | 0.98867 | 23 / 135 | 3105 | 2601 | 2.2 | 164 | 100% |
| 49,152 | 96 | 0.98875 | 23 / 146 | 3362 | 2830 | 2.8 | 164 | 92% |
| 65,536 | 128 | 0.98907 | 23 / 152 | 3506 | 2927 | 3.2 | 164 | 89% |
| 102,400 | 200 | 0.98945 | 23 / 161 | 3694 | 3007 | 4.0 | 164 | 84% |
| 122,800 | 240 | 0.98949 | 23 / 167 | 3850 | 3193 | 4.9 | 165 | 81% |

Since XT5 does not have the same memory constraints as does Blue Gene/P, we used the 33-group $P_3$ cross section data with a mesh containing 1,822,176 quadratic Lagrangian hexahedral elements and 14,845,369 vertices. Table 4 gives the results of the weak scaling study where we partition the mesh over 4,096 cores (3,624 vertices per core) and we again increase the Legendre-Tchebychev cubature order. We point out that SN2ND code has been run on up to 145,000 cores of XT5 (full machine size) earlier. However, some cabinets of the machine are unavailable at the time of writing this manuscript and we could manage to get only 122,800 cores. Similar to the Blue Gene/P results we see a consistent drop-off in scalability.

To understand the degradation in weak scaling performance, we included the timing and load imbalance information we observed on the preconditioner work (columns 6 and 7 in Tables 3 and 4), which displays the same trend as that observed for the total timing. After checking the iteration information, we also found that the total iteration count is increasing proportionally to the number of angles such that the average number of iterations per angular system is constant. Given that these PETSc operations are not connected in angle, the primary factor responsible for this degradation in performance is load imbalance in angle caused by variability in the condition number of each angular coefficient matrix (see Equation 5). As we increase the order of the angular cubature we are making relatively equal refinements of the angular domain (4π) such that we are not, on average, increasing or decreasing the combined spectral radius of all of the angular systems to be solved (the total number of iterations increases proportionally to the number of directions). However, we are not currently making any adjustment in how we distribute the individual angular system work and thus appear to be assigning some processors multiple difficult systems and other processors relatively easy systems thereby creating a load imbalance in work. While this type of load balance by angle can generally be isolated to the ZPR experiments (due to the high degree of heterogeneity in one particular geometric direction), it is relatively easy to treat in UNIC and we intend to verify the situation and pursue remedies to it in the coming months. We also note that the eigenvalue results are also affected by the monolithic plate orientation, and we see a rather haphazard and slow convergence of the eigenvalue as we refine the angular order of the cubature.

In addition to the scaling data on XT5 in Table 4, we discuss the floating-point performance of SN2ND measured using PAPI next. Due to inadequate per core memory availability and robustness issues, we can only afford to use SSOR preconditioned CG solver from PETSc in SN2ND. It is well known that the performance of such sparse-matrix vector operations based solvers is limited by the available memory bandwidth [24]. On XT5, the STREAM Triad operation achieves about 2.27 GB/s per core on XT5 when all eight cores are used (consistent with the approach used for all of our calculations). Following the methodology in Gropp et al [24], we estimate the memory bandwidth limited performance bound to be ~17% of the theoretical machine peak for sparse matrix-vector multiplication (about 6 bytes per flop). However, matrix relaxation operations usually perform at a slower rate than matrix-vector multiplications. To investigate this further, we saved a matrix from the SN2ND solver and used the CG solver with SSOR preconditioner in a standalone PETSc example (serving as the primary kernel in SN2ND). With a similar workload per core as that shown in Table 4, we get only about 340 MFlop/s per core (3.7% of machine peak) where no parallelism is involved. Given that the SN2ND solver is spending about 85% of its execution time in the PETSc CG/SSOR algorithm (column 6 in Table 4), we can expect this PETSc kernel performance to be an upper bound on the SN2ND solver performance. We get 160-198 MFlop/s per core (47-58% of the ideal PETSc kernel performance on one core) in the processor range in Table 4. Two reasons for the reduced performance (relative to the upper bound) are the processor idling during the synthetic acceleration solve (currently done by only one processor from each set of processors responsible for spatial parts) and the necessary parallel communication. Even though we have significantly reduced the execution time since our original submission in April 2009 (as detailed in Figure 6), our flop rates have declined from 5-7% of machine peak (518-647 MFlop/s per core). As noted earlier, the improvements in execution time came from reduction in extraneous floating point operations in local work (via various algorithmic enhancements) and not from machine specific optimizations. Over the coming months, we will work on studying the sources of low floating point performance and carry out further implementation level optimizations for some significant kernels in UNIC and PETSc to get closer to the memory bandwidth limited performance bound.

## 6. CONCLUSIONS AND FUTURE WORK

The preceding calculations make it clear that we will meet the time-dependent neutronics requirements of the multi-year development and analysis project for sodium cooled fast reactors. With relatively little manpower invested into the SN2ND solver, we were able to combine several “off the shelf” computing packages in a novel way and rapidly produce a neutronics solver that can reliably and justifiably utilize large-scale parallel machines. This new tool already provides accurate and reliable solutions to several difficult numerical benchmark problems for neutron transport, and we anticipate obtaining accurate solutions to the problems discussed in this manuscript within the next couple of months. We have demonstrated the potential of SN2ND solver of our UNIC code to effectively use available large-scale machines in addition to future, larger-sized machines. We have also demonstrated in just the past few months that there exists much room for performance improvements by reducing the total time to solution by a factor of 2.5 via the refactoring of several inefficient parts of the solver algorithm.

With regard to parallel performance, we have demonstrated good strong scalability with respect to spatial parallelization on Blue Gene/P (>90%). The use of the new multigrid preconditioner we are developing should improve this performance further and help alleviate current memory requirements. We have also demonstrated good weak angle scalability on both Blue Gene/P (74% at 163,840) and XT5 (81% at 122,800) for problem sizes of up to 120 billion degrees of freedom, and further load-balancing efforts should produce even better weak scalability numbers. Time restrictions prevented us from investigating strong angle and weak spatial scalability.

These calculations demonstrate our strong motivation to continue pushing toward larger meshes, more angles, and more energy groups in order to achieve the desired level of accuracy. At present, the ZPR mesh with 49,800,865 vertices is the largest spatial mesh we have attempted (40,960 processors of XT5). Given that we need to further extend the geometry model of ZPR (include the empty matrix tube), employ more angles (~400-500), and use more energy groups (~100), one can see that further enhancements will be necessary to fully achieve both our short- and long-term goals. As the SN2ND solver evolves, we will continue to carry out extensive performance optimizations at the algorithmic and implementation levels while adapting to the hostile memory hierarchy limitations. The inclusion of our spatial *p*-multigrid preconditioning scheme – nearly ready for deployment – should provide an immediate boost to problem size capability and performance gains since we will be replacing the memory-bandwidth limited sparse-matrix operations with more efficient matrix-vector operations that have previously demonstrated excellent floating-point performance [25]. The inclusion of an effective *h*-multigrid preconditioner scheme, for which we are in the initial research phase, improve the performance further and thereby facilitate larger problem sizes.

In addition to these changes, we intend to implement a fixed-iteration algorithm by angle to impose proper load balancing and focus on putting more angles per process. This will reduce the communication overhead on the angle communicator. We also need to partition the diffusion synthetic acceleration equation (angle preconditioner) over more processors of the angular communicator (currently assigned to just the first process). This procedure should allow us to further reduce the time to solution and reduce the current load imbalance in angle due to the synthetic equation. We have also initiated work on a solution methodology that incorporates parallelization of the last remaining independent variable: energy. To accomplish this, we will implement a Krylov-subspace methodology using the algorithm we have built as a preconditioner. This is essential because with time-dependent problems, the inclusion of the fission source as “upscattering” in $T$ of Eq. (8) will greatly degrade the performance of the Gauss-Seidel algorithm that we currently use in the steady state mode. While our current level of parallelization already saturates the available resources, our desire to refine the energy resolution to improve the accuracies will allow us to continue using the larger future machines with millions of cores effectively. We hope to start performing calculations with 200+ groups, 400+ angles, and 100+ million vertices within the next five years so that explicit geometry time-dependent coupled multiphysics simulations of reactor technology can be realized and uncertainties with the existing approaches can be systematically removed.

**REFERENCES**

[1] E. E. Lewis and Jr. W. F. Miller, “Computational Methods of Neutron Transport,” John Wiley & Sons, New York, 1984.

[2] G. Palmiotti, M. A. Smith, C. Rabiti, M. Leclere, D. Kaushik, A. Siegel, B. Smith, E.E. Lewis, “UNÌC: Ultimate Neutronic Investigation Code,” Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications, Monterey, California, April 15-19, 2007.

[3] Argonne National Laboratory, Argonne Leadership Computing Facility, http://www.alcf.anl.gov.

[4] Oak Ridge National Laboratory, National Center for Computational Sciences, http://www.nccs.gov.

[5] C. Rabiti, E. Wolters, M. A. Smith, G. Palmiotti, “Spherical Quadratures for the Discrete Ordinates Method,” Trans. Am. Nucl. Soc 96, Boston, MA, June 24-28, 2007.

[6] J. Lamarsh, “Introduction to Nuclear Engineering,” Addison-Wesley Publishing Co, Massachusetts, 1983.

[7] R. E. Alcouffe, R. S. Baker, J. A. Dahl, S.A. Turner, and Robert Ward, “PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System,” LA-UR-05-3925, May 2005.

[8] W. A. Rhoades, R. L. Childs, M. B. Emmett, and S. N. Cramer, “Application of the Three-Dimensional Oak Ridge Transport Code,” Proc. Am. Nucl. Soc. Topical Meeting on Reactor Physics and Shielding,” pp. 225-238, Chicago, September 17-19, 1984.

[9] Sjoden, G. and Haghighat, A., “PENTRAN™: Parallel Environment Neutral particle TRANsport in 3-D Cartesian Geometry”, Proc. Int. Conf. on Mathematical Methods and Supercomputing for Nuclear Applications, pp. 232–234. Saratoga Springs, NY (1997). Performance of PENTRAN™ 3-D Parallel Particle Transport Code.

[10] G. Palmiotti, E. E. Lewis and C. B. Carrico, “VARIANT: VARIational Anisotropic Nodal Transport for Multidimensional Cartesian and Hexagonal Geometry Calculation,” ANL-95/40 Argonne National Laboratory, 1995.

[11] C. R. E. de Oliveira, A. J. H. Goddard, “EVENT - A Multidimensional Finite Element-Spherical Harmonics Radiation Transport Code,” Proceedings of the OECD International Seminar on 3D Deterministic Radiation Transport Codes, Paris, December 01-02, 1996.

[12] John M. McGhee, Randy M. Roberts, Jim E. Morel, “The DANTE Boltzmann Transport Sove: An Unstructured Mesh, 3-D, Spherical Harmonics Algorithm Compatible with Parallel Computer Architectures,” Jnt. Int. Conf. on Math. Meth. and super. For Nuc. Apps., Saratoga Springs, New York, Oct. 5-10, 1997.

[13] T. A. Wareing, J. M. McGhee, J. E. Morel, and S. D. Pautz, “Discontinuous Finite Element Sn Methods on 3-D Unstructured Meshes,” Nuclear Science and Engineering, 138:1-13, 2001.

[14] A. Siegel, D. Kaushik, P. Fischer, G. Palmiotti, M. A. Smith, C. Rabiti, J. Ragusa, “Software Design of SHARP,” Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications, Monterey, California, April 15-19, 2007.

[15] Rabiti, M. A. Smith, G. Palmiotti, “A Three-dimensional Method of Characteristics on Unstructured Tetrahedral Meshes,” Trans. Am. Nucl. Soc. 96, 470, 2007.

[16] M. A. Smith, G. Palmiotti, C. Rabiti, D. Kaushik, A. Siegel, B. Smith, E.E. Lewis, “PNFE Component of the UNÌC Code,” Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications, Monterey, California, April 15-19, 2007.

[17] S. Balay, K. R. Buschelman, W. D. Gropp, D. K. Kaushik, M. G. Knepley, L. C. McInnes, and B. F. Smith, “PETSc Home Page,” http://www.mcs.anl.gov/petsc.

[18] M. A. Smith, G. Palmiotti, et al., “Benchmark on Deterministic Transport Calculations Without Spatial Homogenization (MOX Fuel Assembly 3-D Extension Case),” OECD/NEA document, NEA/NSC/DOC(2005)16, October 2005.

[19] Personal Communication, Frederic Varaine, CEA, “IAEA CRP on PHENIX End of Life Tests; Control Rod Withdrawal,” 2009.

[20] Lawrence Livermore National Laboratory, “Visit: A Distributed, Parallel, Visualization Tool,” https://wci.llnl.gov/codes/visit/home.html.

[21] Sandia National Laboratory, “CUBIT: Geometry and Mesh Generation Toolkit,” http://cubit.sandia.gov/.

[22] Amine Abou-Rjeili and George Karypis, “Multilevel Algorithms for Partitioning Power-Law Graphs,” IEEE International Parallel & Distributed Processing Symposium (IPDPS), 2006.

[23] NEA Nuclear Science Committee, “International Handbook of Evaluated Criticality Safety Benchmark Experiments,” NEA/NSC/DOC(95)03, September 2007.

[24] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, “Toward Realistic Performance Bounds for Implicit CFD Codes,” In D. Keyes, A. Ecer, J. Periaux, N. Satofuka, and P. Fox, editors, Proceedings of Parallel CFD’99, pages 233–240. Elsevier, 1999.

[25] D. Kaushik, W. Gropp, M. Minkoff, B. Smith, “Improving the Performance of Tensor Matrix Vector Multiplication in Cumulative Reaction Probability Based Quantum Chemistry Codes,” Proceedings of the 15th International Conference on High Performance Computing (HiPC 2008), Bangalore, India, Springer LNCS5374:120-130, December, 2008.

# Enabling High-Fidelity Neutron Transport Simulations on Petascale Architectures

*Supercomputing 2009, Portland, Oregon, Nov 14-20, 2009*

Dinesh Kaushik
Micheal Smith
Allan Wollaber
Barry Smith
Andrew Siegel
Won Sik Yang

*Argonne National Laboratory*
*Argonne, IL 60439*
*kaushik@mcs.anl.gov*

## Organization of the Presentation

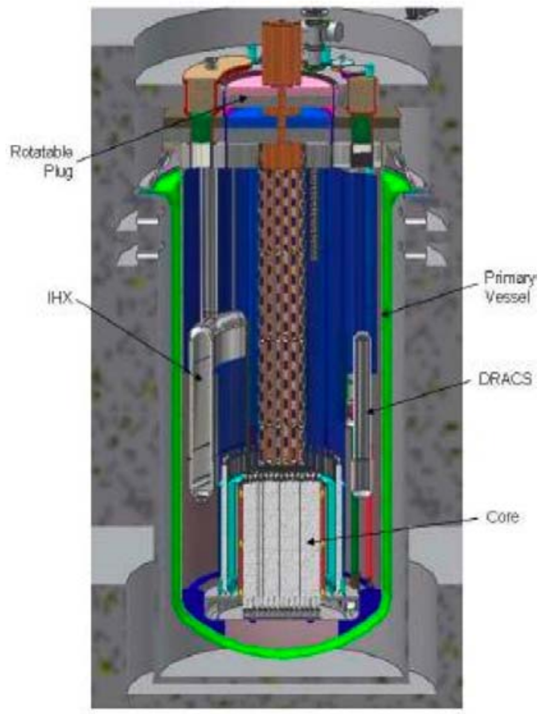
- Overview of SHARP Project at Argonne
- Computational issues for neutronics
- Full-core test problems
- Parallel performance of UNIC
- Summary

## Nuclear Reactor Simulations

- Nuclear fission energy is key component of our current and future energy needs
- Urgent need to develop reactors that are
  - Safe
  - efficient
  - Affordable
- Modeling and simulation tools were simplified to match the available computing technology
  - designers relied on expensive and complicated experiments for satisfactory answers
- Advanced simulation can help in evaluating new designs with reduced dependence on experiments
- This work is supported by Nuclear Energy Advanced Modeling and Simulation (NEAMS) program of US Department of Energy

## Case for Fast Reactor Simulations

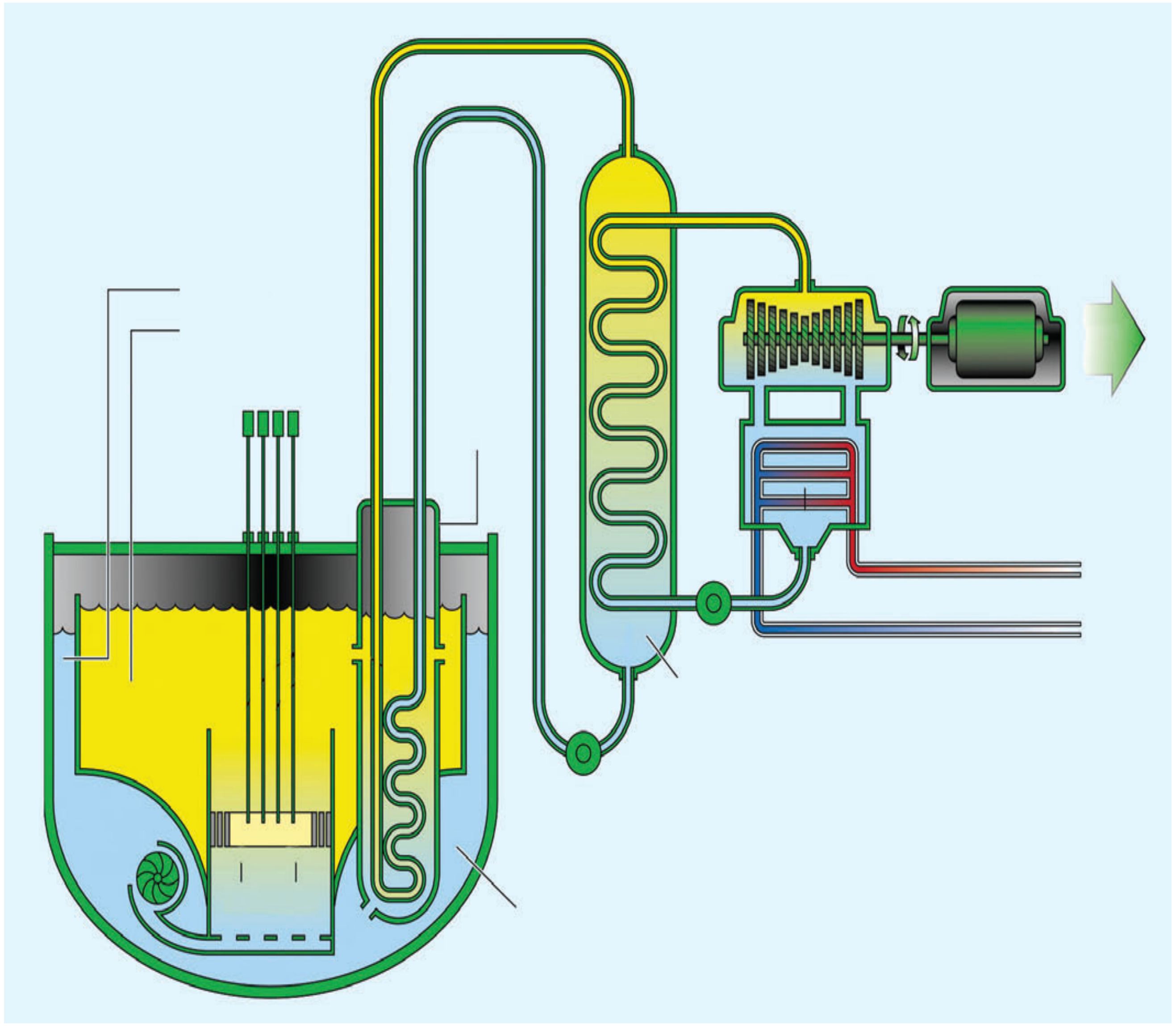
- High energy neutrons are used to convert uranium to plutonium
- Recycle the spent fuel from light water reactors (LWR)
  - Reducing heat load on storage because of lower concentration of transuranic elements
- Through high fidelity simulations,
  - Lower uncertainty margins of the new reactor designs
    - 1% improvement in daily power output translates to millions of dollars for utility companies
  - Global design optimization for enhanced safety and cost



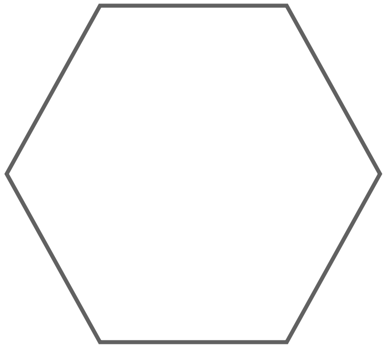
## Simulation based High-efficiency Advanced Reactor Prototyping (SHARP)

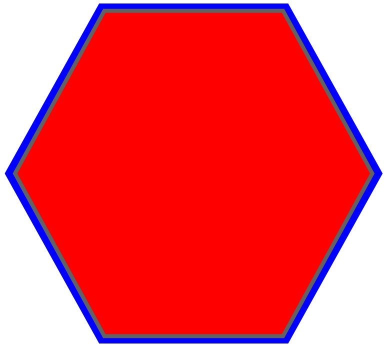
- A tight integration of multiphysics and multiscale modeling of physics phenomena based on a first principles approach
  - an integrated system of software tools
  - accurate description of
    - the overall nuclear plant behavior in a high fidelity way
    - coupling among the different phenomena taking place during reactor operation ranging from neutronics to fuel behavior, from thermal-hydraulics to structural mechanics
- Features
  - Ability to derive basic data and static and dynamic (operating conditions) properties from first principles based methodologies and fundamental experiments
  - to define and plug-in new and different combinations of physics-module implementations to study different phenomena,
  - define and combine different numerical techniques, configure the code easily to run on new platforms
  - develop new physics components without expert knowledge of the entire system.

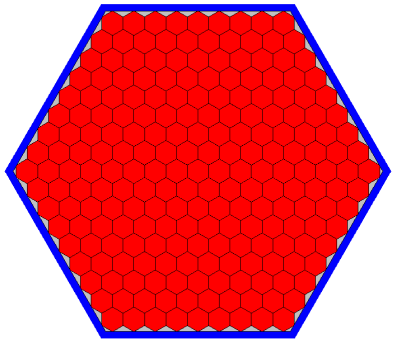
# Schematic diagram of a fast reactor

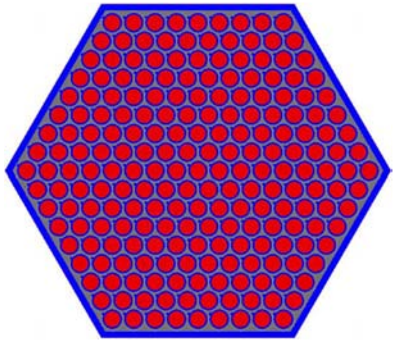


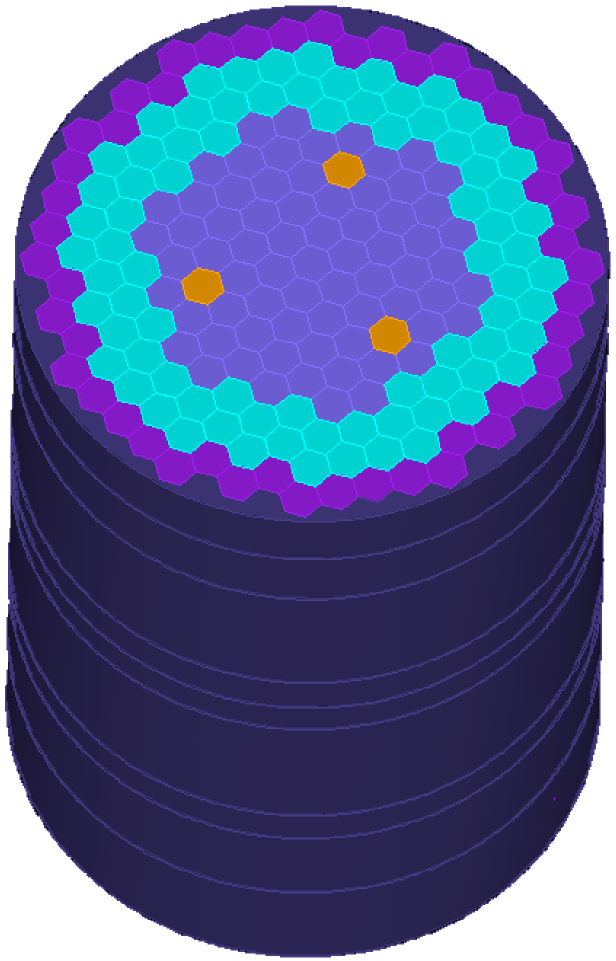
## Homogenization at various levels


Homogenized assembly


Homogenized assembly internals

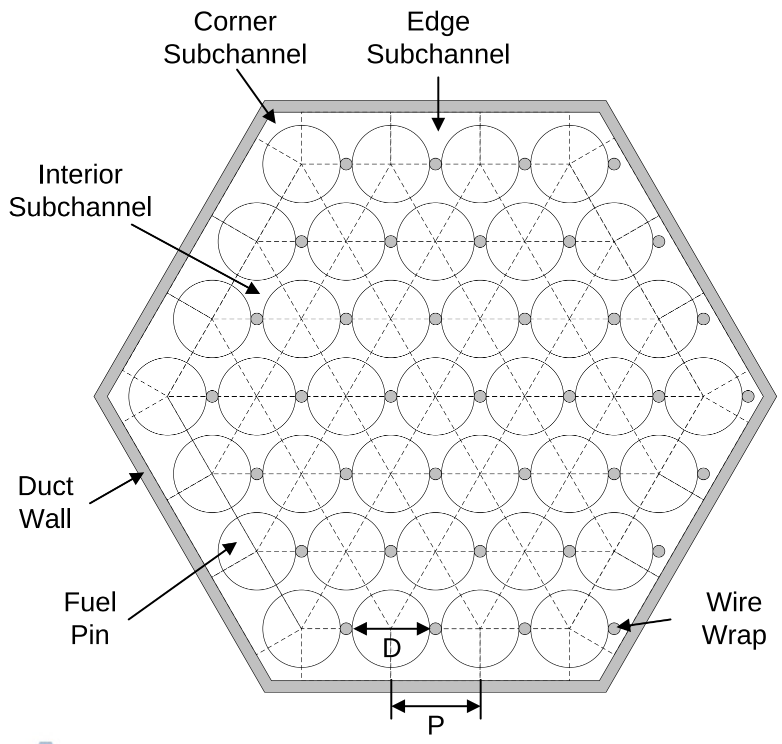

Homogenized pin cells

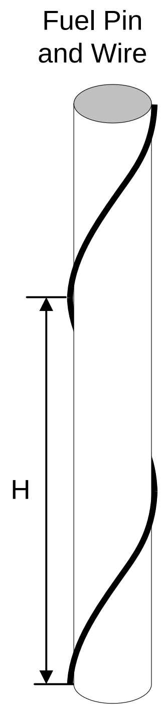

Fully explicit assembly

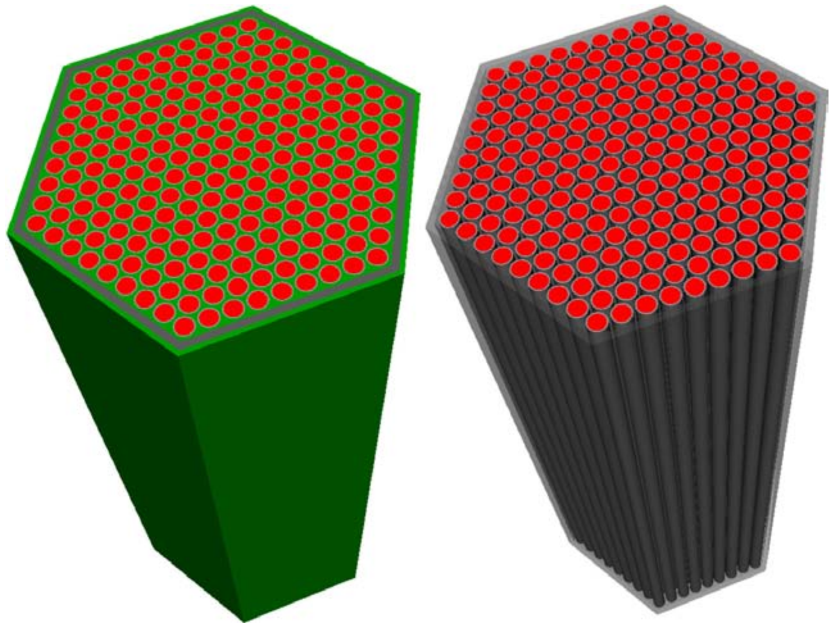


# Fine Detail: Wire Wrapped Pins in Subassembly

- *Resolving wire wrap (diameter = 0.11 cm) leads to 10-100 billion element meshes and about $10^{15}$ degrees of freedom (DOF) for advanced burner test reactor (ABTR) core (2.3 m in diameter and 3.5 meter long)*







## UNIC: Neutronics Module in SHARP

- A 3D unstructured deterministic neutron transport code
- solves
  - second order form of transport using FEM (PN2ND and SN2ND) and
  - first order form by method of characteristics (MOC)
- Parallel implementation using PETSc solvers

# The Steady State Transport Equation (p46 in Lewis)

$$\hat{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\hat{\Omega},E)+\Sigma_t(\vec{r},E)\psi(\vec{r},\hat{\Omega},E)=\int\int\Sigma_s(\vec{r},\hat{\Omega}'\rightarrow\hat{\Omega},E'\rightarrow E)\psi(\vec{r},\hat{\Omega}',E')d\Omega'dE'$$
$$+\frac{1}{k}\chi(\vec{r},E)\int\int\nu(\vec{r},E')\Sigma_f(\vec{r},E')\psi(\vec{r},\hat{\Omega}',E')d\Omega'dE'$$
$$+S(\vec{r},\hat{\Omega},E)$$

| | |
|---|---|
| $\psi(\vec{r},\hat{\Omega},E)$ | The neutron flux (neutron density multiplied by speed) |
| $\Sigma_t(\vec{r},E)$ | The total probability of interaction in the domain |
| $\Sigma_s(\vec{r},\hat{\Omega}'\rightarrow\hat{\Omega},E'\rightarrow E)d\Omega dE$ | The scattering transfer kernel |
| $\chi(\vec{r},E)\quad\nu(\vec{r},E)\quad\Sigma_f(\vec{r},E)$ | The steady state multiplicative fission source |
| $S(r,\hat{\Omega},E)$ | A generic source |
| $k$ | The multiplication eigenvalue |

## Solving the Eigenvalue Problem

$Ax = \lambda x$ Cast the transport equation as a pseudo matrix-vector operation

T = streaming/collision/scattering F = fission

$$\psi = \left[ \psi_1(\vec{r},\hat{\Omega}) \quad \psi_2(\vec{r},\hat{\Omega}) \quad \cdots \quad \psi_G(\vec{r},\hat{\Omega}) \right]^T$$

$$T\psi = \hat{\Omega}\cdot\vec{\nabla}\psi_g(\vec{r},\hat{\Omega}) + \Sigma_{t,g}(\vec{r})\psi_g(\vec{r},\hat{\Omega}) - \sum_{g'=1}^{G}\int \Sigma_{s,g'\to g}(\vec{r},\hat{\Omega}'\to\hat{\Omega})\psi_{g'}(\vec{r},\hat{\Omega}')d\Omega'$$

$$F\psi = \chi_g(\vec{r})\sum_{g'=1}^{G}\int \nu_{g'}(\vec{r})\Sigma_{f,g'}(\vec{r})\psi_{g'}(\vec{r},\hat{\Omega}')d\Omega'$$

$$T\psi = \frac{1}{k}F\psi$$

Standard eigenvalue notation:

$$Ax = \lambda x$$

$$A = T^{-1}F$$

$$x = \psi$$

$$\lambda = k$$

# k-Eigenvalue Power Iteration

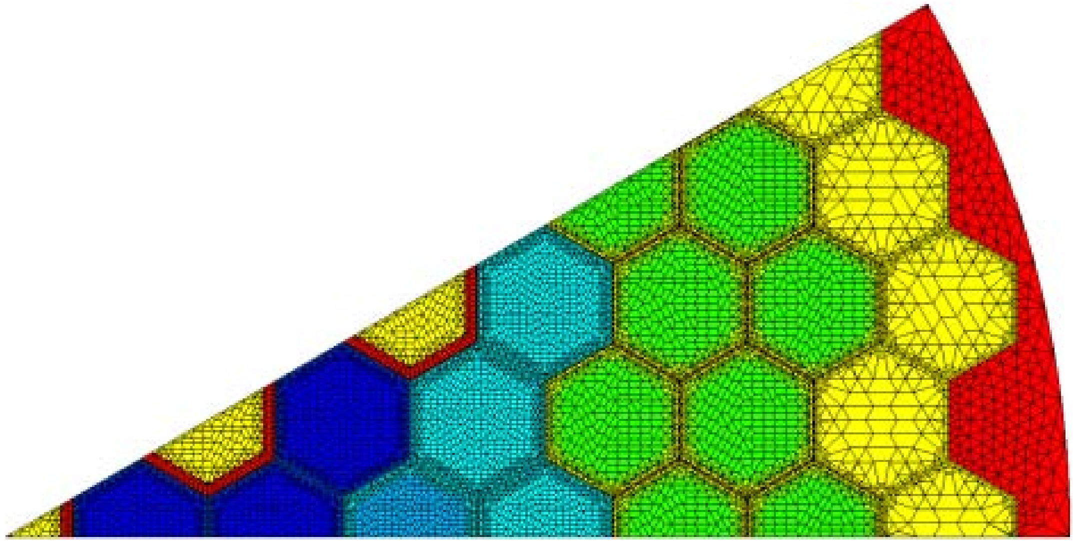
```
Begin Outer Iteration
    Begin Loop over energy groups
      Begin Scattering iteration for the within-group scattering system
        Begin Conjugate gradient over the whole space-angle system
              Obtain group scattering+fission+fixed sources
              Solve a symmetric positive definite linear system for flux
                  (preconditioned conjugate gradient)
        End Conjugate gradient over the whole space-angle system
      End Scattering iteration for the within-group scattering system
    End Loop over energy groups
    Check for convergence in eigenvalue, angular flux, and sources
End Outer Iteration
```

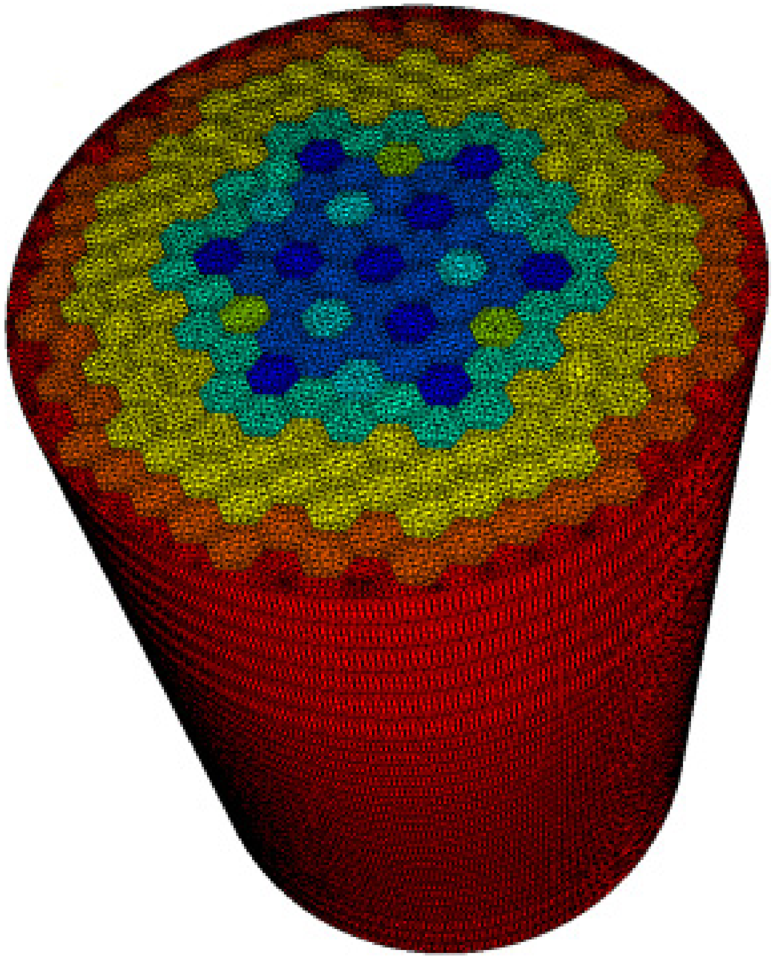
## Features of Second Order Form Solutions in UNIC

- PN2ND and SN2ND solvers have been developed to solve the steady-state, second-order, even-parity neutron transport equation
  - PN2ND: Spherical harmonic method in 1D, 2D and 3D geometries with FE mixed mesh capabilities
  - SN2ND: Discrete ordinates in 2D and 3D geometries with FE mixed mesh capabilities
- These second order methods have been implemented on large scale parallel machines
  - Linear tetrahedral and quadratic hexahedral elements
  - Fixed source and eigenvalue problems
  - Arbitrarily oriented reflective and vacuum boundary conditions
  - PETSc solvers are utilized to solve within-group equations
    - Conjugate gradient method with SSOR and
    - ICCgives better flop rates but requires more memory (not used)
  - Synthetic diffusion acceleration for within-group scattering iteration
  - Inverse Power iteration method for eigenvalue problem
  - MeTiS is employed for mesh partitioning

# ABTR Whole-Core Calculations

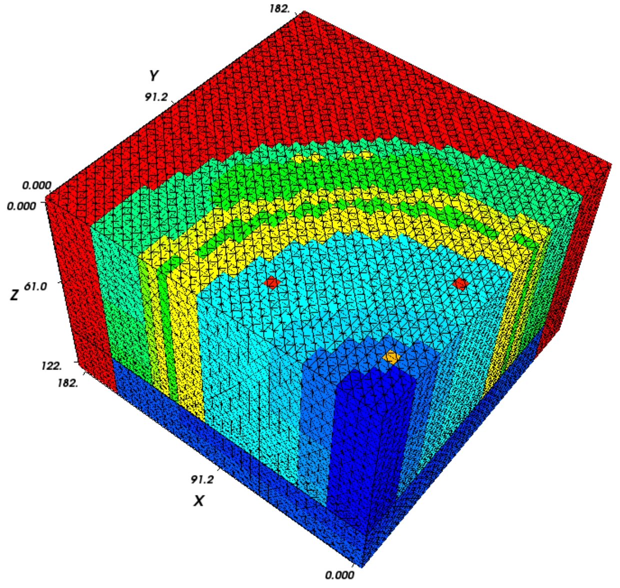
| Angular Directions | Spatial Mesh Approximation | | | | |
|---|---|---|---|---|---|
| | 78243 | 113873 | 461219 | 671219 | 785801 |
| 32 | -241 | -233 | -69 | -64 | -59 |
| 50 | -220 | -210 | -47 | -40 | -37 |
| 72 | -225 | -217 | -51 | | |
| 98 | -216 | -207 | -43 | | |
| 288 | -216 | | | | |

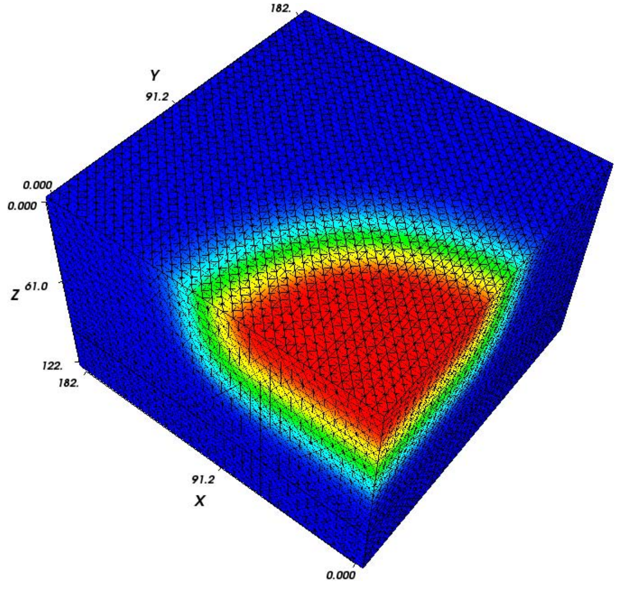


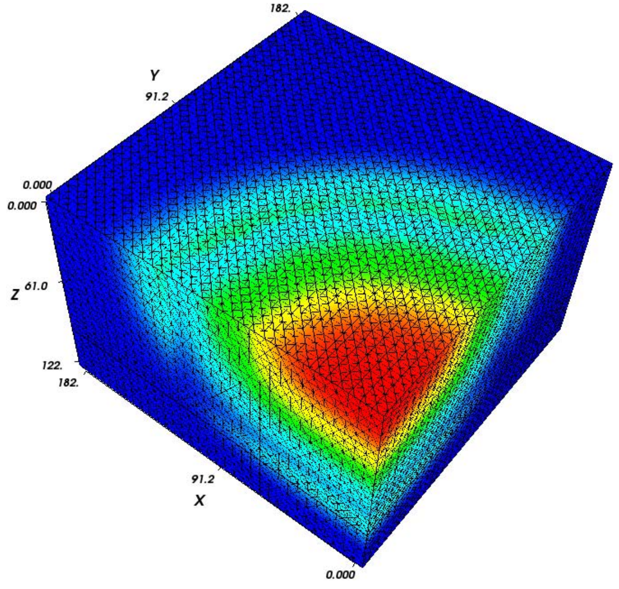


# ZPPR-15 Critical Experiments

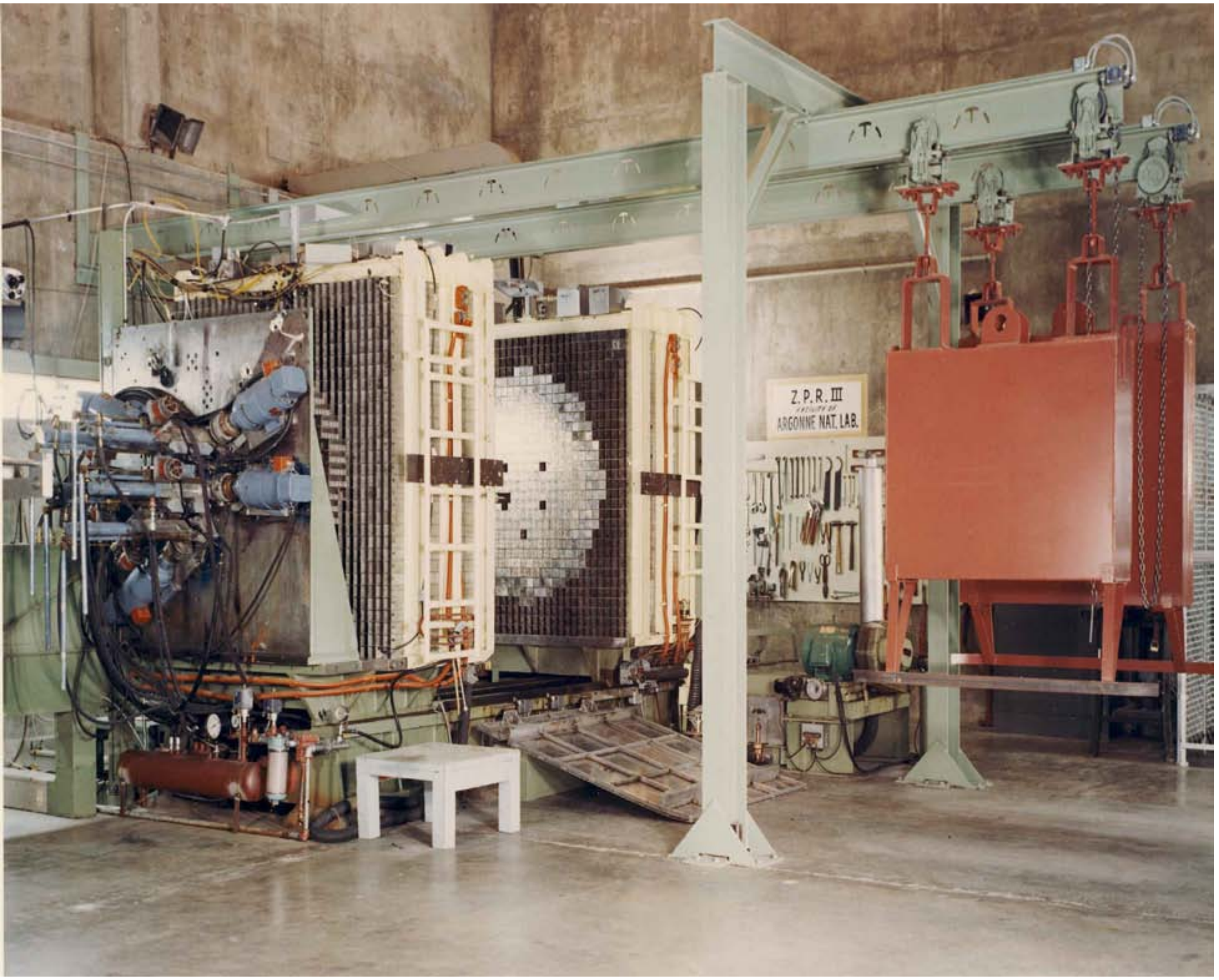
| Flux expansion order | Scattering order | Eigenvalue |
|---|---|---|
| $P_1$ | $P_1$ | 0.99258 |
| $P_3$ | $P_3$ | 0.99640 |
| $P_5$ | $P_3$ | 0.99651 |
| Monte Carlo (VIM) | | $0.99616 \pm 0.00010$ |







Computational Mesh and Example Flux Solutions of ZPPR-15 Critical Experiment

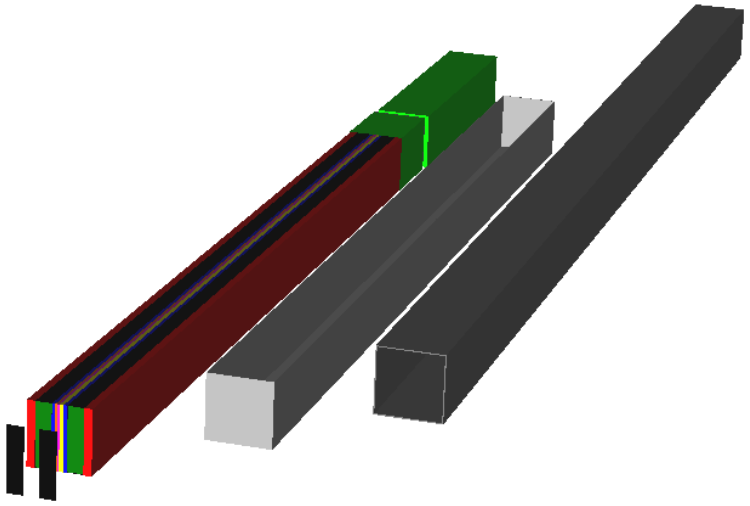


ZPR-3

Over a period of 30 years more than a hundred Zero Power Reactor (ZPR) critical assemblies were constructed at Argonne National Laboratory.

ZPR-3, ZPR-6, ZPR-9 and ZPPR, were all separate fast critical assembly facilities with each machine being used for thousands of individual experiments

## ZPR Test Problem


Single ZPR Drawer

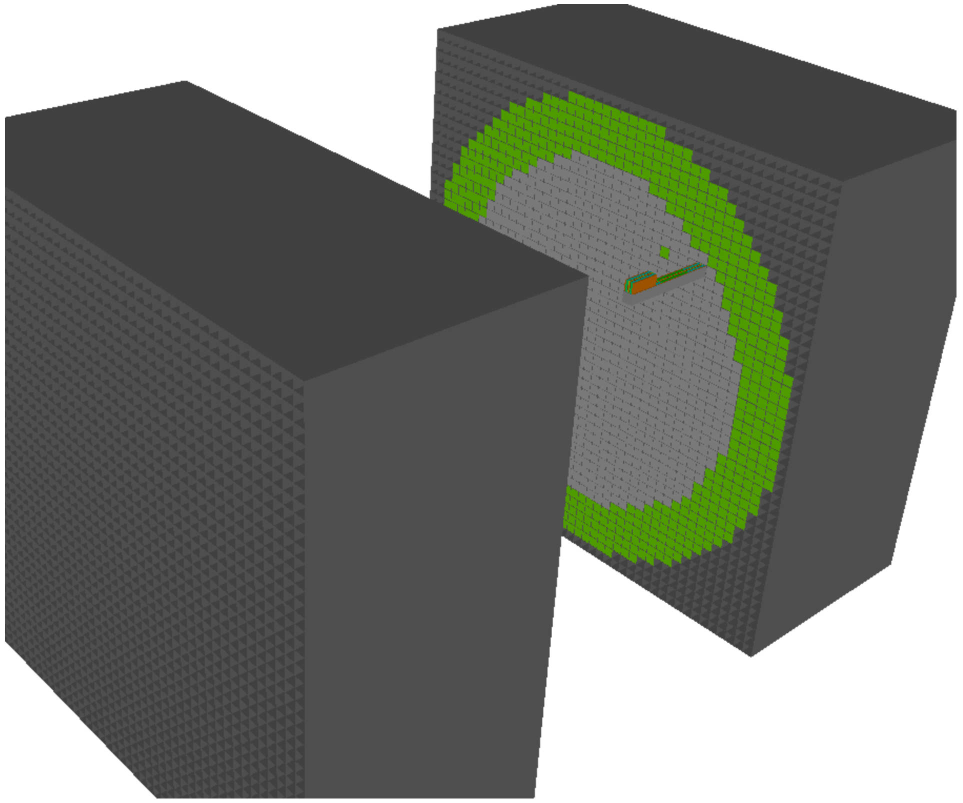


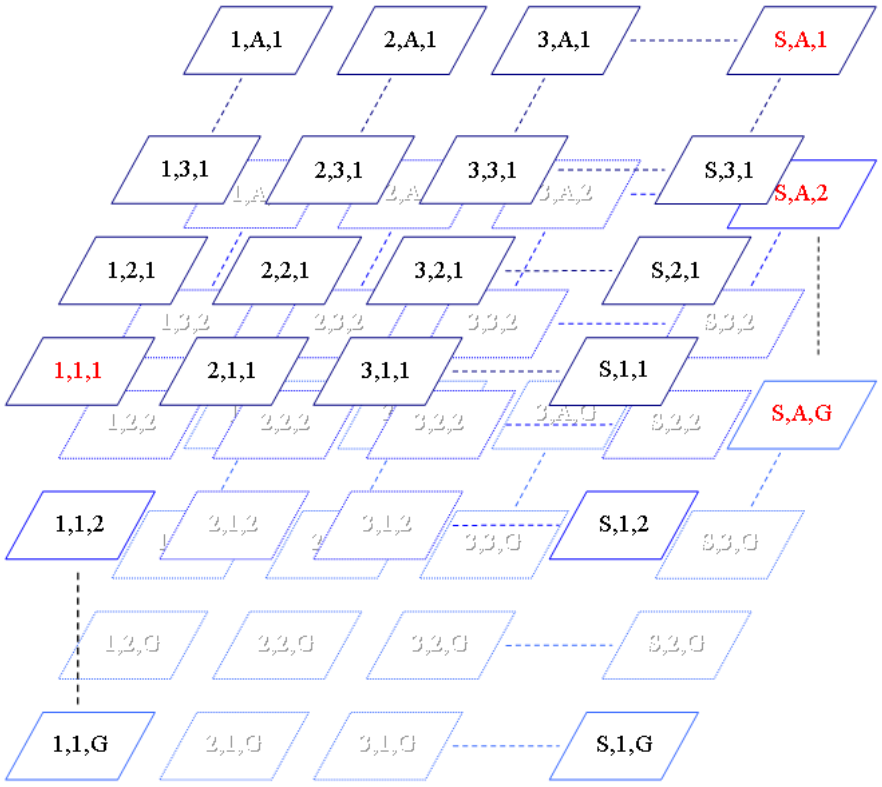
Plate by Plate ZPR Geometry

A ZPR calculation is the first step to full core heterogeneous reactor calculations

- Up to 50 million vertices (~equivalent to 200 million PARTISN finite difference cells)
- 100+ angles with $P_5$ anisotropic scattering
- 100 energy groups
- No thermal-hydraulics considerations (i.e. clean comparison, MCNP/VIM solvable)

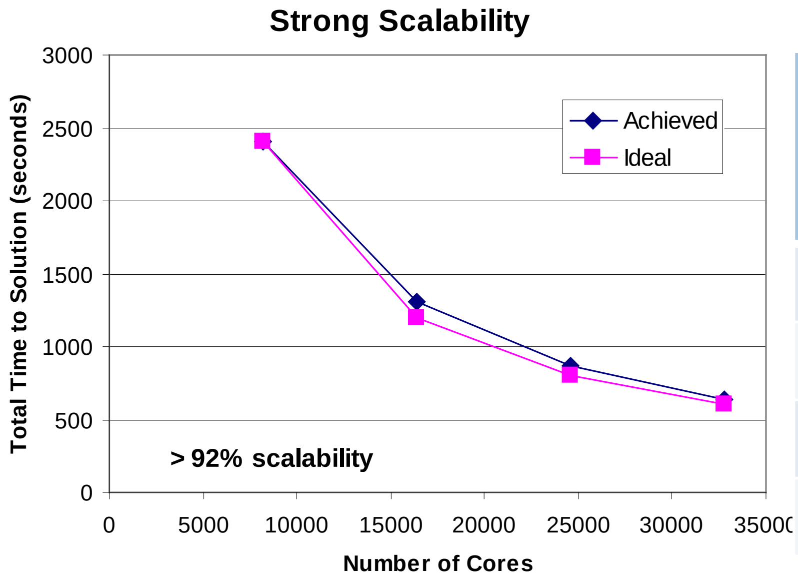
# Parallelism in Space, Angle, and Energy

- Hierarchical Partitioning by using MPI communicators
- Parallelization in every dimension is important (to avoid per-core memory limit)
- User defined MPI communicators are not always optimized for mapping to cores



## Performance on Blue Gene/P – Strong Scaling

- 9 energy groups
- Mesh (simplified geometry)
  - 15 million vertices and 1.8 million hexahedral quadratic elements
  - Spread over 4,096 processor cores (virtual node mode)
  - 4 angles per processor-core



| Total Cores | Vertices/ Process | Total Time (seconds) | Parallel Efficiency |
|---|---|---|---|
| 8,192 | 7,324 | 2,402 | 100% |
| 16,384 | 3,662 | 1,312 | 92% |
| 24,576 | 2,441 | 873 | 92% |
| 32,768 | 1,831 | 637 | 94% |

## Performance on Blue Gene/P – Weak Scaling
### ANL: 40 racks (163,840 cores); JSC: 72 racks (294,912 cores)

- Weak scaling important for scoping studies
- 9 energy groups
- Mesh
  - 7 million vertices and 1.7 million hexahedral quadratic elements
  - Spread over 4,096 processor cores (virtual node mode)
  - 2 angles per processor-core

| Total Cores | 4π Angles | Total Time (seconds) | Weak Scaling |
|---|---|---|---|
| 32,768 | 32 | 579 | 100% |
| 73,728 | 72 | 572 | 101% |
| 131,072 | 128 | 581 | 100% |
| 163,840 | 160 | 691 | 84% |
| 294,912 | 288 | 763 | 76% |

## Performance on XT5

### Recently upgraded hex-core system, 2.6 GHz, 225K total cores

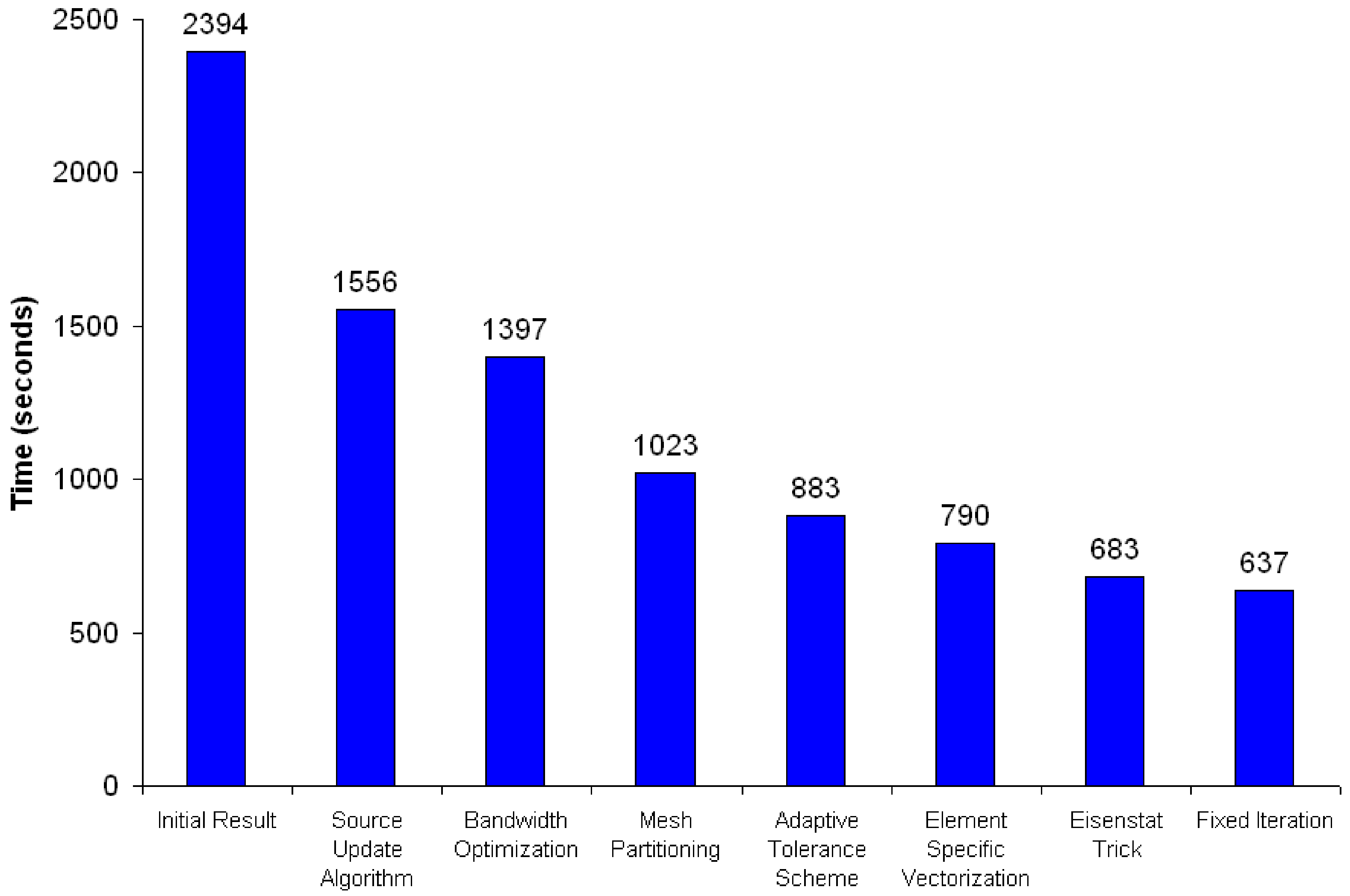
- 33 energy groups
- Mesh (real experimental geometry)
  - 10 million vertices and 2.4 million hexahedral quadratic elements
  - Spread over 2,064 processor cores
  - 2 angles per processor-core

| Total Cores | 4π Angles | Total Time (seconds) | Weak Scaling |
|---|---|---|---|
| 16,512 | 32 | 1891 | 100% |
| 37,152 | 72 | 1901 | 99% |
| 66,048 | 128 | 1829 | 103% |
| 103,200 | 200 | 2050 | 92% |
| 148,608 | 288 | 2298 | 82% |
| 222,912 | 432 | 2517 | 75% |

## Performance Optimizations

- Reordering for better cache reuse
- Unrolled loops for specific element types (better vectorization)
- Weighted partitioning for load balance in mesh partitioning
- Fixed iteration scheme for load balance across angular systems
- Eisenstat’s Trick (lower flop rate but better execution time)

## Performance Optimizations –
### Execution Time Reduced by a Factor of 4 on 16,384 Cores



## Assessing the Single Core Performance

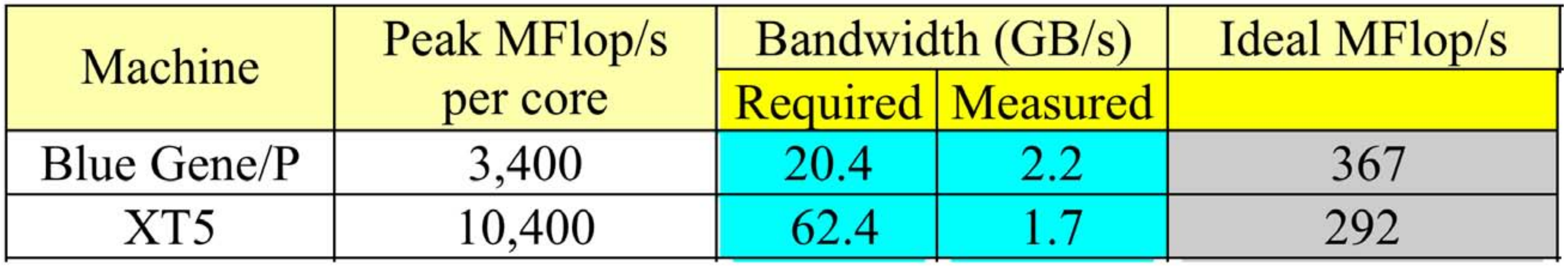
- Execution time was optimized (often) at the cost of flops (likely unnecessary)
- Sparse matrix vector multiplication  (BLAS Level 2) operation is the main kernel
  - Performance is memory bandwidth limited (little data reuse)
  - High ratio of load/store to instructions/floating-point ops
  - Flops not the right metric
  - Inadequate memory bandwidth on both architectures

## Stream Benchmark on Cray XT5 and BlueGene/P (MB/s for the Triad Operation)

| Threads per Node | Cray XT5 | | BlueGene/P | |
|---|---|---|---|---|
| | Total | Per Core | Total | Per Core |
| 1 | 8448 | 8448 | 2266 | 2266 |
| 2 | 10112 | 5056 | 4529 | 2264 |
| 4 | 10715 | 2679 | 8903 | 2226 |
| 6 | 10482 | 1747 | - | - |

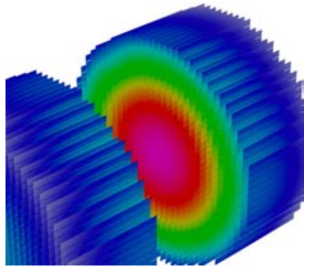
## Ideal Sparse Matrix-Vector Performance

**Required: 6 bytes/flop**



| Machine | Peak MFlop/s per core | Bandwidth (GB/s) | | Ideal MFlop/s |
|---|---|---|---|---|
| | | Required | Measured | |
| Blue Gene/P | 3,400 | 20.4 | 2.2 | 367 |
| XT5 | 10,400 | 62.4 | 1.7 | 292 |

## Summary



- UNIC provides reactor designers a scalable and flexible simulation tool that has the potential to transform the reactor analysis field by exploiting supercomputing with far reaching consequences on reactor development cost and safety.
- We were able to resolve the complex geometric features of the full ZPR core geometry for the first time.
- UNIC scales well on the two largest machines:
  - 76% on 294,942 cores of Blue Gene/P (ANL& JSC, Jugene is the largest in core count)
  - 75% on 222, 912 cores of XT5 (ORNL, #1 in TOP500)
  - Uses up to 500 billion degrees of freedom
- No other code in the field (deterministic neutron transport) has scaled to this level or solved full core-sized problems with this fidelity.
- Computational challenges need to be tackled at the modeling, algorithmic, and architectural levels for future machines with millions of cores.

## Acknowledgements

- Elmer Lewis of Northwestern University for providing us algorthmic guidance
- David Keyes of KAUST/Columbia University, Paul Fisher of ANL, and Jean Ragusa of Texas A&M for many interesting discussions
- ALCF for time on Intrepid and staff help (Paul Messina, and Ray Loy in particular)
- NCCS for time on Jaguar and staff help (especially Ricky Kendall)
- JSC Staff for Jugene time (Bernd Mohr and Jutta Doctor were very helpful)

## Further Reading

- P. Fischer, D. Kaushik, D. Nowak, A. Siegel, W. S. Yang, G. W. Pieper, “Advanced Simulation for Fast Reactor Analysis”, SciDAC Review, Issue 9, Fall 2008, pp 12-21, http://www.scidacreview.org/0803/pdf/nuclear.pdf.
- M. A. Smith, D. Kaushik, A. Wollaber, W. S. Yang, B. Smith, C. Rabiti, G. Palmiotti, “Recent Research Progress on UNIC at Argonne National Laboratory,” Proceedings of the International Conference on Advances in Mathematics, Computational Methods, and Reactor Physics (M&C 2009), Saratoga Springs, New York, May 3-7, 2009.
- D. Kaushik, W. Gropp, M. Minkoff, B. Smith, “Improving the Performance of Tensor Matrix Vector Multiplication in Cumulative Reaction Probability Based Quantum Chemistry Codes,” Proceedings of the 15th International Conference on High Performance Computing (HiPC 2008), Bangalore, India, Springer LNCS5374:120-130, December, 2008.
- A. Siegel, T. Tautges, A. Caceres, D. Kaushik, G. Palmiotti, M. Smith, “Software Design of SHARP,” Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007), Monterey, California, April 15-19, 2007.

## SHARP Framework Tutorial

SciDAC
Scientific Discovery through Advanced Computing

## Outline

- ITAPS Data Model
- iMesh Interface (w/ examples)
- MOAB vs. ITAPS
- Best Practices (for Performance)
- Parallel Data
- MCS Access

## Introduction

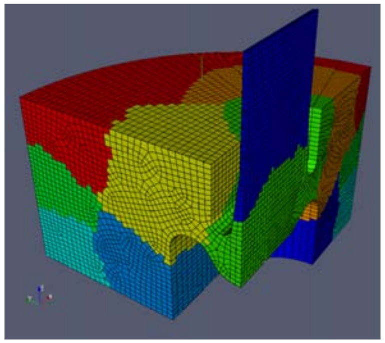
- ITAPS is an *interface specification*
  - Data model + functions
- MOAB is an *implementation* of ITAPS
- MOAB's data model very similar to ITAPS', because ITAPS derives in part from MOAB
- The two are tuned for different things:
  - MOAB tuned for memory efficiency first, speed a close second
  - ITAPS tuned for interoperability (MOAB has only a C++ native interface)
- Use ITAPS from Fortran & where memory isn't important, otherwise MOAB

## ITAPS Data Model

- 4 fundamental "types":
  - *Entity:* fine-grained entities in interface (vertex, tri, hex)
  - *Entity Set:* arbitrary set of entities & other sets
    - Parent/child relations, for embedded graphs between sets
  - *Interface:* object on which interface functions are called and through which other data are obtained
  - *Tag:* named datum annotated to Entitys, Entity Sets, Interface
- Instances accessed using opaque (type-less) "handles"

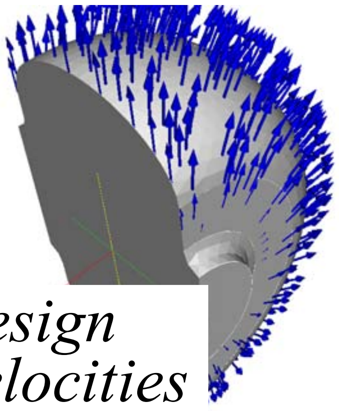
# ITAPS Data Model Usage

## Mesh Partition

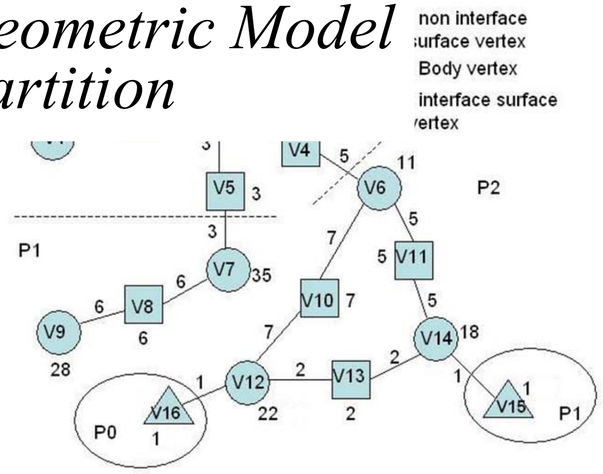


Klystron mesh, SLAC/SNL

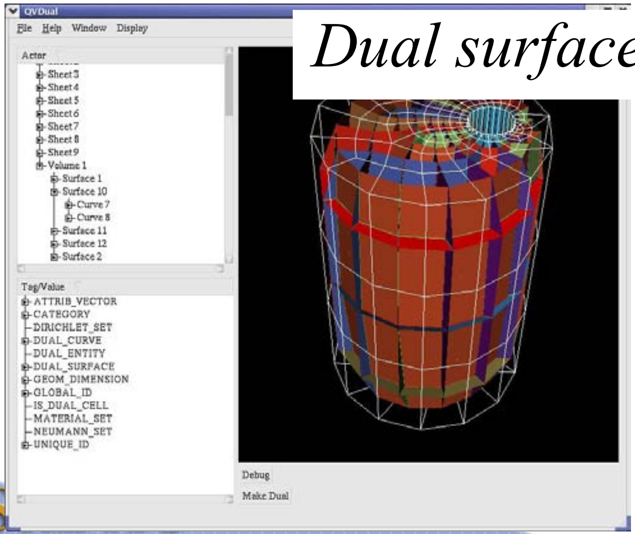
## Design Velocities



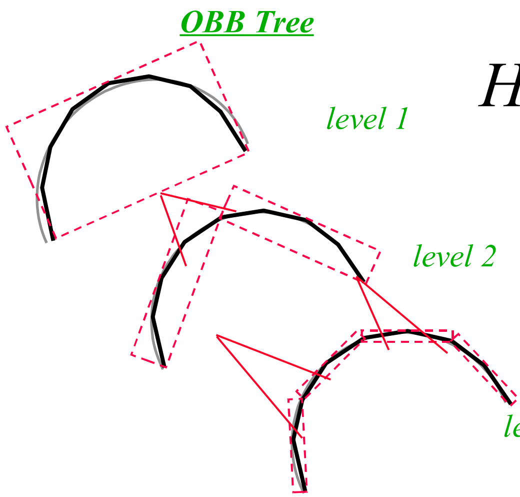
## Geometric Model Partition



## Dual surfaces



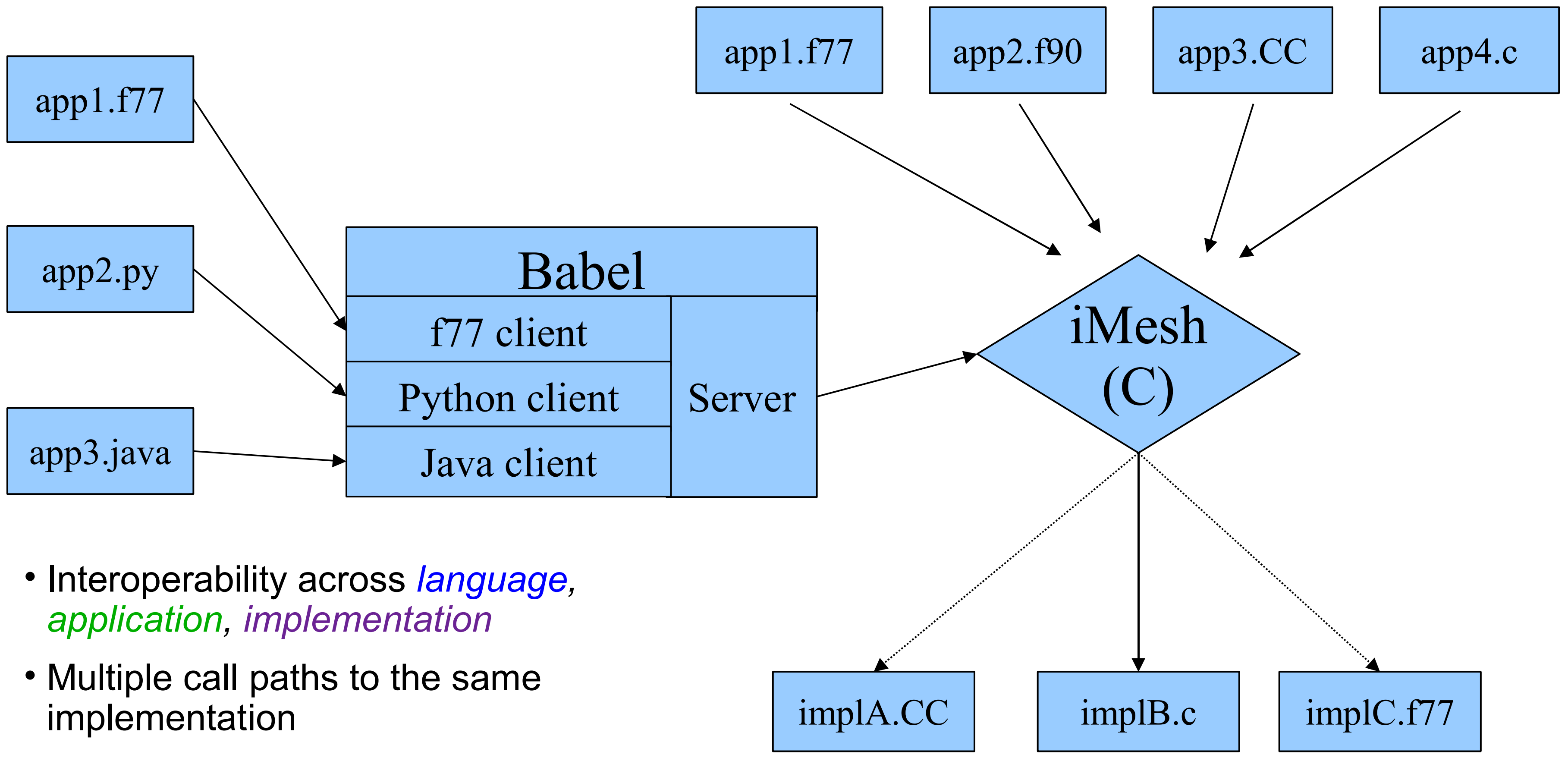
## Hierarchical OBB Tree



## ITAPS Data Model (cont.)

- Important enumerated types:
  - EntityType (iBase_VERTEX, EDGE, FACE, REGION)
  - EntityTopology (iMesh_POINT, LINE, TRI, QUAD, ...)
  - StorageOrder (iBase_BLOCKED, INTERLEAVED)
  - TagDataType (iBase_INTEGER, DOUBLE, ENTITY_HANDLE)
  - ErrorType (iBase_SUCCESS, iBase_FAILURE, ...)
- Enumerated type & function names have iBase, iMesh, iGeom, other names prepended

# ITAPS Interfaces Designed for Interoperability



- Interoperability across *language*, *application*, *implementation*
- Multiple call paths to the same implementation
- Efficiency preserved using direct, C-based interface

# Simple Example: HELLO iMesh (C++)

- Simple, typical application which 1) Instantiates iMesh interface, 2) Reads mesh from disk, 3) Reports # entities of each dimension

```
#include <iostream>
#include "iMesh.h"

int main( int argc, char *argv[] )
{
    // create the Mesh instance
  char *options = NULL;
  iMesh_Instance mesh;
  int ierr, options_len = 0;
  iMesh_newMesh(options, &mesh, &ierr,          1
                options_len);

    // load the mesh
  iMesh_load(mesh, argv[1], options, &ierr,     2
             strlen(argv[1]), options_len);

    // report the number of elements of each dimension
  for (int dim = iBase_VERTEX; dim <= iBase_REGION; dim++) {
    int numd;
    iMesh_getNumOfType(mesh, 0, dim, &numd, &ierr);     3
    std::cout << "Number of " << dim << "d elements = "
              << numd << std::endl;
  }
  return true;}
```

- Makefile:

```
include ../../iMesh-Defs.inc

HELLOiMesh: HELLOiMesh.o  ${iMesh_FILES}
     $(CXX)  $(CXXFLAGS) -o $@ HELLOiMesh.o \
          ${iMesh_LIBS}

.cpp.o:
     ${CXX} -c ${CXXFLAGS} $iMesh_INCLUDES} $<
```

***Note: no error checking here for brevity, but there should be in your code!!!***

## ITAPS API's: Argument Handling Conventions

- ITAPS API's are C-like and can be called directly from C, Fortran, C++
- Arguments pass by value (in) or reference (inout, out)
  - Fortran: use %VAL extension
- Memory allocation for lists done in application *or* implementation
  - If inout list comes in allocated, length must be long enough to store results of call
  - By definition, allocation/deallocation done using C malloc/free; application required to free memory returned by implementation
  - Fortran: Use “cray pointer” extension (equivalences to normal f77 array)
- Handle types typedef'd to size_t (iBase_EntityHandle, iBase_EntitySetHandle, iBase_TagHandle, iMesh_Instance)
- Strings: char*, with length passed by value after all other args
- Enum's: *values (iBase_SUCCESS, etc.) available for comparison operations, but passed as integer arguments*
  - *Fortran: named parameters*

## Argument Handling Conventions

| Issue | C | FORTRAN |
|---|---|---|
| Function Names | iXxxx_ prefix | Same as C |
| Interface Handle | Typedef'd to size_t, as type iXxxx_Instance; instance handle is 1st argument to all functions | #define'd as type Integer; ha is 1st argument to all functio |
| Enumerated Variables | All arguments integer-type instead of enum-type; values from enumerated types | Same, with enum values def FORTRAN parameters |
| Entity, Set, Tag Handles | Typedef'd as size_t; typedef types iBase_EntityHandle, iBase_EntitySetHandle, iBase_TagHandle | #define'd as type Integer |

## iMesh API Summary

- Logically arranged into interfaces, but not explicitly arranged as such in C
  - See iMesh.h or iMesh.sidl
- Basic (Mesh): load, save, getEntities, getNumOfType/Topo, getAllVtxCoordinates, getAdjacencies
- Entity: init/get/reset/endEntIter (iterators), getEntType/Topo, getEntAdj, getVtxCoord
- Arr (Entity arrays): like Entity, but for arrays of entities
- Modify: createVtx/Ent, setVtxCoord, deleteEnt

# Imesh API Summary (cont.)

- From iBase:
  - Tag: create/destroyTag, getTagName/SizeBytes/SizeValues/Handle/Type
  - EntTag: get/setData, get/setInt/Dbl/EHData, getAllTags, rmvTag
  - ArrTag: like EntTag, but for arrays of entities
  - SetTag: like EntTag, but for entity sets
  - EntSet: create/destroyEntSet, add/remove entity/entities/set, isEnt/EntSetContained
  - SetRelation: add/rmvPrntChld, isChildOf, getNumChld/Prnt, getChldn/Prnts
  - SetBoolOps: subtract, intersect, unite
- iBase-inherited function names still start with 'iMesh_' to avoid name collision with other iBase-inherited interfaces (iGeom, iRel, etc.)

# Slightly More Complicated Example: FindConnect (C)

```
#include <iostream>
#include "iMesh.h"

typedef void* EntityHandle;

int main( int argc, char *argv[] )
{
    // create the Mesh instance
  iMesh_Instance mesh;
  int ierr;
  iMesh_newMesh("", &mesh, &ierr, 0);

    // load the mesh
  iMesh_load(mesh, 0, "125hex.vtk", "",
    &ierr, 10, 0);

    // get all 3d elements
  iMesh_EntityHandle *ents;
1 int ents_alloc = 0, ents_size;
  iMesh_getEntities(mesh, 0, iBase_REGION,
                    iMesh_ALL_TOPOLOGIES,
                    &ents, &ents_alloc,
                    &ents_size, &ierr);

  int vert_uses = 0;

    // iterate through them
  for (int i = 0; i < ents_size; i++) {
      // get connectivity
    iBase_EntityHandle *verts;                2
    int verts_alloc = 0, verts_size;

    iMesh_getEntAdj(mesh, ents[i], iBase_VERTEX,
                    &verts, &verts_alloc, &verts_size,
                    &ierr);
      // sum number of vertex uses
    vert_uses += verts_size;
    free(verts);          3
  }

    // now get adjacencies in one big block
  iBase_EntityHandle *allv;
  int *offsets;
  int allv_alloc = 0, allv_size,
    offsets_alloc = 0, offsets_size;
  iMesh_getEntArrAdj(mesh, ents, ents_size,
       iBase_VERTEX,
       &allv, &allv_alloc, &allv_size,
       &offsets, &offsets_alloc, &offsets_size,
       &ierr);

    // compare results of two calling methods
  if (allv_size != vert_uses)
    std::cout << "Sizes didn't agree" << std::endl;
  else
    std::cout << "Sizes did agree" << std::endl;

  return true;
}
```

## FindConnect (C) Notes

- Typical inout list usage
  - X *list,  int list_alloc = 0, int list_size
  - Setting list_alloc to zero *OR list = NULL indicates list is unallocated, so it will be allocated inside iMesh_getEntities*
  - Addresses of these parameters passed into iMesh_getEntities

2. Inout list declared inside 'for' loop

3. Memory de-allocated inside loop

# Slightly More Complicated Example: FindConnect (Fortran)

```
      program findconnect
#include "iMesh_f.h"

c declarations
      iMesh_Instance mesh
      integer*8 ents
      pointer (rpents, ents(0:*))
      integer*8 rpverts, rpallverts, ipoffsets
1     pointer (rpverts, verts(0:*))
      pointer (rpallverts, allverts(0:*))
      pointer (ipoffsets, ioffsets(0,*))
      integer ierr, ents_alloc, ents_size
      integer verts_alloc, verts_size
      integer allverts_alloc, allverts_size
      integer offsets_alloc, offsets_size

c create the Mesh instance
      call iMesh_newMesh("MOAB", mesh, ierr)

c load the mesh
      call iMesh_load(%VAL(mesh), %VAL(0),
     1    "125hex.vtk", "", ierr)

c get all 3d elements
      ents_alloc = 0
      call iMesh_getEntities(%VAL(mesh),
     1     %VAL(0), %VAL(iBase_REGION),
2    1     %VAL(iMesh_ALL_TOPOLOGIES),
     1     rpents, ents_alloc, ents_size,
     1     ierr)
```

```
      ivert_uses = 0

c iterate through them;
      do i = 0, ents_size-1
c get connectivity                                   3
        verts_alloc = 0
        call iMesh_getEntAdj(%VAL(mesh),
     1  %VAL(ents(i)), %VAL(iBase_VERTEX),
     1  rpverts, verts_alloc, verts_size, ierr)
c sum number of vertex uses
        vert_uses = vert_uses + verts_size
        call free(rpverts)
      end do                                          4

c now get adjacencies in one big block
      allverts_alloc = 0
      offsets_alloc = 0
      call iMesh_getEntArrAdj(%VAL(mesh),
     1  %VAL(rpents), %VAL(ents_size),
     1  %VAL(iBase_VERTEX), rpallverts,
     1  allverts_alloc, allverts_size, ipoffsets,
     1  offsets_alloc, offsets_size, ierr)

c compare results of two calling methods
      if (allverts_size .ne. vert_uses) then
         write(*,'("Sizes didn''t agree!")')
      else
         write(*,'("Sizes did agree!")')
      endif

      end
```

## FindConnect (Fortran) Notes

1. Cray pointer usage
   - "pointer" (rpverts, rpoffsets, etc.) declared as type integer
     - Careful – integer*8 or integer*4, 64- or 32-bit
   - "pointee" (verts, ioffsets, etc.) implicitly typed or declared explicitly
   - pointer statement equivalences pointer to start of pointee array
   - pointee un-allocated until explicitly allocated
2. Set allocated size (ents_alloc) to zero to force allocation in iMesh_getEntities; arguments passed by reference by default, use %VAL extension to pass by value; pointers passed by reference by default, like arrays
3. Allocated size set to zero to force re-allocation in every iteration of do loop
4. Use C-based free function to de-allocate memory

# FindConnect Makefile

```
include /sandbox/tautges/MOAB/lib/iMesh-Defs.inc

FC = ${iMesh_FC}
CXX = g++
CC = gcc

CXXFLAGS = -g
CFLAGS = -g
FFLAGS = -g
FLFLAGS = -g  ${iMesh_FCFLAGS}

FindConnectS: FindConnectS.o
$(CXX)  $(CXXFLAGS) -o $@ FindConnect.o ${iMesh_SIDL_LIBS}

FindConnectC: FindConnectC.o
$(CC)  $(CFLAGS) -o $@ FindConnectC.o ${iMesh_LIBS}

FindConnectF: FindConnectF.o
$(FC) -o $@ FindConnectF.o $(FLFLAGS) ${iMesh_LIBS}

.cpp.o:
     ${CXX} -c ${CXXFLAGS} ${iMesh_INCLUDES} ${iMesh_SIDL_INCLUDES} $<
.cc.o:
     ${CC} -c ${CFLAGS} ${iMesh_INCLUDES} ${iMesh_SIDL_INCLUDES} $<
.F.o:
     ${FC} -c ${FFLAGS} ${iMesh_INCLUDES} $<
```

## Using MOAB/ITAPS on MCS Systems

- On MCS Systems: ~tautges/MOAB, ~tautges/MOABpar
  - MOABpar compiled for parallel operation
- If you want to build your own:
  - Will need hdf5, and if you want to read Exodus files, netcdf (incl. C++ library for netcdf)
  - svn: https://svn.mcs.anl.gov/repos/ITAPS/MOAB/trunk
    - Will need relatively up-to-date autotools stack, follow directions in README.CONFIGURE
  - tarball: http://www.mcs.anl.gov/~tautges/downloads/MOAB-current.tar.gz
  - Follow building directions in README
- Example code from this course on gnep in /sandbox/tautges/shortcourse/Examples (or download from http://www.mcs.anl.gov/~tautges/downloads/shortcourse_examples.tar.gz)

# ListSetsNTags Example

- Read in a mesh
- Get all sets
- For each set:
  - Get tags on the set and names of those tags
  - If tag is integer or double type, also get value
  - Print tag names & values for each set

- Various uses for sets & tags, most interesting ones involve both together
  - Geometric topology
  - Boundary conditions
  - Processor decomposition

# ListSetsNTags Example (SIDL/C++)

```
#include "iBase.hh"
#include "iMesh_SIDL.hh"

typedef void* iBase_EntityHandle;
typedef void* iBase_EntitySetHandle;
typedef void* iBase_TagHandle;

int main( int argc, char *argv[] )
{
  // Check command line arg
  std::string filename = argv[1];

    // create the Mesh instance
  iMesh::Mesh mesh;
  iMesh_SIDL::MeshSidl::newMesh("", mesh);

    // load the mesh
  string options;
  mesh.load(0, filename, options);

    // get all sets; use EntSet interface
  sidl::array<iBase_EntitySetHandle> sets;
  int sets_size;
  iBase::EntSet mesh_eset = mesh;
  mesh_eset.getEntSets(0, 1,
                       sets, sets_size);

    // iterate through them, checking whether
   they have tags
  iBase::SetTag mesh_stag = mesh;
  for (int i = 0; i < sets_size; i++) {
      // get connectivity
    sidl::array<iBase_TagHandle> tags;
    int tags_size;
    mesh_stag.getAllEntSetTags(sets[i],
                               tags, tags_size);

    if (0 != tags_size) {
      cout << "Set " << sets[i] << ": Tags: ";
        // list tag names on this set
      for (int j = 0; j < tags_size; j++) {
        string tname;
        int int_val;
        double dbl_val;
        mesh_stag.getTagName(tags[j], tname);
        cout << tname;
1       iBase::TagValueType tag_type;
        mesh_stag.getTagType(tags[j], tag_type);
2       if (iBase::TagValueType_INTEGER ==
            tag_type) {
          mesh_stag.getEntSetIntData(sets[i],
                                     tags[j], int_val);
          cout << " (val = " << int_val << "); ";
        }
        else if (iBase::TagValueType_DOUBLE ==
                 tag_type) {
          mesh_stag.getEntSetDblData(sets[i],
                 tags[j], dbl_val);
          cout << " (val = " << dbl_val << "); ";
        }
        else cout << "; ";
      }
    }
    cout << endl;
  }

  return true;
}
```

## ListSetsNTags Example Notes

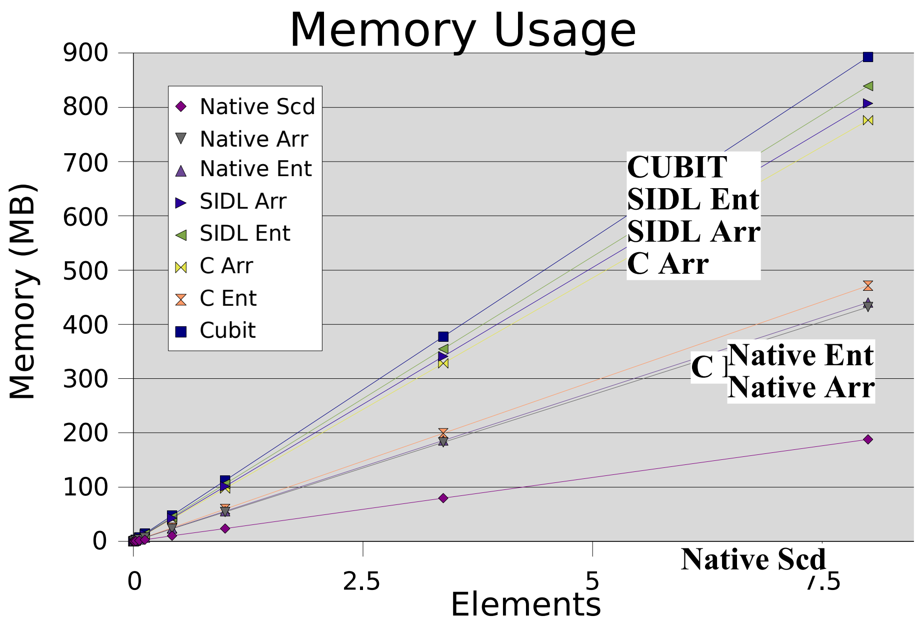
- Enumerated variables declared in SIDL-based code as Iface::enumNAME, e.g. iBase::EntityType or iBase::TagType
- Enumerated variable values appear as Iface::enumNAME_enumVALUE, e.g. iMesh::EntityTopology_TETRAHEDRON or iBase::TagType_INTEGER

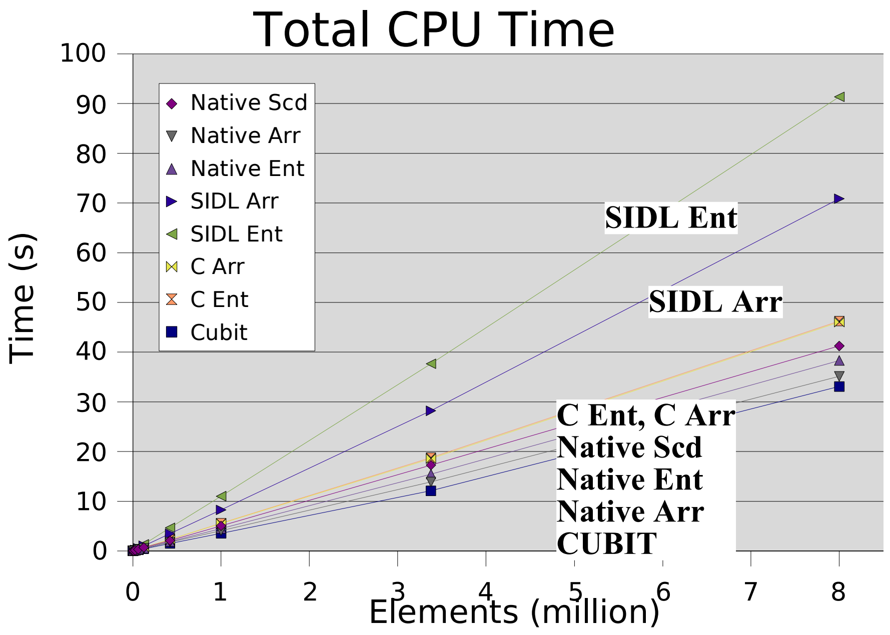
# ListSetsNTags Assignment

- Translate this app to your favorite language (C++, C, Fortran) & call through ITAPS C interface

# Performance

- Large applications balance memory and cpu time performance
- Implementations of iMesh vary on speed vs. memory performance
  - Create, v-E, E-v query, square all-hex mesh
  - Entity- vs. Array-based access
- Compare iMesh (C, SIDL), Native (MOAB), Native Scd (MOAB), CUBIT
  - Ent-, Arr-based access
  - All-hexahedral square structured mesh





# iMesh Review

- Data model consists of 4 basic types: Interface, Entity, Entity Set, Tag
- Applications reference instances of these using opaque handles
- ITAPS interfaces use C-based APIs, for efficiency and interoperability
  - SIDL-based implementation also available, which work through C-based API
- Not covered here:
  - Iterators (intermediate-level, “chunked” access to mesh)
  - Modify (relatively coarse-grained, basically create and delete whole entities)
  - Set parent-child links

## MOAB

- MOAB entity types include entity set & are sorted by dimension (defined in MBEntityType.h)
  - MBVERTEX, MBEDGE, MBTRI, MBQUAD, MBPOLYGON, MBTET, ..., MBENTITYSET
- MBEntityHandle properties
  - MOAB entity handle is an integer type, embeds entity type, proc rank, entity id
  - List of contiguous handles can be stored in ranges, const-space lists
  - Sort by type, dimension
  - Set booleans (intersect, union, subtract) very fast on ranges
  - Config option to use 64-bit handles on 32-bit apps if id space is a concern
- MBRange class: series of sub-ranges of entity handles
- MOAB functionality accessed through MBInterface, an abstract base class
  - Most functionality similar to what's in iMesh, plus a little more

# MOAB Tools

- Some functions support set booleans implicitly:

```
virtual MBErrorCode get_adjacencies(const MBRange &from_entities,
                 const int to_dimension, const bool create_if_missing,
                 MBRange &adj_entities,
                 const int operation_type = MBInterface::INTERSECT);
```

  - For multiple from_entities, adj_entities will be intersection of queries on each from_entity
    - To get common vertices between two entities, call with to_dimension=0 and MBInterface::INTERSECT
    - To get all vertices used by group of entities, call with to_dimension=0 and MBInterface::UNION
- MBSkinner: gets skin (bounding (d-1)-dimensional entities) of a set of entities
- IO: format designated by file extension;
  - CUBIT .cub (R), Exodus (.g, .exoII) (RW), vtk (.vtk) (RW), native HDF5 format (.h5m, .mhdf) (RW)
  - Use .h5m/.mhdf to save everything MOAB can represent in data model (sets, tags, set parents/children, polygons/polyhedra, etc.)

# MOAB Parallel

- Configure with --with-mpi= option
- Parallel read (bcast_delete, read_delete) working
  - Can use any “covering” set of sets as partition
  - Read method designated with options to MOAB's load_file function, e.g. “PARALLEL=BCAST_DELETE;PARALLEL_PARTITION=MATERIAL_SET” “PARALLEL=BCAST_DELETE;PARALLEL_PARTITION=GEOM_DIMENSION;PARTITION_VAL=3;PARTITION_DISTRIBUTE”
- Other classes
  - MBParallelComm: pass entities/tags/sets between processors, define communicator
  - MBParallelData: convenience functions for getting partition, interface entities
- Relevant tags (defined in MBParallelConventions.h):
    - PARALLEL_SHARED_PROC: 2 ints, ranks of sharing procs on 2-proc interface
    - PARALLEL_SHARED_PROCS: N ints, ranks of sharing procs when > 2 procs share iface
    - PARALLEL_OWNER: rank of owning processor for interface entities, sets
    - PARALLEL_GHOST: rank of owning processor for ghost entities, sets
    - PARALLEL_GID: global id, used to match vertices, other entities

# MOAB Parallel Example

(condensed from mbparallelcomm_test.cpp in MOAB source dir)

```
int main(int argc, char **argv)
{
  int err = MPI_Init(&argc, &argv);
  int nprocs, rank;
  err = MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
  err = MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    // create MOAB instance based on that
  MBInterface *mbImpl = new MBCore(rank, nprocs);
  MBParallelComm *pcomm = new MBParallelComm(mbImpl);

    // read a file in parallel
  const char *options =
   "PARALLEL=BCAST_DELETE;PARTITION=GEOM_DIMENSION;PARTITION_VAL=3;PARTITION_DI
   STRIBUTE";
  MBEntityHandle file_set;
  MBErrorCode result = mbImpl->load_file(filename, file_set, options);

    // resolve shared vertices
  result = pcomm->resolve_shared_ents();

    // get shared vertices on this proc
  MBRange shared_ents;
  result = pcomm->get_shared_entities(0, shared_ents);

  MPI_Finalize();
  return 0;
}
```

# ITAPS Interfaces Best Practices

- Use C-based interface where possible, for efficiency
- Pre-allocate memory in application or re-use memory allocated by implementation
  - E.g. getting vertices adjacent to element – can use static array, or application-native storage
- Take advantage of implementation-provided capabilities to avoid re-inventing
  - Partitioning, IO, parallel communication, (parallel) file readers
- Be careful about integer*8, integer*4, memory corruption in Fortran apps (valgrind is your friend)
- Implement iMesh on top of your data structure
  - Take advantage of tools which work on iMesh API
- Let us help you
  - Not all the tricks can be easily described and may not be self-evident

## Best Practices: MOAB vs. ITAPS

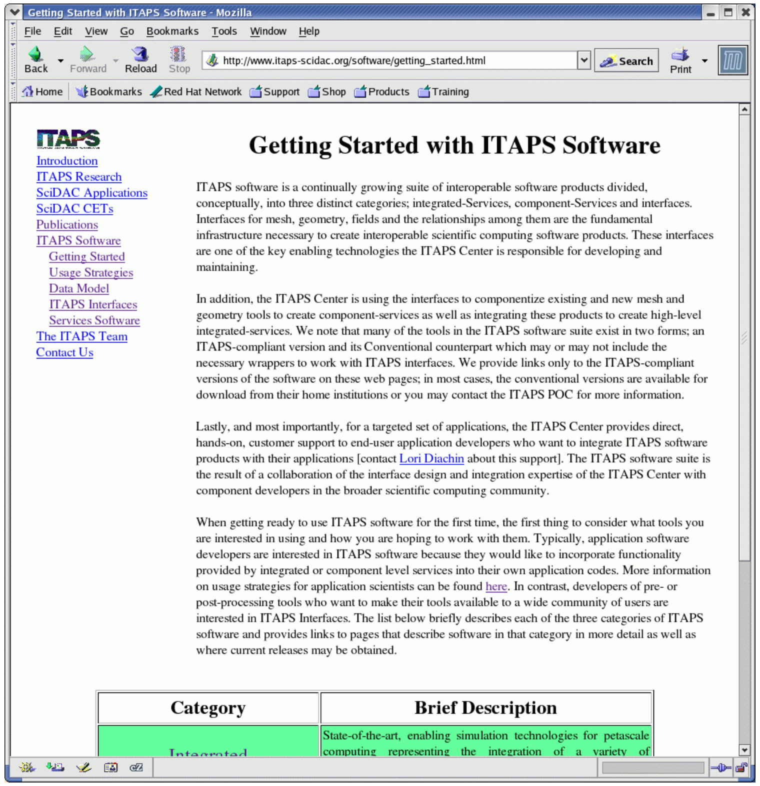
- MOAB has a few functions not included in ITAPS
  - get_entities_by_type_and_tag
  - parallel functions
- When handling & manipulating large lists of entity handles, using MBRange can save lots of memory
- Lower overhead (~5-10%) with MOAB native interface
- NO MOAB Fortran or C interface, only C++

## Obtaining the ITAPS Software

# ITAPS Software Web Pages

http://www.itaps- scidac.org/software/

- Provides help getting started
- Usage strategies
- Data model description
- Access to interface specifications, documentation, implementations
- Access to compatible services software


Getting Started with ITAPS Software - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.itaps-scidac.org/software/getting_started.html
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training

Introduction
ITAPS Research
SciDAC Applications
SciDAC CETs
Publications
ITAPS Software
Getting Started
Usage Strategies
Data Model
ITAPS Interfaces
Services Software
The ITAPS Team
Contact Us

## Getting Started with ITAPS Software

ITAPS software is a continually growing suite of interoperable software products divided, conceptually, into three distinct categories; integrated-Services, component-Services and interfaces. Interfaces for mesh, geometry, fields and the relationships among them are the fundamental infrastructure necessary to create interoperable scientific computing software products. These interfaces are one of the key enabling technologies the ITAPS Center is responsible for developing and maintaining.

In addition, the ITAPS Center is using the interfaces to componentize existing and new mesh and geometry tools to create component-services as well as integrating these products to create high-level integrated-services. We note that many of the tools in the ITAPS software suite exist in two forms; an ITAPS-compliant version and its Conventional counterpart which may or may not include the necessary wrappers to work with ITAPS interfaces. We provide links only to the ITAPS-compliant versions of the software on these web pages; in most cases, the conventional versions are available for download from their home institutions or you may contact the ITAPS POC for more information.

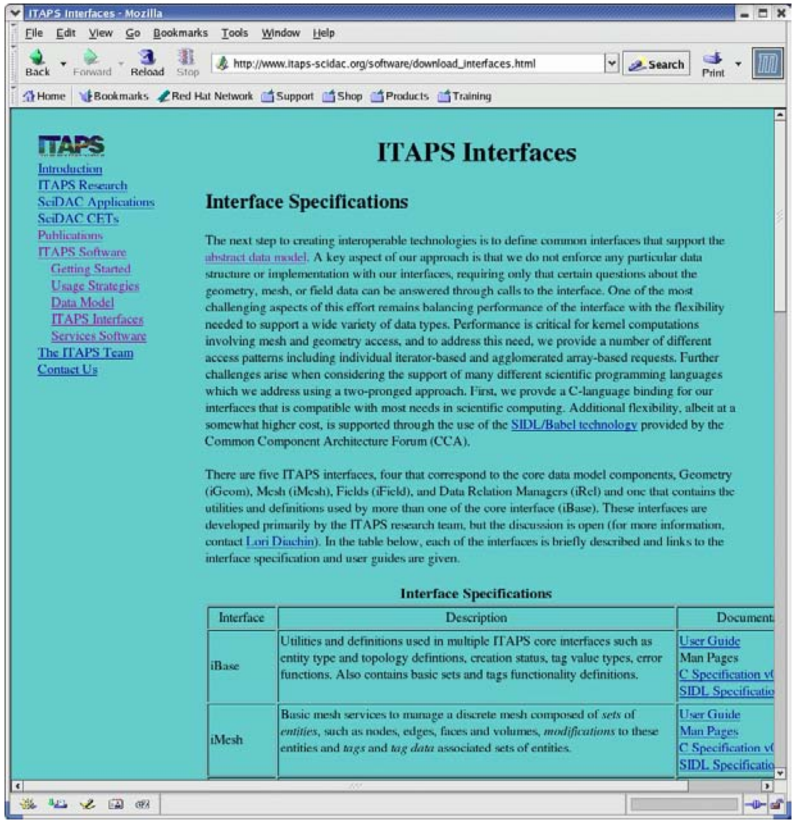
Lastly, and most importantly, for a targeted set of applications, the ITAPS Center provides direct, hands-on, customer support to end-user application developers who want to integrate ITAPS software products with their applications [contact Lori Diachin about this support]. The ITAPS software suite is the result of a collaboration of the interface design and integration expertise of the ITAPS Center with component developers in the broader scientific computing community.

When getting ready to use ITAPS software for the first time, the first thing to consider what tools you are interested in using and how you are hoping to work with them. Typically, application software developers are interested in ITAPS software because they would like to incorporate functionality provided by integrated or component level services into their own application codes. More information on usage strategies for application scientists can be found here. In contrast, developers of pre- or post-processing tools who want to make their tools available to a wide community of users are interested in ITAPS Interfaces. The list below briefly describes each of the three categories of ITAPS software and provides links to pages that describe software in that category in more detail as well as where current releases may be obtained.

| Category | Brief Description |
|---|---|
| Integrated | State-of-the-art, enabling simulation technologies for petascale computing representing the integration of a variety of |

# Interface Software Access

- Links to the interface user guides and man pages where available
- Links to the C-binding and SIDL-binding files
- Links to implementations for iMesh, iGeom, iRel
  - Version 0.7 compatible software
  - Links to the home pages for more information
- Simple examples, compliance testing tools and build skeletons coming soon



## ITAPS Interfaces

### Interface Specifications

The next step to creating interoperable technologies is to define common interfaces that support the abstract data model. A key aspect of our approach is that we do not enforce any particular data structure or implementation with our interfaces, requiring only that certain questions about the geometry, mesh, or field data can be answered through calls to the interface. One of the most challenging aspects of this effort remains balancing performance of the interface with the flexibility needed to support a wide variety of data types. Performance is critical for kernel computations involving mesh and geometry access, and to address this need, we provide a number of different access patterns including individual iterator-based and agglomerated array-based requests. Further challenges arise when considering the support of many different scientific programming languages which we address using a two-pronged approach. First, we provde a C-language binding for our interfaces that is compatible with most needs in scientific computing. Additional flexibility, albeit at a somewhat higher cost, is supported through the use of the SIDL/Babel technology provided by the Common Component Architecture Forum (CCA).

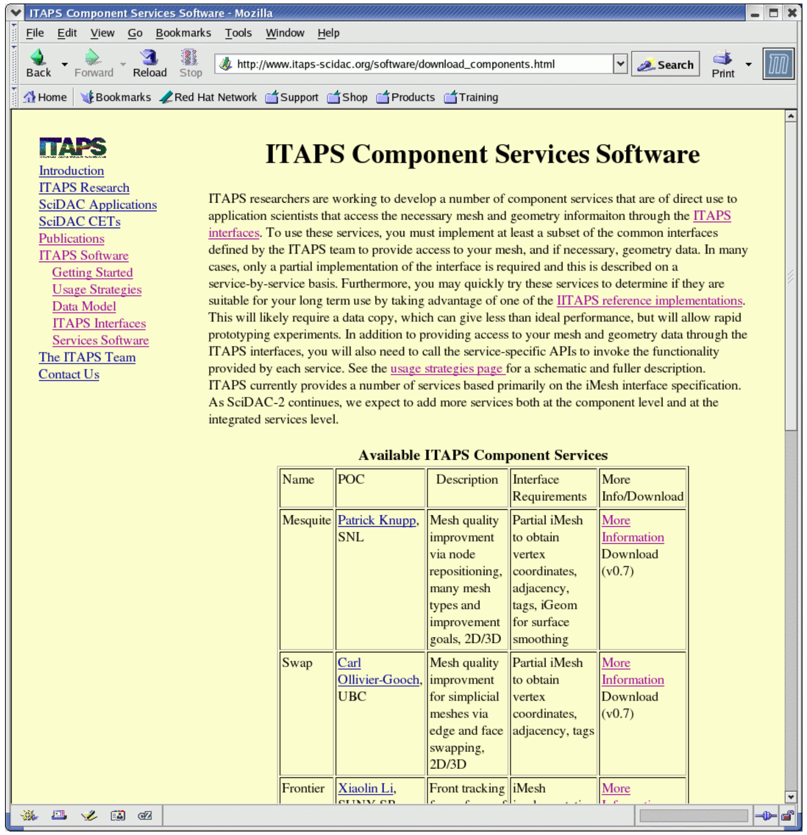
There are five ITAPS interfaces, four that correspond to the core data model components, Geometry (iGeom), Mesh (iMesh), Fields (iField), and Data Relation Managers (iRel) and one that contains the utilities and definitions used by more than one of the core interface (iBase). These interfaces are developed primarily by the ITAPS research team, but the discussion is open (for more information, contact Lori Diachin). In the table below, each of the interfaces is briefly described and links to the interface specification and user guides are given.

**Interface Specifications**

| Interface | Description | Document |
|---|---|---|
| iBase | Utilities and definitions used in multiple ITAPS core interfaces such as entity type and topology defintions, creation status, tag value types, error functions. Also contains basic sets and tags functionality definitions. | User Guide<br>Man Pages<br>C Specification v<br>SIDL Specificatio |
| iMesh | Basic mesh services to manage a discrete mesh composed of *sets* of *entities*, such as nodes, edges, faces and volumes, *modifications* to these entities and *tags* and *tag data* associated sets of entities. | User Guide<br>Man Pages<br>C Specification v<br>SIDL Specificatio |

# Services Software Access

- Links to the services built on the ITAPS interfaces
- Currently or very soon to be available
  - Mesquite (C, SIDL)
  - Zoltan (C, SIDL)
  - Swapping (SIDL)
  - Frontier (SIDL)
  - VisIt Plug In (C, SIDL)
- Links to home pages for more information
- Instructions for build and links to supporting software



ITAPS Component Services Software

ITAPS researchers are working to develop a number of component services that are of direct use to application scientists that access the necessary mesh and geometry informaiton through the ITAPS interfaces. To use these services, you must implement at least a subset of the common interfaces defined by the ITAPS team to provide access to your mesh, and if necessary, geometry data. In many cases, only a partial implementation of the interface is required and this is described on a service-by-service basis. Furthermore, you may quickly try these services to determine if they are suitable for your long term use by taking advantage of one of the IITAPS reference implementations. This will likely require a data copy, which can give less than ideal performance, but will allow rapid prototyping experiments. In addition to providing access to your mesh and geometry data through the ITAPS interfaces, you will also need to call the service-specific APIs to invoke the functionality provided by each service. See the usage strategies page for a schematic and fuller description. ITAPS currently provides a number of services based primarily on the iMesh interface specification. As SciDAC-2 continues, we expect to add more services both at the component level and at the integrated services level.

Available ITAPS Component Services

| Name | POC | Description | Interface Requirements | More Info/Download |
|---|---|---|---|---|
| Mesquite | Patrick Knupp, SNL | Mesh quality improvment via node repositioning, many mesh types and improvement goals, 2D/3D | Partial iMesh to obtain vertex coordinates, adjacency, tags, iGeom for surface smoothing | More Information Download (v0.7) |
| Swap | Carl Ollivier-Gooch, UBC | Mesh quality improvment for simplicial meshes via edge and face swapping, 2D/3D | Partial iMesh to obtain vertex coordinates, adjacency, tags | More Information Download (v0.7) |
| Frontier | Xiaolin Li, | Front tracking | iMesh | More |

## MOAB Software Web Pages

http://trac.mcs.anl.gov/projects/ITAPS/wiki/MOAB

- General information
- Browse svn repo
- FAQ
- Pointers to mailing list archives