# IMPROVED QUASI-STATIC METHODS FOR TIME-DEPENDENT NEUTRON DIFFUSION AND IMPLEMENTATION IN RATTLESNAKE

A Thesis

by

ZACHARY MERRITT PRINCE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

| | |
|---|---|
| Chair of Committee, | Jean C. Ragusa |
| Committee Members, | Jim E. Morel |
| | Bojan Popov |
| Head of Department, | Yassin Hassan |

May 2017

Major Subject: Nuclear Engineering

Copyright 2017 Zachary Merritt Prince

# ABSTRACT

Transient reactor simulations are a chronically formidable challenge due to the computational rigor of evaluating the neutron transport equation, as well as its coupling with other physical phenomena. The goal of this thesis research is to mitigate the computational expense of these simulations by investigating and developing the improved quasi-static method (IQS). IQS is a rigorous space/time multiscale approach whereby the neutron flux is represented by a time-dependent amplitude and a time-space-energy dependent shape. The objective of the IQS factorization is to evaluate amplitude and shape on different time scales in order to reduce computational burden associated with solving the multi-dimensional flux equations, while maintaining solution accuracy. IQS factorization leads to a nonlinear system of equations that requires iteration of shape and amplitude. Furthermore, reactor simulations often require coupling to additional multiphysics components for temperature feedback. These additional physics may evolve on different time scales, and the applicability of IQS in multiphysic simulations needs to be evaluated from this perspective as well.

The objectives of this research are to establish IQS performance with various iteration techniques, validate time step convergence of IQS, and apply IQS to multiphysics simulation. IQS iteration techniques involve fixed-point (Picard) iteration with various convergence criteria and Newton iteration, namely preconditioned Jacobian-free Newton Krylov (PJFNK) method. Nonlinear convergence of each of these techniques is investigated. Validation of IQS with analysis of time step convergence is vital for implementation of time adaptive methods and error prediction. The time derivative of the shape function is discretized through fourth order using implicit-Euler, Crank-Nicolson, backward difference formulae (BDF), and singly-diagonally-implicit Runge-Kutta (SDIRK) methods. IQS application to multiphysics simulations involves its implementation into the Rattlesnake/MOOSE framework. These simulations allow insight into the performance of IQS for full transient reactor simulations.

The results of the iteration convergence analysis show that the most rigorous and comprehensive iteration technique is fixed-point iteration with consistency in the IQS uniqueness specification as the convergence criteria. This iteration technique revealed the need for analytical treatment of the precursor equation for proper convergence. For time step convergence analysis, IQS was applied to a one-dimensional prototype example, as well as the TWIGL and LRA benchmark. The prototype results show that IQS has proper error convergence through fourth order discretization schemes. The TWIGL and LRA benchmark results show that IQS has proper convergence for implicit Euler, second-order BDF, and Crank-Nicolson schemes, validating IQS for more complex problems. For multiphysics simulation, IQS was applied to the LRA benchmark and a full core TREAT model. The results show that integration of temperature into the quasi-static process made considerable improvement to IQS performance. These results and conclusions helped gain insight into the behavior of IQS and furthered its development in complete transient reactor models.

## DEDICATION

To my mother, my father, my grandmother, and my grandfather.

## CONTRIBUTORS AND FUNDING SOURCES

### Contributors

This work was supported by a thesis committee consisting of Professors Jean Ragusa and Jim Morel of the Department of Nuclear Engineering and Professor Bojan Popov of the Department of Mathematics.

All other work conducted for the thesis was completed by the student independently with the direction of his adviser Dr. Jean Ragusa.

### Funding Sources

Graduate study was supported by the Endowed Doctoral Fellowship from Texas A&M University, Idaho National Laboratory, and Integrated University Program Fellowship from the Department of Energy.

NOMENCLATURE

| | |
|---|---|
| IQS | Improved Quasi-Static Method |
| INL | Idaho National Laboratory |
| MOOSE | Multiphysics Object-Oriented Simulation Environment |
| DOE | Department of Energy |
| NEAMS | Nuclear Energy Advanced Modeling and Simulation |
| PRKE | Point Reactor Kinetics Equation |
| FEM | Finite Element Method |
| P-C | Predictor-Corrector |
| TREAT | Transient Reactor Testing Facility |
| BDF | Backward Difference Formula |
| SDIRK | Singly-Diagonally-Implicit Rung-Kutta Method |
| PDE | Partial Differential Equation |
| ODE | Ordinary Differential Equation |
| GMRES | Generalized Minimal Residual Method |
| PJFNK | Preconditioned Jacobian-Free Newton-Krylov Method |
| DT2 | Step Doubling Time Adaptation Method |

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

# 1. INTRODUCTION

## 1.1 Background on Transient Reactor Testing

The primary purpose of transient reactor testing is the safety and performance analysis of fuel and other reactor components. Since the March 2011 events at the Fukushima Daiichi Nuclear Power Plant in Japan, there has been significant governmental demand for the development of accident tolerant fuels in light water reactors (LWR) [6]. Currently, two facilities are capable of testing fuels under these extreme conditions: the Annular Core Research Reactor (ACRR) at Sandia National Laboratory (SNL) and the Transient Reactor Testing Facility (TREAT) at Idaho National Laboratory (INL) [7]. TREAT had been put in stand-by status in 1994, but in 2014 a Final Environmental Assessment by the Department of Energy (DOE) approved the restart of the facility in order to resume nuclear fuel testing[8].

TREAT is an graphite-moderated, air-cooled, thermal reactor which began operation in 1959. The reactor was designed to subject fuels and experimental apparatus to extreme power pulses, for fast reactor type components [9]. The primary purpose for the restart of the facility is to use these pulses to simulate accident scenarios and test the integrity of accident tolerant LWR fuels. TREAT is expected to resume experimentation and testing by 2020. In addition to the substantial upgrades of TREAT's electronic and mechanical systems during the restart, advance computer models of the reactor are in development. These models have a mutualistic relationship with TREAT: validating fuel models with experimentation and using reactor models to streamline experimental procedures. Development of these models has renewed a significant interest in implementing transient reactor simulation methods in modern computational tools.

Transient reactor simulation methods have been developed for several decades. The revitalization of TREAT and other transient reactor testing efforts have brought special attention to the methods' implementation in modern computational tools. The DOE, through its NEAMS (Nuclear Energy Advanced Modeling and Simulation) program, has especially sought the development of transient multiphysics capability in INL's MOOSE (Multiphysics Object-Oriented Simulation Environment). This research intends to investigate and develop the improved quasi-static method (IQS) with these modern tools. IQS is a transient neutronics method intended to improve computational efficiency. Improving this efficiency is vital for reactor simulations that would otherwise be overbearing to evaluate. In order to apply IQS to full transient reactor models, implementation of multiphysics is required. The following two sections provide an overview of time-dependent neutronics and multiphysics solution methods.

## 1.2 Time-Dependent Neutron Transport

Transient modeling of nuclear reactors has been a chronically formidable task due to its computationally expensive nature. The difficulty in transient reactor modeling is due to:

1. the high-dimensionality of the phase-space for the governing equations that describe the flux of neutrons (6-D+time for multigroup neutron transport and 4-D+time for multigroup neutron diffusion), and
2. the fact that the time discretization has to be implicit, which leads to a stiff system of equations.

The transport equation has seven independent variables: space ($\vec{r}$), energy ($E$), direction ($\vec{\Omega}$), and time ($t$) [10]. Nuclear reactor simulations often utilize the neutron diffusion approximation, which carefully eliminates the dependence on direction. For the purpose of this research, neutron diffusion is discussed exclusively.

For computational purposes, the neutron diffusion equation is discretized in space, energy, and time. There are several viable discretization schemes for each of these variables. In this research effort, FEM is used for space discretization and multigroup for energy discretization [11, 10]. Since the purpose of this research involves temporal dependence, discussion of tempoeral discretization is kept general at this stage. Most reactor computation frameworks discretize the time variable of flux directly using a multitude of schemes (Euler, Crank-Nicholson, Runge-Kutta, etc.). However, the solutions of flux can be particullary stiff due to the speed of neutrons, especially in a pulsing reactor [12]. Therefore, computationally expensive implicit schemes are necessary with many points of discretization (time steps). The following subsections describe two alternate techniques for transient simulation: point reactor kinetics and the improved quasi-static method (IQS). These methods are meant to reduce computational expense of evaluating time-dependent neutron population distributions while maintaining accuracy.

### 1.2.1 Point Reactor Kinetics

A common solution process for neutron kinetics is the evaluation of the point reactor kinetics equations (PRKE) [13]. The derivation of this equation involves factorizing flux into a space-dependent shape and time-dependent amplitude. This factorization immediately creates a large assumption that the spatial variance of the flux is time-independent. The amplitude has the same temporal stiffness as flux; however, the PRKE takes very little computational effort to evaluate because the many variables from the spatial discretization only need to be evaluated for the initital (steady state) distribution. For all time steps in the transient, the spatial shape is assumed fixed and only the PRKE, a small system of ODEs, needs to be evaluated for the time-dependent amplitude.

### 1.2.2 Improved Quasi-Static Method

IQS is a spatial kinetics method that involves factorizing the flux solution into space- and time-dependent components [14, 15, 16, 17, 18]. These components are the flux's amplitude and its shape. Amplitude is only time-dependent, while the shape is both space- and time-dependent (as opposed to the PRKE approach where the shape was constant in time). However, the impetus of the method is the assumption that the shape is only weakly dependent on time. Therefore, the variable for the multi-dimensional diffusion evaluation is expected to be less stiff than the flux itself, without making the assumption that spatial variance is time-independent. As a result, the shape may not require an update at the same frequency as the amplitude function, but only on larger macro-time steps.

Due to the factorization of the flux into a shape and an amplitude, the latter two variables are coupled. Ott in [14] first investigated the coupling of shape and amplitude in a quasi-static nature, but did not include the time derivative in the shape equation. Later, in [19], Ott incorporated the time derivative of shape in the equation, yielding better results; this also led to the technique's name: the Improved Quasi-Static method. The resulting system of equations is nonlinear [15]. Nonlinear problems require an iterative process to evaluate, either fixed-point (Picard) or Newton iteration. Sissaoui et al. [18], Koclas et al. [20], Devooght et al. [16], and Monier [17] all use fixed-point iterative techniques for their IQS simulations, the main difference among them is their criteria for convergence. Devooght et al. in [16] also utilizes a Newton iteration technique.

## 1.3 Time-Dependent Reactor Dynamics

Transient reactor analysis involves coupling neutronics with other physics such as thermal hydraulics, fuel mechanics, etc. to enable temperature feedback in the neutronics simulations. This coupling is known as multiphysics simulation. Multiphysics simulation is often a daunting task because it requires communication of coupled variables that are being evaluated with a myriad of techniques. The coupling is also usually nonlinear requiring Picard or Newton iteration techniques. Programs, such as MOOSE, provide the capability of evaluating, communicating, and iterating different physics.

MOOSE is a multiphysics framework being developed at INL that presents the architecture for physics-based applications [21]. In order to robustly implement IQS in the MOOSE framework for multiphysics simulation, each coupled physics application requires consideration. Rattlesnake is a large application in MOOSE that involves deterministic radiation transport physics. Currently, Rattlesnake is able to solve steady-state, transient, and k-eigenvalue neutron transport and diffusion problems using finite element methods (FEM) [22]. BISON is the application for evaluating fuel temperature and structural properties [23]. RELAP-7 is the main safety analysis application, but focuses on variables involved with hydrodynamic analysis [24]. MAMMOTH is the over-arching application that couples all these physics by recomputing cross-sections for Rattlesnake and source terms for BISON and RELAP-7 [5].

IQS is a manipulation of neutronics variables, so it is solely developed within the Rattlesnake application. However, proper treatment of multiphysics with IQS is vital for computational optimization. The quasi-static nature of IQS can be extended to multiphysics simulation by determining the proper time scales for each physics. Considering TREAT, shape is the slowest varying variable and amplitude is fastest, while fuel temperature is in between. Varying time step sizes for each variable can considerably improve computational efficiency, which is the nature and purpose of IQS application.

## 1.4 Objective

The goal of this research is to continue the investigation and development of the improved quasi-static method for minimizing computation expense for transient reactor simulations, while maintaining accuracy. This goal is pursued with three objectives:

1. Establish IQS performance for various iteration techniques

2. Validate time step convergence for IQS with high order time discretization schemes

3. Apply IQS to multiphysics simulations

Iteration convergence analysis involves investigating iteration techniques for nonlinear problems and convergence criteria. For this objective, IQS is implemented into an easily modifiable, prototype-like code in MATLAB. High order time discretizing schemes were also implemented in the prototyping code to investigate time step convergence. The prototype also guides the development of IQS in the large FEM framework of Rattlesnake/-MOOSE, where more complex examples and benchmarks can be tested. Time step convergence is also analyzed for these examples. Development in Rattlesnake also spawns the opportunity to test IQS with multiphysics, namely temperature feedback and time adaptation. Applying IQS to multiphysics simulations allows insight to the performance of IQS for full transient reactor simulations, particularly for TREAT modeling.

# 2. IMPROVED QUASI-STATIC METHOD

## 2.1 Theory

To derive the IQS equations, the time dependent neutron diffusion equation with delayed neutron precursors are presented by Equations (2.1a) and (2.1b).

$$\frac{1}{v^g}\frac{\partial \phi^g}{\partial t} = \frac{\chi_p^g}{k_{\mathit{eff}}}(1-\beta)\sum_{g'=1}^{G}\nu^{g'}\Sigma_f^{g'}\phi^{g'} + \sum_{g'\neq g}^{G}\Sigma_s^{g'\to g}\phi^{g'} + \sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i - \left(-\nabla\cdot D^g\nabla + \Sigma_r^g\right)\phi^g\ , \quad 1 \le g \le G \tag{2.1a}$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{k_{\mathit{eff}}}\sum_{g=1}^{G}\nu^g\Sigma_f^g\phi^g - \lambda_i C_i\ , \quad 1 \le i \le I \tag{2.1b}$$

where,

| | | |
|---|---|---|
| $\phi^g$ | $=$ | Scalar flux in energy group $g$ |
| $C_i$ | $=$ | Concentration of delayed neutron precursor $i$ |
| $\Sigma_f^g$ | $=$ | Fission cross section in energy group $g$ |
| $k_{eff}$ | $=$ | k-eigenvalue of initial flux evaluation |
| $\Sigma_r^g$ | $=$ | Removal cross section in energy group $g$ |
| $\Sigma_s^{g'\to g}$ | $=$ | Scattering cross section from energy group $g'$ to $g$ |
| $v^g$ | $=$ | Neutron velocity in energy group $g$ |
| $\chi_p^g$ | $=$ | Fission spectrum of prompt neutrons |
| $\chi_{d,i}^g$ | $=$ | Fission spectrum of delayed neutrons from precursor $i$ |
| $\nu^g$ | $=$ | Total number of neutrons per fission |
| $D^g$ | $=$ | Diffusion coefficient in energy group $g$ |

| | | |
|---|---|---|
| $\lambda_i$ | $=$ | Decay constant of precursor $i$ |
| $\beta_i$ | $=$ | Delayed neutron fraction from precursor $i$ |
| $\beta$ | $=$ | Total delayed neutron fraction ($\beta = \sum_{i=1}^{I} \beta_i$) |

Most reactor computation frameworks, including Rattlesnake, discretize the time variable directly with these equations using a multitude of schemes (Implicit Euler, Crank-Nicholson, implicit Runge-Kutta, etc.). In this paper, the method of discretizing Equations (2.1a) and (2.1b) is generally referred to as "implicit discretization". This research intends to improve upon this method by instead implementing the improved quasi-static method (IQS) for neutron kinetics and implement it within a multiphysics setting.

IQS involves factorizing the flux from Equation (2.1a) into time-dependent amplitude ($p$) and space- and time- dependent shape ($\varphi$). The resulting equation is the shape-diffusion equation with precursors represented by Equations (2.2a) and (2.2b).

$$\begin{aligned}\frac{1}{v^g}\frac{\partial \varphi^g}{\partial t} =& \frac{\chi_p^g}{k_{eff}}(1-\beta)\sum_{g'=1}^{G} \nu^{g'}\Sigma_f^{g'}\varphi^{g'} + \sum_{g'\neq g}^{G} \Sigma_s^{g'\to g}\varphi^{g'} \\ &- \left(-\nabla\cdot D^g\nabla + \Sigma_r^g + \boxed{\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}}\right)\varphi^g + \boxed{\frac{1}{p}}\sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i, \quad 1 \le g \le G\end{aligned} \tag{2.2a}$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{k_{eff}}\boxed{p}\sum_{g=1}^{G}\nu^g\Sigma_f^g\varphi^g - \lambda_i C_i\ , \quad 1 \le i \le I \tag{2.2b}$$

We note that the time-dependent shape equation is similar to the time-dependent flux equation, with the following modifications:

1. The shape equation contains an additional term equivalent to a removal cross section,

$\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}$.

2. The delayed neutron source term is divided by $p$.

3. The system of equations is now nonlinear due to the factorization.

4. An equation is needed to obtain the amplitude $p$.

To derive the amplitude equation, the shape/precursors equations are weighted by a space-dependent function and integrated over the phase-space. The weight function is typically the adjoint flux $\phi^{*g}$ at the initial time, which can be proven to minimize truncation error [10]. The final expressions are given below:

$$\frac{dp}{dt} = \left[\frac{\rho - \bar{\beta}}{\Lambda}\right] p + \sum_{i=1}^{I} \bar{\lambda}_i \xi_i \tag{2.3a}$$

$$\frac{d\xi_i}{dt} = \frac{\bar{\beta}_i}{\Lambda} - \bar{\lambda}_i \xi_i \quad 1 \le i \le I \tag{2.3b}$$

This equation is also known as the point reactor kinetics equation (PRKE); where the reactivity, effective delayed-neutron fraction, and delayed-neutron precursor decay constant are defined as follows:

$$\frac{\rho - \bar{\beta}}{\Lambda} = \frac{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{\chi_p^g}{k_{\mathit{eff}}} \sum_{g'=1}^{G} \nu_p^{g'} \Sigma_f^{g'} \varphi^{g'} + \sum_{g' \neq g}^{G} \Sigma_s^{g' \to g} \varphi^{g'} + \left(\nabla \cdot D^g \nabla - \Sigma_r^g\right) \varphi^g\right)}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g\right)} \tag{2.4a}$$

$$\frac{\bar{\beta}}{\Lambda} = \sum_{i=1}^{I} \frac{\bar{\beta}_i}{\Lambda} = \sum_{i=1}^{I} \frac{1}{k_{\mathit{eff}}} \frac{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g \beta_i \sum_{g'=1}^{G} \nu^{g'} \Sigma_f^{g'} \varphi^{g'})}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g\right)} \tag{2.4b}$$

$$\bar{\lambda}_i = \frac{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g \lambda_i C_i)}{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g C_i)} \tag{2.4c}$$

The following inner product definition has been used: $(\phi^{*g}, f) := \int_D \phi^{*g}(\vec{r}) f(\vec{r}) d^3r$.

Additionally, in order to impose uniqueness on the factorization and to derive the

PRKE, the following normalization condition is imposed: $\sum_{g=1}^{G}\left(\phi^{*g}, \frac{1}{v^g}\varphi^g\right) = constant$ [15].

Solving for the shape in Equation (2.2a) can become expensive, especially in two or three dimensions, and even more so when using the transport equations in lieu of the diffusion equations. Using IQS, one expects the time dependence of the shape to be weaker than that of the flux itself, thus allowing for larger time step sizes in updating the shape. The PRKE equations form a small ODE system and can be solved using a much smaller time step size. In transients where the shape varies much less than the flux, IQS can thus be very computationally effective. The two-time scale solution process, a micro scale for the PRKE and a macro scale for the shape, is illustrated in Figure 2.1.

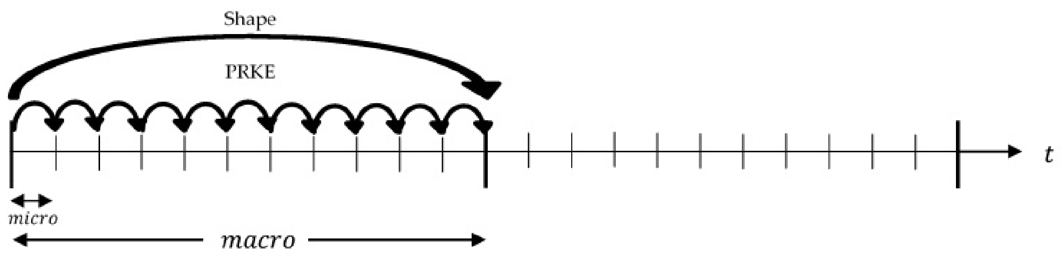


Figure 2.1: IQS method visualization [1]

### 2.1.1 Operator Notation and Extension of IQS to Multigroup Transport Equations

For simplicity, the previous section only described the IQS implementation using the neutron diffusion equation. However, deriving the IQS form for other neutron balance equations (e.g., transport, simplified transport, etc.) is very similar. To this end, we re-write the neutron conservation equations in operator form, shown in Equation (2.5).

$$\frac{1}{v^g}\frac{\partial \Psi^g}{\partial t} = \sum_{g'}\left(H^{g'\to g} + P_p^{g'\to g}\right)\Psi^{g'} - L^g\Psi^g + S_d^g \tag{2.5}$$

where $\Psi^g$ is the multigroup neutron flux (angular flux in the case of transport), $H^{g'\to g}$ is the scattering operator, $P_p^{g'\to g}$ is the prompt neutron production operator, $L^g$ is the loss operator, and $S_d^g$ is the delayed neutron source operator. Using Equation (2.1a), the reader may easily obtain the functional form for these operators in the case of a diffusion approximation. Next, the factorization $\Psi^g(\vec{r},\vec{\Omega},t) = p(t)\psi^g(\vec{r},\vec{\Omega},t)$ is introduced, where the shape is denoted by $\psi^g$, leading to the following shape equations, Equation (2.6):

$$\frac{1}{v^g}\frac{\partial \psi^g}{\partial t} = \sum_{g'}\left(H^{g'\to g} + P_p^{g'\to g}\right)\psi^{g'} - \left(L^g + \frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}\right)\psi^g + \frac{1}{p}S_d^g \tag{2.6}$$

Finally, the PRKE parameters are defined by Equations (2.7) and (2.8), where $(\Psi^{*g}, f^g) = \int_{4\pi}\int_D \Psi^{*g}(\vec{r},\vec{\Omega})f^g(\vec{r},\vec{\Omega})d^3r d^2\Omega$.

$$\frac{\rho-\bar{\beta}}{\Lambda} = \frac{\sum_{g=1}^{G}\left(\Psi^{*g}, \sum_{g'}(H^{g'\to g}g + P_p^{g'\to g} - L^{g'}\delta_{g'g})\psi^{g'}\right)}{\sum_{g=1}^{G}\left(\Psi^{*g}, \frac{1}{v^g}\psi^g\right)} \tag{2.7}$$

$$\frac{\bar{\beta}}{\Lambda} = \sum_{i=1}^{I}\frac{\bar{\beta}_i}{\Lambda} = \sum_{i=1}^{I}\frac{\sum_{g=1}^{G}(\Psi^{*g}, \sum_{g'}P_{d,i}^{g'\to g}\psi^{g'})}{\sum_{g=1}^{G}\left(\Psi^{*g}, \frac{1}{v^g}\psi^g\right)} \tag{2.8}$$

where $P_{d,i}^{g'\to g}$ is the delayed-neutron operator for precursor group $i$.

This section is simply meant to show the theoretical expandability of IQS to transport problems and its derivation in operator notation. As stated previously, this research only applies and tests IQS with diffusion problems.

## 2.2 Iterative Solution Techniques

As we noted in Section 2.1, shape-PRKE equations are a nonlinear system and thus may be solved in a iterative manner. Each macro time step can be iterated so the best shape is used to compute power at the micro time steps. Sissaoui et al. from [18], Koclas et al. from [20], Devooght et al. from [16], and Monier from [17] all use iterative techniques for their quasi-static simulations. They all undergo a similar process:

*Step 1:* Compute the PRKE parameters at the end of the macro step using the last computed shape

*Step 2:* Linearly interpolate the computed PRKE parameters over the macro step

*Step 3:* Solve the PRKE on micro steps over the entire macro step

*Step 4:* Solve the shape equation on the macro step using the computed values of $p$ and $dp/dt$.

*Step 5:* Check if the shape solution has converged:

- *No:* Repeat the same macro time step
- *Yes:* Move on to the next macro time step

This process can be visualized by Figure 2.2.

The major difference between the methods of these authors is the convergence criteria used. Sissaoui and Koclas [18, 20] use fixed point iteration where the criteria is the simply the normalized difference between the last two computed shapes. Monier in [17] also does fixed point iterations with the same criteria, except the solution is scaled by $\frac{\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi^g(t_n)\right)}{\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi^g(t_{n+1})\right)}$ after each iteration. Devooght in [16] does a Newton-SOR iteration where the residual of the shape function evaluation is the convergence criteria and next iteration's solution is computed using Newton-Raphson method.

These techniques are by no means an exhaustive list of the possible iteration techniques for IQS. Dulla et al. in [15] does an in depth analysis of the fixed point iteration technique most similar to Sissaoui and Koclas, involving convergence rates and solution results. However, no comprehensive analysis of iteration techniques exists, comparing

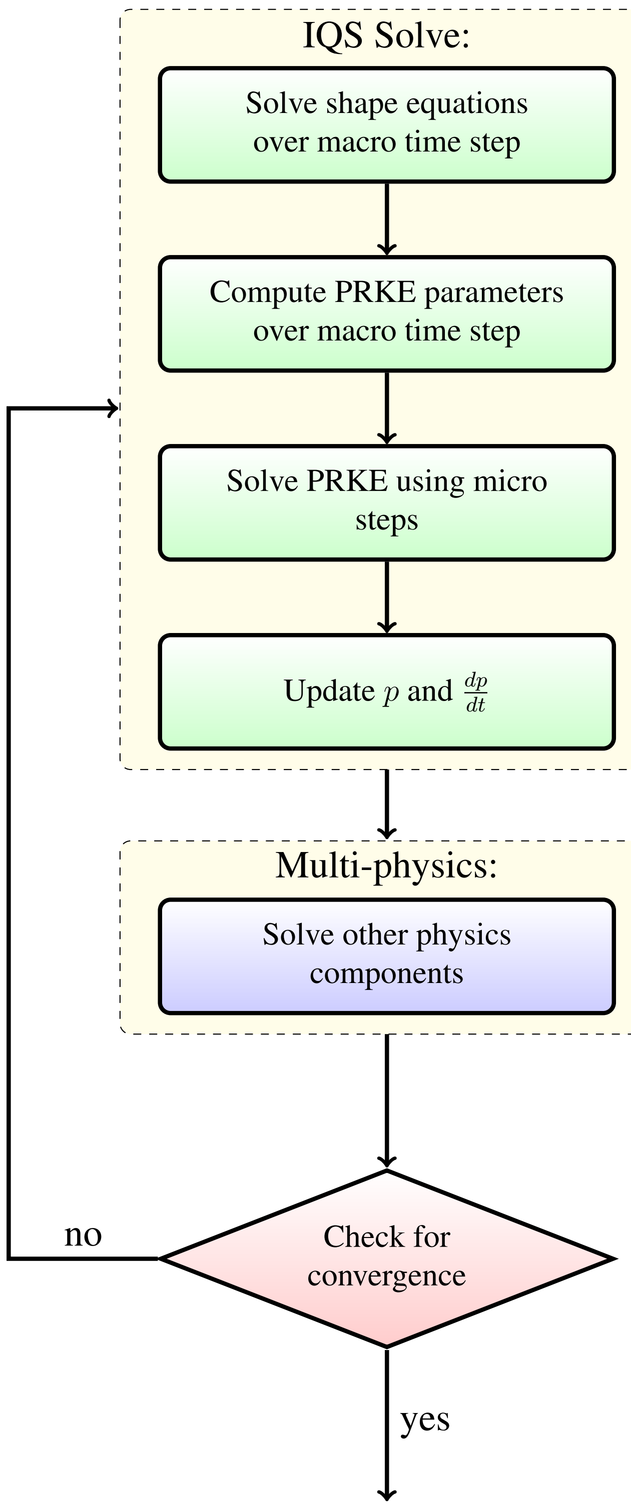

IQS Solve:
Solve shape equations over macro time step
Compute PRKE parameters over macro time step
Solve PRKE using micro steps
Update $p$ and $\frac{dp}{dt}$
Multi-physics:
Solve other physics components
Check for convergence
no
yes

Figure 2.2: Visualization of IQS fixed-point iteration process

both Newton and fixed-point convergence rates. The following sections describes each iteration technique investigated by this research.

### 2.2.1 Shape Convergence

The most obvious convergence criteria is to observe the change in the shape from one iteration to the next. Monier in [17] observes the $L^\infty$ norm of the shape for the convergence criteria, described in Equation (2.9). However, any norm can be used for the criteria, so a $L^2$ norm is another possible criteria, described by Equation (2.10).

$$\frac{\max\left|\varphi_n^{(k+1)} - \varphi_n^{(k)}\right|}{\max\left|\varphi_n^{(k+1)}\right|} < \epsilon_\varphi \tag{2.9}$$

$$\frac{\left\|\varphi_n^{(k+1)} - \varphi_n^{(k)}\right\|_{L^2}}{\left\|\varphi_n^{(k+1)}\right\|_{L^2}} < \epsilon_\varphi \tag{2.10}$$

where $n$ is the time step, $k$ is the iteration number, and $\epsilon_\varphi$ is the numerical criteria provided by a user. Through testing, iterations do not always converge; in this circumstance, the maximum number of iterations is reached.

### 2.2.2 Property Convergence

Monier in [17] describes other properties, other than shape, to observe for convergence, shown in Equations (2.11) - (2.13). These criteria can be added constraints to Equation (2.9) or be in supplement to.

$$\left(\frac{\rho}{\Lambda}\right)^{(k+1)} - \left(\frac{\rho}{\Lambda}\right)^{(k)} < \epsilon_\rho \tag{2.11}$$

$$p_n^{k+1} - p_n^{k} < \epsilon_p \tag{2.12}$$

$$\frac{K_n^{(k+1)} - K_0}{K_0} < \epsilon_K \tag{2.13}$$

where $K$ is the IQS uniqueness expression:

$$K_n^{(k+1)} = \sum_{g=1}^{G} \left( \phi^{*g}, \frac{1}{v^g} \varphi_n^{g,(k+1)} \right) \tag{2.14}$$

$\epsilon_\rho$ is the reactivity convergence criteria, $\epsilon_p$ is the amplitude convergence criteria, and $\epsilon_K$ is the constraint convergence criteria.

### 2.2.3 Solution Scaling

In order to preserve the uniqueness criteria, it is beneficial to scale the shape such that the $K_n$ is constant, shown in Equation (2.15). This scaling can also be done after each iteration, to insure that the uniqueness criteria is satisfied whenever the shape is evaluated.

$$\varphi_n^g = \varphi_n^{g,(\text{last})} \frac{K_0}{K_n^{(\text{last})}} \tag{2.15}$$

### 2.2.4 Preconditioned Jacobian-Free Newton-Krylov

By far, the most common nonlinear system iteration method in MOOSE is Preconditioned Jacobian-Free Newton-Krylov with Generalized Minimal RESidual method (GMRES) as the linear system solver. This section only describes the methods' application to IQS; a very detailed description of PJFNK is transcribed by Knoll in [25]. Essentially, the IQS system of equations can be described by Equation (2.16). Where $A$ is a matrix operator, $\varphi$ is the solution vector, and $F$ is the forcing vector.

$$A(\varphi(p))\varphi = F(\varphi, p, t) \tag{2.16}$$

Applying the residual based Newton-method yields:

$$J\delta\varphi = -R(\varphi, p) \tag{2.17}$$

where $J$ is the Jacobian matrix defined as $J_{ij} = \partial R_i / \partial \varphi_j$, $\delta\varphi$ is the error in the iterating solution, and $R$ is the residual vector defined as $A\varphi - F$ (most methods simply define the residual instead of evaluating this expression). For IQS, a residual evaluation entails a PRKE parameter evaluation and the PRKE evaluation over the entire macro-step. Applying a preconditioner $P$ yields:

$$(JP^{-1})(P\delta\varphi) = -R(\varphi, p) \tag{2.18}$$

This resulting system can be split into two systems, represented by Equations (2.19) and (2.20).

$$(JP^{-1})w = -R(\varphi, p) \tag{2.19}$$

$$\delta\varphi = P^{-1}w \tag{2.20}$$

Solving Equation (2.19) requires the evaluation of Jacobian operator, which is done each GMRES iteration in two steps:

1. Approximately solve the preconditioner system for $y$: $Py = w$
2. Perform matrix-free Jacobian operation: $Jy \approx [R(\varphi + \epsilon y, p') - R(\varphi, p)]/\epsilon$

This Jacobian operation is a finite difference approach. $\epsilon$ is a perturbation scalar and $p'$ is the amplitude computed with PRKE parameters calculated from the perturbed shape. For IQS, the PRKE and its parameters must be evaluated for the perturbed and original system, although the unperturbed residual is stored from the beginning of the PJFNK iteration.

## 2.3 Predictor-Corrector IQS (IQS P-C)

The Predictor-Corrector (P-C) version of IQS factorizes the flux and derives the PRKE the same way as the standard version, but the solution of the coupled system of equations is different. In the IQS P-C version, the flux equations (not the shape equations) are solved (represented by Equations (2.1a) and (2.1b)) in order to obtain a predicted flux solution. This predicted flux is then converted to a shape by rescaling it as follows:

$$\varphi_{n+1}^g = \underbrace{\phi_{n+1}^g}_{\text{predicted}} \frac{K_0}{K_{n+1}} \tag{2.21}$$

where the scaling factors are given by

$$K_{n+1} = \sum_{g=1}^{G} \left( \phi^{*g}, \frac{1}{v^g} \phi_{n+1}^g \right) \tag{2.22}$$

$$K_0 = \sum_{g=1}^{G} \left( \phi^{*g}, \frac{1}{v^g} \varphi_{n+1}^g \right) = \sum_{g=1}^{G} \left( \phi^{*g}, \frac{1}{v^g} \phi_0^g \right) \tag{2.23}$$

The PRKE parameters are then computed with this shape using Equations (2.4a)-(2.4c) and interpolated over the macro step, then the PRKE ODE system is solved on the micro time scale. With the newly computed amplitude, the shape is rescaled into a flux and the final corrected flux is given by:

$$\underbrace{\phi_{n+1}^g}_{\text{corrected}} = p_{n+1} \times \varphi_{n+1}^g \, . \tag{2.24}$$

The advantage to the predictor-corrector method is there is no iteration necessary for this method and, in turn, is much simpler and faster than the standard IQS. Ikeda et al. in [26] and Goluoglu et al. in [27] both use IQS P-C for complex, three-dimensional problems. Their results prove IQS P-C to be impressively effective, despite the de-coupling of the system. Dulla et al. in [15] also describes an in depth comparison of IQS P-C with traditional IQS.

## 2.4 Temperature Feedback

IQS is first and foremost a nuclear reactor simulation method. In nuclear reactors, multiple physics affect the profile of the neutron flux. One of the simplest examples of mulitphysics reactor simulations is adiabatic heat up with Doppler feedback. The principle of Doppler feedback is that fission in a fuel causes the material to increase temperature and induces a change in the neutronics properties. The material heat up is described by Equation (2.25); where $\rho$ is the material density, $c_p$ is the specific heat, $T$ is temperature, and $\kappa_f$ is the energy released per fission [3]. The change in temperature of the material mainly affects the thermal macroscopic absorption cross section described by Equation (2.26) [3].

$$\rho c_p \frac{\partial T(\vec{r},t)}{\partial t} = \kappa_f \sum_{g=1}^{G} \Sigma_f^g \phi^g(\vec{r},t) \tag{2.25}$$

$$\Sigma_a^{thermal}(\vec{r},t) = \Sigma_a^{thermal}(\vec{r},0) \left[1 + \gamma\left(\sqrt{T} - \sqrt{T_0}\right)\right] \tag{2.26}$$

### 2.4.1 Temperature Evaluation

Temperature evaluation (Equation (2.25)) is quite similar to the evaluation of the delayed neutron precursors from Section 2.6. A typical implicit solver would simply use the interpolated flux at end of the temperature time step for the right hand side of the equation, a theta-method discretization. However, IQS has much more information about the profile of the flux along the time step because of the micro-step amplitude evaluation. Therefore, it is possible to solve for temperature using a semi-analytical approach, shown by Equation (2.27).

$$T^{n+1} = T^n + \frac{\kappa_f}{\rho c_p} \sum_{g=1}^{G} \left(a_2(\Sigma_f^g \varphi^g)^{n+1} + a_1(\Sigma_f^g \varphi^g)^n\right) \tag{2.27}$$

where $n$ corresponds to the beginning of the temperature step. $a_1$ and $a_2$ are integration coefficients defined by Equation (2.28) and Equation (2.29). Any interpolation of the amplitude along the micro steps is possible for the integration, this application uses piecewise linear.

$$a_1 = \int_{t_n}^{t_{n+1}} \left( \frac{t_{n+1} - t'}{\Delta t} \right) p(t') dt' \tag{2.28}$$

$$a_2 = \int_{t_n}^{t_{n+1}} \left( \frac{t' - t_n}{\Delta t} \right) p(t') dt' \tag{2.29}$$

### 2.4.2 Intermediate Time Scale

For IQS, this temperature feedback affects both the shape equation and the reactivity of the PRKE; thus, it is an additional nonlinear component to the already coupled shape-amplitude equations. In foresight to the application of this component, temperature is much more time dependent than the shape, but less so than the amplitude. Therefore, the evaluation of temperature will have its own time scale. A possible solution process for a problem with temperature feedback will have time three time scales portrayed in Figure 2.3. The first time scale is the shape solve, the second is the temperature evaluation as well as the computation of PRKE parameters, and the third is the PRKE scale. It is important to note that the number of time steps in each scale is arbitrary and meant only for visual purposes.

### 2.4.3 Dynamical Time Scale Analysis

The temporal variance of flux, shape, temperature, and amplitude can be quantified by defining a dynamical time scale ($\tau$) for each physics. A small value $\tau$ means the variable is quickly varying in time and consequently needs relatively small time steps for accuracy, vice-versa for large $\tau$. The general definition of $\tau$ is defined by Equation (2.30), where $\theta$

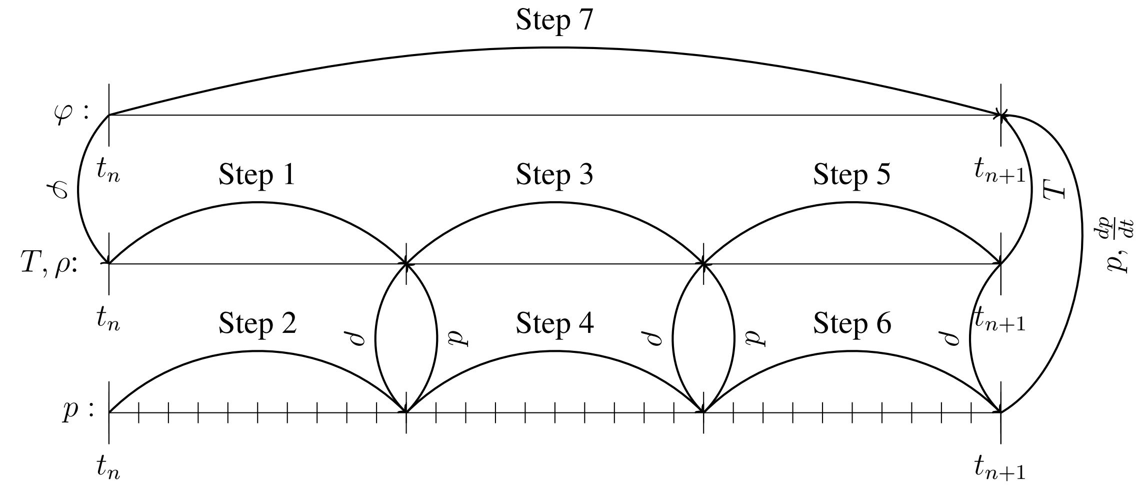


Figure 2.3: Time scales and process of IQS with temperature feedback

is the physic component of interest.

$$\tau = \frac{1}{\left|\frac{1}{\theta}\frac{d\theta}{dt}\right|} \tag{2.30}$$

Since each variable in discretized in time, a finite difference approximation will be made for the $\frac{d\theta}{dt}$ term and the average between the two corresponding time steps will be made for the $\frac{1}{\theta}$ term. Additionally, $\tau$ is spatially dependent for flux and temperature, but only the time dependent behavior of this quantity is of interest. Therefore, the $L^2$ norm of each term will be used to compute the approximate time scale ($\tilde{\tau}$), formally defined by Equation (2.31). $\theta$ represents a summation over groups for flux and shape.

$$\tilde{\tau}_{n+1} = \frac{\|\theta_{n+1} + \theta_n\|_{L^2}}{2} \frac{\Delta t}{\|\theta_{n+1} - \theta_n\|_{L^2}} \tag{2.31}$$

According to the a priori hypothesis from previous sections, $\tau$ is large for shape, somewhat smaller for temperature, and much smaller for amplitude and flux.

### 2.4.4 Programming Logic

The couplings between temperature, amplitude, and shape are all nonlinear, so iteration processes are needed for each time scale. The amplitude and temperature need to be iterated on the middle time scale until convergence on each temperature step. Then another iterative process needs to occur in the shape time scale on all three variables. Figure 2.4 shows the programming diagram implement to execute this process. The time increment of $\Delta t^T$ corresponds to the time step that the temperature is evaluated.

## 2.5 Time Discretization Schemes

A vital part of the verification and validation for IQS is analyzing error convergence. Since any time discretization scheme can be applied to the shape equation of IQS, it is important to investigate IQS's performance to a variety of these schemes. There is lack of literature that applies IQS to schemes other than implicit Euler; higher order schemes are never rigorously tested. This research intends to apply a variety of schemes, including implicit Euler, Crank-Nicholson, backward difference formula (BDF), and diagonally implicit Runge-Kutta (DIRK), to test stability and error convergence. For brevity in the following sections, the shape-diffusion equation is represented by a general operator notation, described by Equation (2.32).

$$IV\frac{\partial \varphi}{\partial t} = A\varphi + b \tag{2.32}$$

where $IV$ is the inverse velocity operator, $A$ contains all the operations on $\varphi$ from the left-hand-side of Equation (2.2a), and $b$ is the source from the precursors.

### 2.5.1 Theta Method Time Discretization

A fairly simple way to evaluate the shape equation is to employ the $\theta$-scheme ($0 \leq \theta \leq 1$, explicit when $\theta = 0$, implicit when $\theta = 1$, and Crank-Nicholson when $\theta = 1/2$)

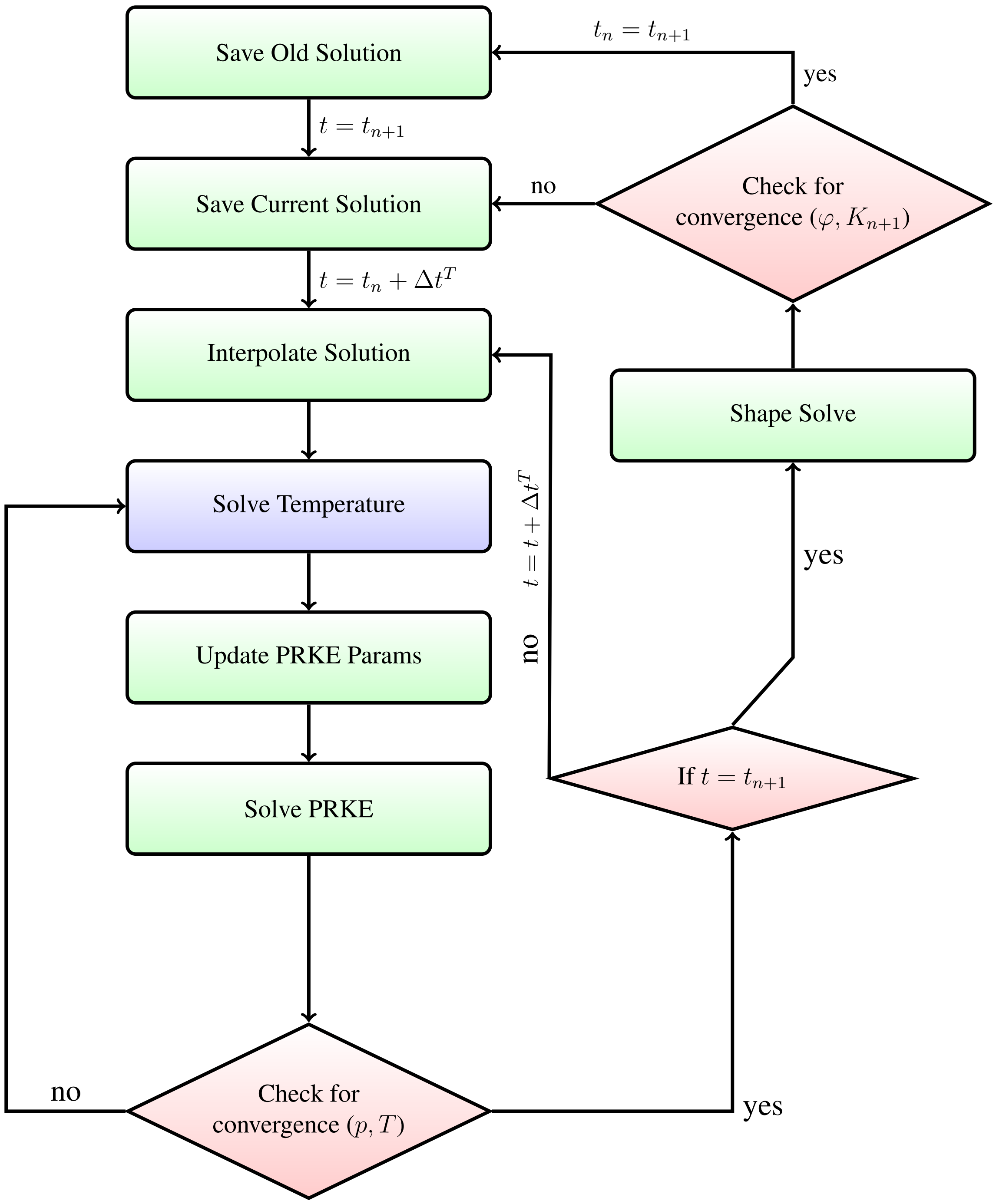


Figure 2.4: Visualization of fixed-point iteration and temperature update process for IQS

[28]. Generally, if there is a function $u$ whose governing equation is $\frac{du}{dt} = f(u,t)$, then the $\theta$-discretization is:

$$\frac{u^{n+1} - u^n}{\Delta t} = (1-\theta) f(u^n, t) + \theta f(u^{n+1}, t)\,. \tag{2.33}$$

where $n$ is the previous time step and $n+1$ is the time step being evaluated. Applying this to Equation (2.32) yields:

$$\varphi_{n+1} = (IV - \Delta t A_{n+1})^{-1} \left[\Delta t (1-\theta)(A_n \varphi_n + b_n) + \Delta t \theta b_{n+1} + IV \varphi_n\right] \tag{2.34}$$

### 2.5.2 Backward Difference Formulae

An extension of implicit discretization is the backward difference formulae (BDF) [29]. BDF's can increase the order of error convergence by interpolating solutions from previous time steps. The general formula for BDF is described by Equation (2.35).

$$\varphi_{n+1} = (IV - \Delta t A_{n+1})^{-1} \left[ IV \sum_{j=1}^{k} \alpha_j^k \varphi_{n-(k-j)} + \Delta t \alpha_{k+1}^k b_{n+1} \right] \tag{2.35}$$

where $k$ is the order of the error convergence and $\alpha_j^k$ (vector of length $k+1$) are coefficients chosen such that the temporal truncation error is minimized. Table 2.2 shows the values of $\alpha$ for every order implemented in this thesis. The benefit of using BDF is that any order of convergence can be applied and are practically stable up through fourth order, so the IQS approximation can easily be validated for high order discretization.

### 2.5.3 Singly-Diagonally-Implicit Runge-Kutta Method

Singly-Diagonally-Implicit Runge-Kutta Method (SDIRK) is a powerful discretization method that involves solving the linear system in stages to reach a high order solution [30]. Generally, the method can be depicted by using a system where $dy/dt = f(t,y)$ and

| Order ($k$) | $\alpha_1^k$ | $\alpha_2^k$ | $\alpha_3^k$ | $\alpha_4^k$ | $\alpha_5^k$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | | | |
| 2 | -1/3 | 4/3 | 2/3 | | |
| 3 | 2/11 | -9/11 | 18/11 | 6/11 | |
| 4 | -3/25 | 16/25 | -36/25 | 48/25 | 12/25 |

Table 2.2: $\alpha$ values for relevant BDF methods

Equation (2.36).

$$y_{n+1} = y_n + \Delta t \sum_{i=1}^{s} b_i k_i \tag{2.36}$$

where,

$$k_i = f(t_n + c_i \Delta t, y_n + \Delta t \sum_{j=1}^{i} a_{ij} k_j) \tag{2.37}$$

The coefficients $a_{ij}$, $b_i$, and $c_i$ can be represented by a Butcher tableau:

$$\begin{array}{c|cccc} c_1 & a_{11} & & & \\ ,c_2 & a_{21} & a_{22} & & \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

An example used extensively in this research is SDIRK33, which is a third-order method with three stages.

$$\begin{array}{c|ccc} \lambda & \lambda & & \\ \frac{1}{2}(1+\lambda) & \frac{1}{2}(1-\lambda) & \lambda & \\ 1 & \frac{1}{4}(-1+16\lambda-6\lambda^2) & \frac{1}{4}(5-20\lambda+6\lambda^2) & \lambda \\ \hline & \frac{1}{4}(-1+16\lambda-6\lambda^2) & \frac{1}{4}(5-20\lambda+6\lambda^2) & \lambda \end{array}$$

where $\lambda \approx 0.4358665215$ satisfies $1 - 9\lambda + 18\lambda^2 - 6\lambda^3 = 0$.

### 2.5.4 Time Adaptation

IQS aims at reducing the time discretization error in the flux solution by splitting the flux into an amplitude (highly resolved at a micro time scale) and a shape (whose time-dependence is weaker than that of the flux itself). Thus, by construction, the IQS approach may employ larger time-step sizes for comparable temporal error. Further enhancements can be gained by using time adaptation (or time step control) in order to increase or reduce the macro time step size for the shape evaluation, depending on error estimates. A step-doubling technique is chosen as the time adaptation technique [31]. The step doubling technique involves estimating the local error for a certain time step by taking the difference between a solution with one full step ($\varphi^g_{\Delta t}$) and a solution with two half steps ($\varphi^g_{\Delta t/2}$). Note: $\varphi$ is changed to $\phi$ for implicit discretization and IQS P-C.

The relative error is computed as follows:

$$e_n = \frac{\left\| \sum_{g=1}^{G} \varphi^g_{\Delta t/2} - \sum_{g=1}^{G} \varphi^g_{\Delta t} \right\|_{L^2}}{\max\left( \left\| \sum_{g=1}^{G} \varphi^g_{\Delta t/2} \right\|_{L^2}, \left\| \sum_{g=1}^{G} \varphi^g_{\Delta t} \right\|_{L^2} \right)} \tag{2.38}$$

If the error is smaller than the user-specified tolerance, $e_{tol}$, the time step is accepted. In addition, a new time step size is estimated as follows:

$$\Delta t_{new} = S \Delta t \left( \frac{e_{tol}}{e_n} \right)^{\frac{1}{1+q}} \tag{2.39}$$

where $q$ is the convergence order of the time integration scheme being used and $S \simeq 0.8$ is a safety factor. If the error is larger than the user-specified tolerance, the time step is rejected. A new time step size is estimated using Equation (2.38) as well. This process can be visualized by Figures 2.5 and 2.6. Where a step involves a full convergence of shape, amplitude, and any multiphysics on the respective time step.

To investigate IQS's performance with step-doubling time adaptation, the adaptation will be applied to implicit discretization method, traditional IQS, and IQS P-C. Each of these methods will be applied to several diffusion problems; the number of time steps taken and the resulting error will be used to compare the methods.

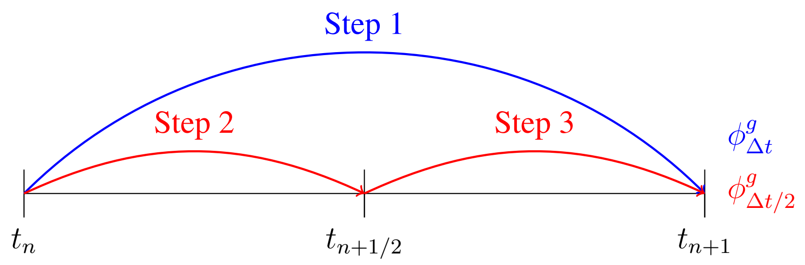


Figure 2.5: Visualization of step doubling process on time-line

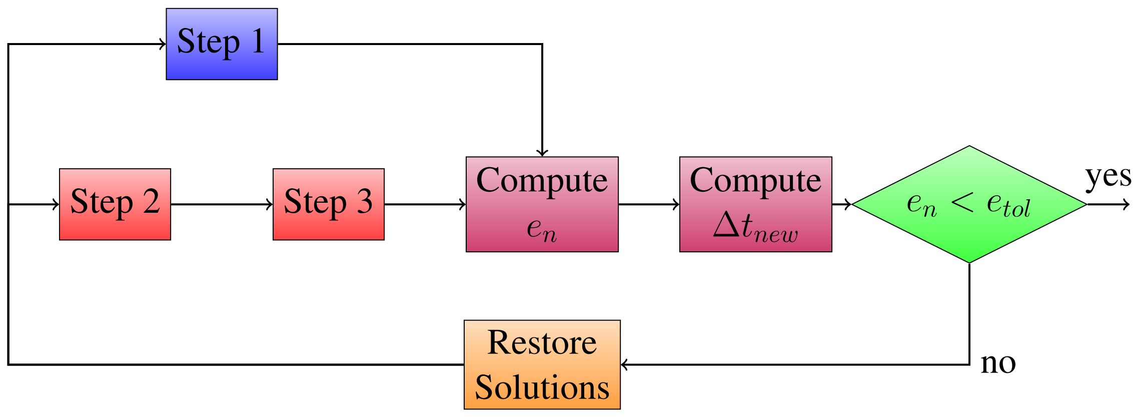


Figure 2.6: Visualization of step doubling process with coding logic

## 2.6 Delayed Neutron Precursor Updates

This section presents the time-integration method used to solve coupled flux/shape and precursor equations, represented by Equations (2.1a)/(2.2a) and (2.1b)/(2.2b). First, we note that we could keep this system of two time-dependent equations and solve it as a coupled system. However, this is unnecessary and a memory expensive endeavor because the precursor equation is only an ODE and not a PDE. Instead, one may discretize in time the shape equation, which typically requires the knowledge of the precursor concentrations at the end of the time step. This precursor value is taken from the solution, numerical or analytical, of the precursors ODE.

### 2.6.1 Precursor Evaluation Using Theta Method Time Discretization

A common precursor evaluation technique is using the theta method, described in Section 2.5.1. Applying this to Equation (2.2b):

$$\frac{C_i^{n+1} - C_i^n}{\Delta t} = (1-\theta)\beta_i S_f^n p^n - (1-\theta)\lambda_i C_i^n + \theta\beta_i S_f^{n+1} p^{n+1} - \theta\lambda_i C_i^{n+1} \tag{2.40}$$

where $S_f$ is the fission source equivalent for shape ($S_f^n = \sum_{g=1}^{G}(\nu\Sigma_f^g \varphi^g)^n$). Rearranging to solve for the precursor at the end of the time step yields

$$C_i^{n+1} = \frac{1-(1-\theta)\Delta t\lambda_i}{1+\theta\Delta t\lambda_i} C_i^n + \frac{(1-\theta)\Delta t\beta_i}{1+\theta\Delta t\lambda_i} S_f^n p^n + \frac{\theta\Delta t\beta_i}{1+\theta\Delta t\lambda_i} S_f^{n+1} p^{n+1} \tag{2.41}$$

Reporting this value of $C_i^{n+1}$, one can solve for the shape $\varphi^{n+1,g}$ as a function of $\varphi^{n,g}$ and $C_i^n$ (and $p^n$, $p^{n+1}$, $dp/dt|_n$ and $dp/dt|_{n+1}$). Once $\varphi^{n+1,g}$ has been determined, $C_i^{n+1}$ is updated. Applying this technique to implicit discretization is done by changing the definition the fission source ($S_f^n = \sum_{g=1}^{G}(\nu\Sigma_f^g \phi^g)^n$) and eliminating all $p$ terms.

### 2.6.2 Analytical Precursor Integration

Another technique to evaluating the precursor equation is to use a exponential operator and integrate the time derivative analytically. Applying this operation to Equation

(2.2b) yields:

$$C_i^{n+1} = C_i^n e^{-\lambda_i(t_{n+1}-t_n)} + \int_{t_n}^{t_{n+1}} \beta_i(t')S_f(t')p(t')e^{-\lambda_i(t_{n+1}-t')}dt' \tag{2.42}$$

Again, this can be applied to implicit discretization by altering the definition of the fission source and eliminating $p$. Because $S_f$ is not known continuously over the time step, the integration can be done using any schemet (Riemann, trapezoid, Simpson's, etc.). However, there is a very accurate representation of $p(t)$ over the macro step from the PRKE solve. In order to utilize this information, another possibility is to interpolate $S_f$ linearly over the macro step. Such that:

$$S_f(t) = \frac{t_{n+1}-t}{t_{n+1}-t_n}S_f^n + \frac{t-t_n}{t_{n+1}-t_n}S_f^{n+1} \quad t_n \le t \le t_{n+1} \tag{2.43}$$

Applying this to Equation (2.42) yields:

$$C_i^{n+1} = C_i^n e^{-\lambda_i \Delta t} + \left(\hat{a}_{2,i}S_f^{n+1} + \hat{a}_{1,i}S_f^n\right)\beta_i \tag{2.44}$$

With integration coefficients defined as:

$$\hat{a}_{1,i} = \int_{t_n}^{t_{n+1}} \frac{t_{n+1}-t'}{\Delta t}p(t')e^{-\lambda_i(t_{n+1}-t')}dt' \tag{2.45}$$

$$\hat{a}_{2,i} = \int_{t_n}^{t_{n+1}} \frac{t'-t_n}{\Delta t}p(t')e^{-\lambda_i(t_{n+1}-t')}dt' \tag{2.46}$$

The amplitude $(p)$ is included in the integration coefficient because it has been highly accurately calculated in the micro step scheme, so a piecewise interpolation (linear, cubic, etc.) between those points can be done to maximize accuracy.

## 2.7 Rattlesnake Implementation

Rattlesnake is a MOOSE-based application developed INL specific to solving radiation transport problems with multiphysics capabilities. MOOSE is a finite-element based, multiphysics framework that gives the general architecture for the development of physics application like Rattlesnake. At the heart of Rattlesnake is the action system, which provides a means to consolidate the MOOSE input syntax (which can be quite larger for multigroup transport simulations), so that a user does not have to define every kernel, variable, etc., used in the problem. Rather, the user inputs an equation description (e.g., Diffusion, S$_n$, P$_n$, etc.) and a solution method (e.g., SAAF, LS, CFEM, DFEM, etc.), and the action system will incorporate all the necessary physics involved (kernels, boundary conditions, postprocessor, etc.).

### 2.7.1 Executioner

An IQS excecutioner was created to implement the IQS method. The IQS executioner derives from the Transient executioner in MOOSE. The IQS executioner contains a loop over micro time steps that computes the PRKE solution and then passes $p$ and $\frac{dp}{dt}$ for the Transient executioner to evaluate the shape equation at each macro step. The PRKE solve is performed with a user specified option of backward-Euler, Crank-Nicholson, or SDIRK33. The IQS executioner also supplements Transient Picard iteration process by adding its own error criteria:

$$Error_{IQS} = \left| \frac{\sum_{g=1}^{G} \left( \phi^{*g}, \frac{1}{v^g} \varphi^{g,n} \right)}{\sum_{g=1}^{G} \left( \phi^{*g}, \frac{1}{v^g} \varphi^{g,0} \right)} - 1 \right| \tag{2.47}$$

### 2.7.2 Action System

The IQS implementation mostly requires the specific IQS executioner, described above. However, IQS additional changes are in the Rattlesnake action system in order to support the IQS execution. First, changes needed to be made in order to evaluate the shape equation. The shape equation, after some manipulation, is very similar to the time-dependent governing laws that Rattlesnake already solves. Using multigroup diffusion as an example again, we show the various kernels employed and highlight the new or modified kernels.

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{\varphi^g}{v^g}\right) = &\underbrace{\frac{\chi_p^g}{k_{\mathit{eff}}}\sum_{g'=1}^{G}(1-\beta)\nu^{g'}\Sigma_f^{g'}\varphi^{g'}}_{FluxKernel} + \underbrace{\sum_{g'\neq g}^{G}\Sigma_s^{g'\to g}\varphi^{g'}}_{FluxKernel} - \underbrace{\left(-\nabla\cdot D^g\nabla\right)\varphi^g}_{FluxKernel} - \underbrace{\Sigma_r^g\varphi^g}_{FluxKernel} \\ &\underbrace{-\frac{1}{v^g}\boxed{\overbrace{\frac{1}{p}\frac{dp}{dt}}^{FromExecutioner}}\varphi^g}_{IQSKernel} + \underbrace{\frac{1}{p}\sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i}_{ModifiedFluxKernel}\end{aligned} \tag{2.48}$$

To enable Rattlesnake to solve this equation, an IQS removal kernel was created to evaluate $\sum_{g=1}^{G}\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}\varphi^g$ and added when the IQS executioner is called. Also, the precursor kernel was modified to include the $\frac{1}{p}$ term. Finally, the precursor auxkernel that evaluates Equation (2.2b) using the analytical integration method described in Section 2.6.

### 2.7.3 PRKE Coefficients

In order to evaluate the PRKE coefficients, defined by Equations (2.4a)-(2.4c), four postprocessors were created. The parameter calculations were separated by $\frac{\bar{\beta}_i}{\Lambda}$ numerator, $\bar{\lambda}_i$ numerator/denominator, $\frac{\rho-\bar{\beta}}{\Lambda}/\frac{\bar{\beta}}{\Lambda}$ denominator, and $\frac{\rho-\bar{\beta}}{\Lambda}$ numerator. The first three are relatively simple, only relying on material properties and solution quantities, then computing the elemental integral. The $\frac{\rho-\bar{\beta}}{\Lambda}$ numerator requires the use of the MOOSE `save_in` feature. This feature saves the residual from a calculated kernel or boundary contribution in the shape evaluation to an auxiliary variable. The postprocessor then computes the inner product of this variable and the initial adjoint solution. After each of these postprocessors are evaluated, a user object pulls together all the values and performs the numerator/denominator divisions. The resulting values are then passed to the executioner for the PRKE solve.

### 2.7.4 Other Action Systems

For simplicity, IQS implementation has only been described for CFEM diffusion. However, Rattlesnake has other action systems capable of transient simulation, where IQS can be implemented and be effective. One of these action systems is DFEM diffusion, where the only major difference from CFEM is the diffusion term in Equations (2.1a) and (2.2a). However, in the derivation for IQS, this term is unaffected between shape and flux evaluation. So saving the residual for this diffusion kernel in the `save_in` variable is the only alteration to this action for IQS to function.

Additional action systems involve transport, but it is evident from Section 2.1.1 that IQS implementation in these is straightforward as well. The main differences between a diffusion implementation and a transport implementation are outlined below:

1. The form of the operators in the shape equations is different, but Rattlesnake has already implemented all the kernels necessary to represent these operators. So no change is necessary to these kernels is necessary for IQS. Additionally, the $\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}$ is the same, so both action systems can use the same kernel.

2. The PRKE parameters also change because of the operators. For the $(\rho - \bar{\beta})$ parameter, the same post-processor can be used with the `save_in` functionality in MOOSE. The post-processor for $\bar{\beta}_i$ must be re-written for transport, but takes a very similar form as diffusion.

It is possible to implement IQS to any action system capable of transient simulation, which include:

- CFEM-Diffusion
- DFEM-Diffusion
- SAAF-CFEM-SN
- SAAF-CFEM-PN
- LS-CFEM-SN
- DFEM-SN

However, for TREAT models and examples simulated in this research, CFEM-Diffusion is a sufficient approximation. Therefore, IQS was only implemented and tested in this action system.

### 2.7.5 Predictor-Corrector Modification

In order to preserve the already implemented standard version of IQS, an option in the IQS executioner was created to specify which method is desired. Because the diffusion solve is flux instead of shape, when predictor-corrector option is specified, the IQS removal kernel ($\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}$) and the modified precursor kernel are bypassed, while all the postprocessors are still executed. However, it is difficult to rescale the flux to shape before the PRKE parameter postprocessor are executed. So the parameters are computed using the full flux, but amplitude is space independent and comes out of the integrals. As seen in parameter definitions, when shape is replaced with flux, the amplitude comes out of the integral and cancels out. So the conversion of the predicted flux in Equation (2.21) to shape is unnecessary if the corrected flux is solved with Equation (2.49). After obtaining the corrected flux, the precursors are re-evaluated using a `EXEC_LINEAR` statement.

$$\underbrace{\phi^g_{n+1}}_{\text{corrected}} = \underbrace{\phi^g_{n+1}}_{\text{predicted}} \frac{K_0}{K_{n+1}} p_{n+1} \tag{2.49}$$

### 2.7.6 Input

The input file for IQS is very similar to the current transient diffusion input file. The IQS input has a different executioner type and parameters. The executioner type is simply IQS and input parameters include the number of micro time steps per macro step, the IQS error tolerance, and the initial power. The Rattlesnake transient action system currently requires a multi-app and transfer to compute and pass the initial $\phi$ and $k_{eff}$, which is present in the transient input deck. However, IQS also requires an initial evaluation of the adjoint flux, for the weighting function. So another input file and multi-app transfer was made for the adjoint calculation.

Below is the syntax for the executioner block for an IQS input file. The `predictor_corrector` logical determines whether to do IQS P-C or regular IQS. The `IQS_error_tol` is the tolerance for the IQS error represented by Equation (2.47) and will at most `picard_max_its` iterations until convergence. The `prke_scheme` defines the time discretization used for the PRKE solution. Where `RK` is SDIRK33, `CN` is Crank-Nicholson, and `IE` is implicit-Euler.

```
[Executioner]
  type = IQS
  predictor_corrector = true/false
  picard_max_its = 5
  ...
  n_micro = 10000
  IQS_error_tol = 1e-7
  prke_scheme = 'RK'
[]
```

Since IQS needs to use the adjoint solution for the PRKE parameter evaluation, auxiliary variables need to be created for each group in the input file. Below is an example of their definition:

```
[AuxVariables]
  [./adjoint_flux_g0]
    family = LAGRANGE
    order = FIRST
  [../]
  [./adjoint_flux_g1]
    family = LAGRANGE
    order = FIRST
  [../]
  ...
[]
```

Below is the syntax for the multi-app block to perform forward and adjoint steady-state evaluations:

```
[MultiApps]
 [./initial_solve]
   type = FullSolveMultiApp
   execute_on = initial
   input_files = initial.i
 [../]
 [./adjoint_solve]
   type = FullSolveMultiApp
   execute_on = initial
   input_files = adjoint.i
 [../]
[]
```

Below is the syntax for the Transfer block to copy the initial and adjoint solutions from the multi-apps.

```
[Transfers]
 [./copy_solution]
   type = TransportSystemVariableTransfer
   direction = from_multiapp
   multi_app = initial_solve
   execute_on = initial
   from_transport_system = diff
   to_transport_system = diff
 [../]
 [./copy_adjoint]
   type = MultiAppVariableTransfer
   execute_on = initial
   direction = from_multiapp
   multi_app = adjoint_solve
   from_variables = 'sflux_g0 sflux_g1 ...'
   to_variables = 'adjoint_flux_g0 adjoint_flux_g1 ...'
 [../]
 [./copy_eigenvalue]
   type = EigenvalueTransfer
   execute_on = initial
   direction = from_multiapp
   multi_app = initial_solve
 [../]
[]
```

### 2.7.7 Linear Solution Techniques

MOOSE contains a suite of solution techniques to solve the linear system generated by the Rattlesnake action system. For most neutronics and reactor applications, direct inversion (e.g., LU solve) of the system is overbearingly expensive, both computationally and by memory usage. Therefore, the most common technique (default in MOOSE) is GMRES linear iteration. For nonlinear systems, MOOSE employs Picard or PJFNK nonlinear iterations, PJFNK is the default technique in MOOSE and will be the basis of the following discussion. This technique is briefly described in Section 2.2.4. PJFNK entails approximately solving the Jacobian action on the solution with a finite difference approach using GMRES iteration. Each GMRES iteration requires a residual evaluation, which becomes globally expensive with many iterations. Employing a preconditioner on the solution can decrease the number of iterations and save execution time. The purpose of the preconditioner is to resemble the Jacobian (reduce number of iterations), while being easily invertable (preconditioner overhead). MOOSE has four useful assembly methods for building an effective preconditioner:

1. Block Diagonal Preconditioning (default)
2. Single Matrix Preconditioner (SMP)
3. Finite Difference Preconditioner (FDP)
4. Physics Based Preconditioner (PBP)

When developing a kernel in MOOSE, a function is implement that evaluates the Jacobian for each variable it depends on. In a multi-variable system, like multigroup diffusion, the Jacobian contains "blocks" for each variable and the coupling between them. Diagonal blocks contain the variable contribution to itself, i.e., diffusion and removal. The off-diagonal blocks contain variable-to-variable contributions, i.e. scattering and fission. The default method assembles only the diagonal blocks; this preconditioner is relatively easy to invert, but is not optimal for reducing linear iterations. SMP assembles the diagonal blocks and any off-diagonal blocks specified by the user. Assembling more off-diagonal blocks reduces linear iterations (inputting all blocks requires only one GMRES iteration for a linear system), but increases the expense of inverting the preconditioner. Therefore, SMP requires extensive analysis for optimization. FDP simply does a finite difference approximation for the Jacobian based on residual evaluations. This method only requires one GMRES iteration for a linear system, but is impractically expensive. PBP is an advanced adaptation to SMP. This method assembles the preconditioner similarly to SMP, but each block is inverted as a independent system and iteratively communicates with the rest of the blocks. PBP can be effective for a lower block diagonal system, i.e., including only downscattering blocks, because no communication iterations are required. Selecting an effective preconditioner and assembly method is essential for maximizing computational efficiency of large reactor simulations.

# 3. KINETICS EXAMPLES

Neutron kinetics is the study of the time-dependent nature of neutrons in a reactor. More specifically, there is no coupling of neutron behavior with other physics, like thermal hydraulic feedback. This section describes kinetics examples that IQS is tested with and analysis of its performance. The examples range in complexity and application. The first is a one-dimensional problem, designed for the prototype code in MATLAB. The next one is from the Argonne National Lab (ANL) Benchmark Problem Book (BPB), and is a common problem for testing codes and developing methods [3].

## 3.1 One Dimensional Prototype Problem

This example is very simple and computes quickly; it entails a one dimensional, homogeneous 400 cm slab with a heterogenous perturbation in absorption cross section. Figure 3.1 shows how the regions of the slab are divided and Table 3.1 shows the initial material properties. Regions 2, 3, and 4 have slope perturbations at different points in time, Table 3.2 shows the values of the absorption cross-section in each region at the times of interest. The values of $\Sigma_a$ between these times of interest are linear interpolations between the given values.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 4 | 1 | 1 | 1 | 1 |

Figure 3.1: 1-D slab region identification [1]

Table 3.1: 1-D slab material properties and problem parameters [1]

| $D(cm)$ | $\Sigma_a(cm^{-1})$ | $\nu\Sigma_f(cm^{-1})$ | $v(cm/s)$ | $\beta$ | $\lambda(s^{-1})$ |
|---|---|---|---|---|---|
| 1.0 | 1.1 | 1.1 | 1,000 | 0.006 | 0.1 |

Table 3.2: 1-D slab absorption cross-section at times of interest [1]

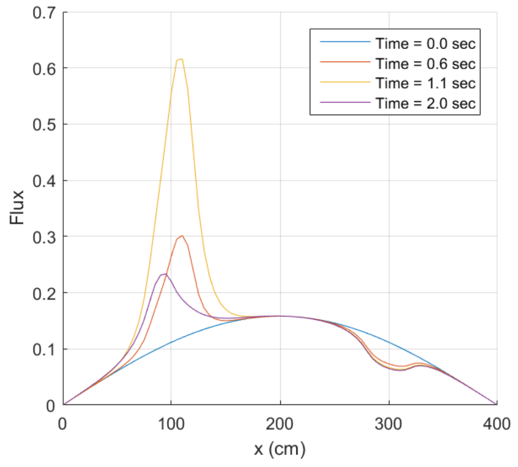
| Region | Material Property | 0.0 s | 0.1 s | 0.6 s | 1.0 s | 1.7 s |
|---|---|---|---|---|---|---|
| 2 | $\Sigma_a(cm^{-1})$ | 1.1 | 1.1 | 1.095 | 1.095 | 1.095 |
| 3 | $\Sigma_a(cm^{-1})$ | 1.1 | 1.1 | 1.09 | 1.09 | 1.1 |
| 4 | $\Sigma_a(cm^{-1})$ | 1.1 | 1.1 | 1.105 | 1.105 | 1.105 |

Figure 3.2 shows the resulting baseline flux and relative power profile of the one-dimensional problem. The baseline was computed using MATLAB's ode15s function which is a embedded Runge-Kutta time adaptive method for stiff problems, the error tolerance was set very tightly ($10^{-12}$). The flux distribution shows that the core is relatively large, thus different regions are weakly coupled. This baseline computation is used to compute the error of the other time discretization methods. Figure 3.3a shows the shape profile at various times during the transient. It is apparent that the shape is very time-dependent, so it is expected that IQS has marginal accuracy gain for a given time step size.

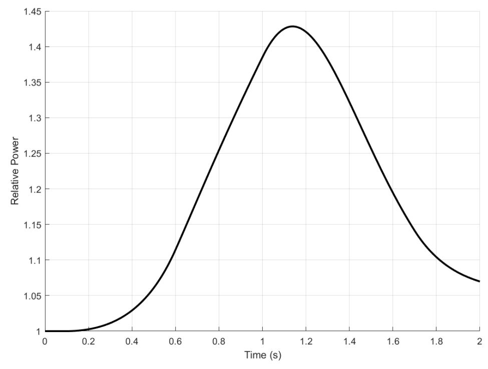
### 3.1.1 IQS Iteration Convergence

IQS, as previously stated, is a system of nonlinear equations between shape and amplitude. These equations needed to be iterated to numerically converge to an accurate solution. Section 2.2 describes various iteration techniques for fixed-point and Newton schemes. Figure 3.4 shows the number of fixed-point iterations required for a $10^{-11}$ tolerance over the transient. The criteria listed in the legend are described as:

1. $L^\infty \rightarrow$ Equation (2.9)

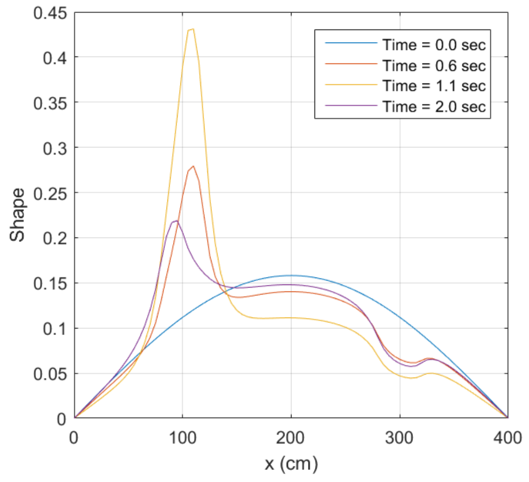


(a) Flux profile at various times

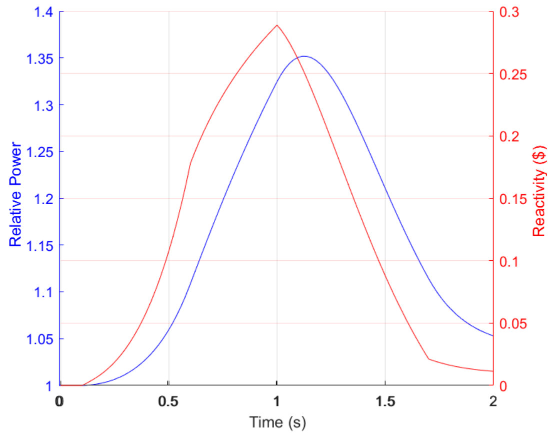


(b) Power profile over transient

Figure 3.2: Baseline flux and power distribution



(a) Shape profile at various times



(b) Power and reactivity profile over transient

Figure 3.3: IQS flux and power distribution

2. $L^2 \rightarrow$ Equation (2.10)

3. Reactivity $\rightarrow$ Equation (2.11)

4. Amplitude $\rightarrow$ Equation (2.12)

5. K criteria $\rightarrow$ Equation (2.13)

6. All properties $\rightarrow$ Equations (2.11)-(2.13)

This plot shows that 1-4 have approximately the same convergence behavior, but the K criteria converges to a certain error. Figure 3.5 shows the resulting error the K criteria converges to for different points of rescaling the shape. The rescaling is described by Equation (2.15). Rescaling shape more frequently helps the error. However, rescaling every iteration is somewhat artificial because it does not consider changes spatially. Regardless, an error of $10^{-5}$ is quite large and it is expected that the magnitude is due to the explicit treatment of precursors (Equation (2.41)). Switching to an analytical elimination (Equation (2.44)) does not converge as well, but the error is much smaller, seen in Figure 3.6.

### 3.1.2 Time Step Convergence

Time step convergence analysis involves evaluating a problem with various refinements in step size and comparing the resulting errors with the time step size. Plotting error versus $\Delta t$ on a log-scale should produce a relatively strait line with a slope equal to the order of the time discretization method. In order to evaluate the performance and error convergence of IQS, the slab was simulated with varying time discretization methods and time step sizes. Figure 3.7 shows these convergence plots of five different discretization methods for implicit discretization, IQS, and IQS P-C. These plots were generated from the results using the MATLAB prototype program. The plots show that IQS and IQS P-C are convergent through fourth order BDF. Third order SDIRK did not show third order

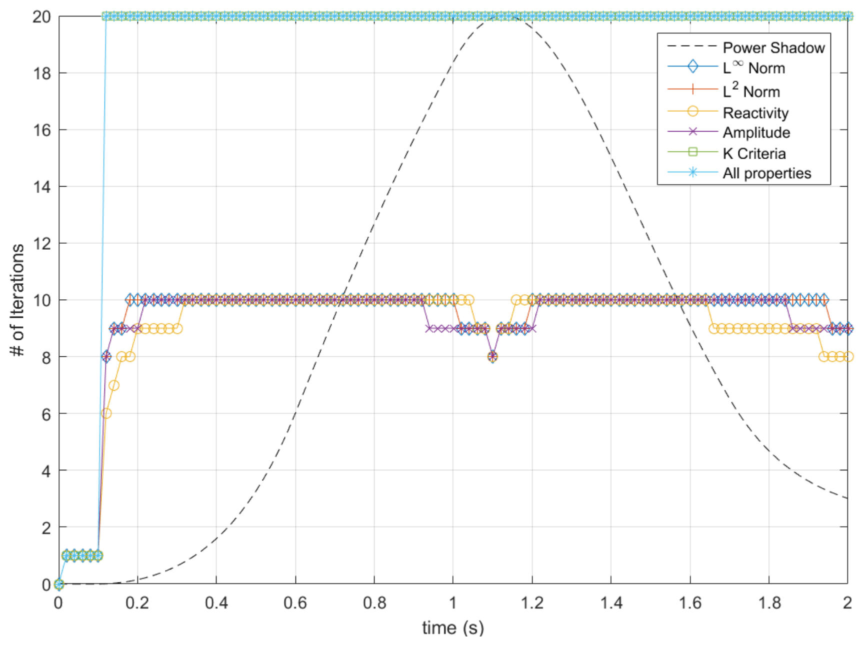


Figure 3.4: Number of iterations for various convergence criteria, tolerance= $10^{-11}$, max iterations= 20

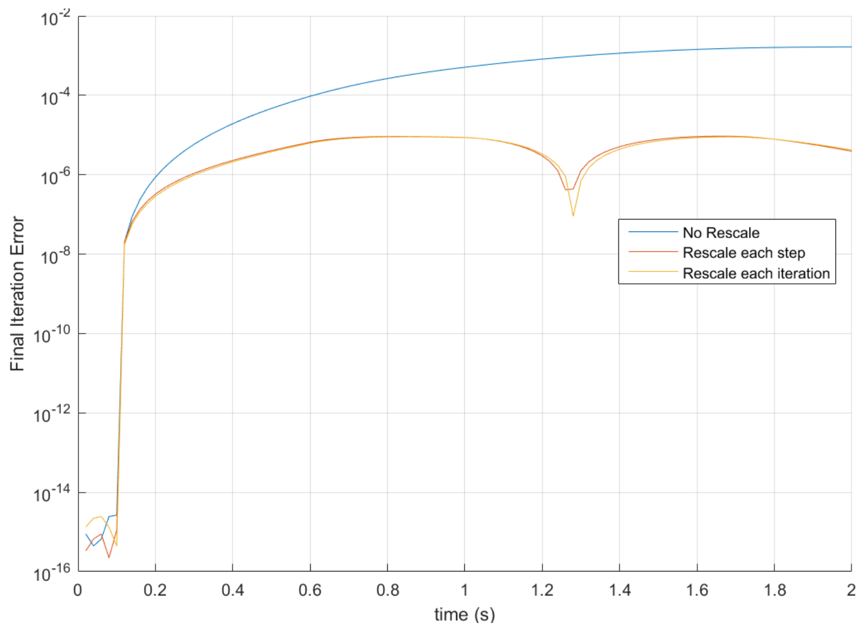


Figure 3.5: Final iteration error for K convergence criteria

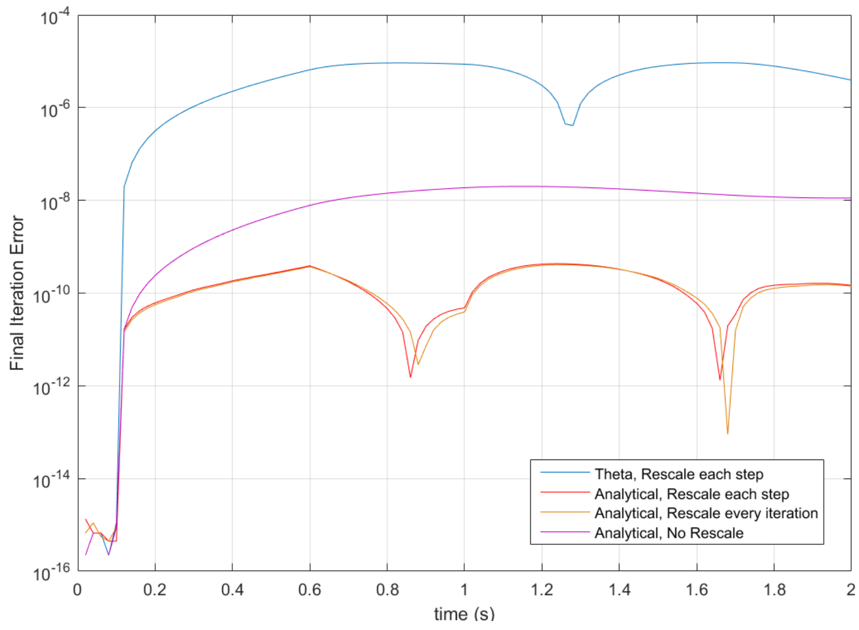


Figure 3.6: Final iteration error for K convergence criteria with analytical precursor elimination

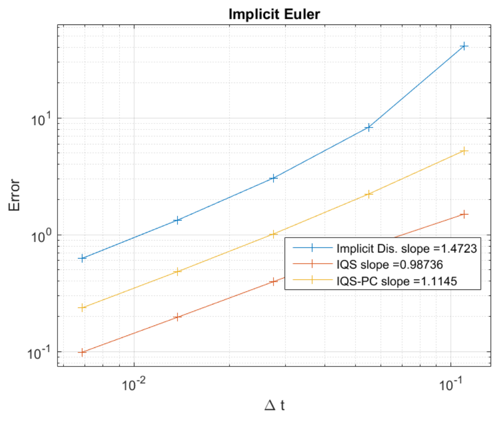
convergence, but, through extensive testing, SDIRK shows non-convergent behavior for too stiff of problems.

There are higher discretization order that can be tested, but most practical application do not go beyond second order. This paper shows an analysis of the first publicized application of IQS with higher than second order discretization, which exposed unforeseen properties of IQS. When using higher order techniques, the interpolation of PRKE parameters and shape for precursor integration become important to consider. Every other application that was investigated linearly interpolates parameters for the PRKE evaluation. Similarly, the shape used for the integration of the ODE for the precursors needs to have higher order interpolation to preserve high order error convergence. This interpolation was done using Lagragian and Hermite methods, both leading to successful convergence.

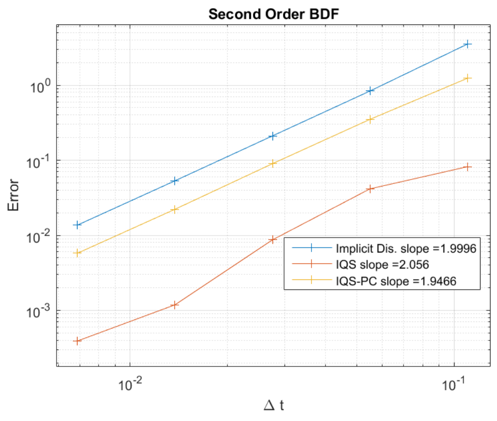
The 1D slab problem was also applied to the Rattlesnake implementation of IQS. Figure 3.8 shows the error convergence for implicit Euler and BDF2 discretization of the three methods. The results differ slightly from the prototype, especially in the fact that IQS P-C performs better than IQS. Since Rattlesnake uses a PJFNK solver for the FEM model, the number of time steps isn't strictly proportional to the execution time. Figure 3.9 shows the error of the three methods as compared to the number of linear GMRES iterations, which is a better mark for comparison in computation time. This figure shows that IQS performs worse than implicit discretization for most time-step sizes, but IQS P-C performs significantly better. This is due to the fact that IQS needs to iterate between amplitude and shape to resolve its nonlinearity.

### 3.1.3 One-Dimensional Mini-Core Problem

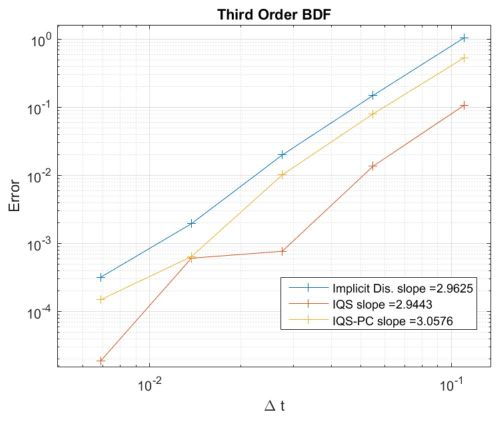
This problem is exactly the same as the previous one-dimensional problem, except the the core was reduced to 80 cm in length. The purpose of testing this problem is to determine if a more tightly coupled core will yield better performance for IQS. Figure 3.10 shows the resulting baseline flux and relative power profile of the one-dimensional mini-core problem. This plot shows that the perturbed regions affect the domain more evidently. Figure 3.11 shows the shape profile at various times during the transient. This plot shows that the shape is much less time-dependent than the previous large core. Figure 3.12 shows that IQS performs significantly better with this example than the large core.



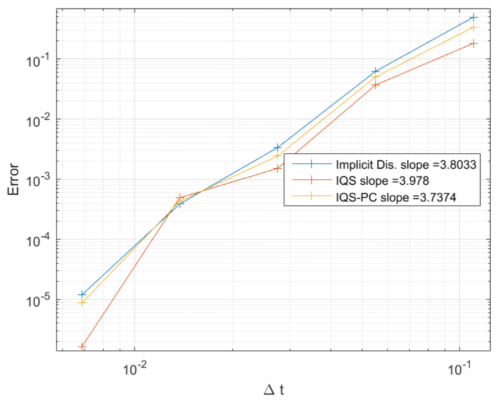
(a) Implicit Euler



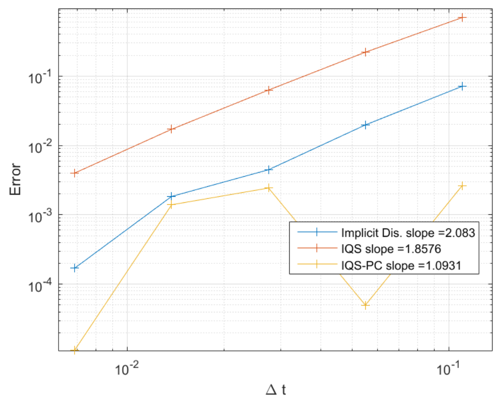
(b) BDF2



(c) BDF3



(d) BDF4



(e) SDIRK33

Figure 3.7: Error convergence plots of implicit discretization, IQS, and IQS P-C with various time discretization schemes

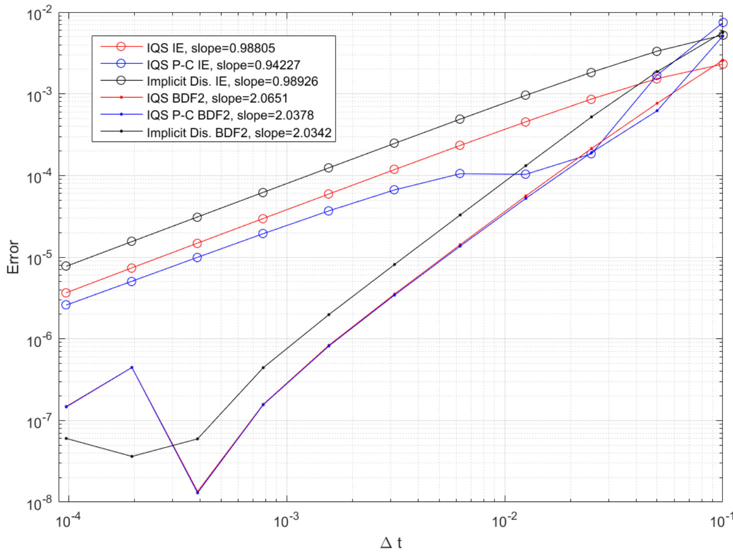


Figure 3.8: Error convergence plots of implicit discretization, IQS, and IQS P-C from Rattlesnake implementation

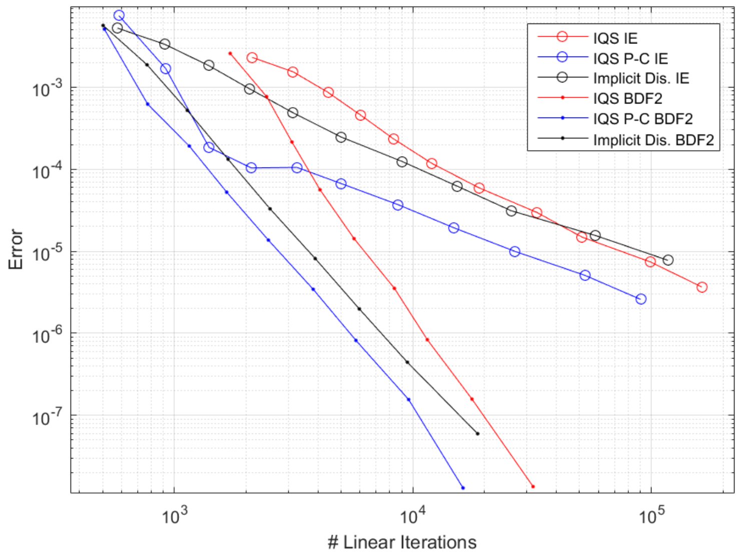
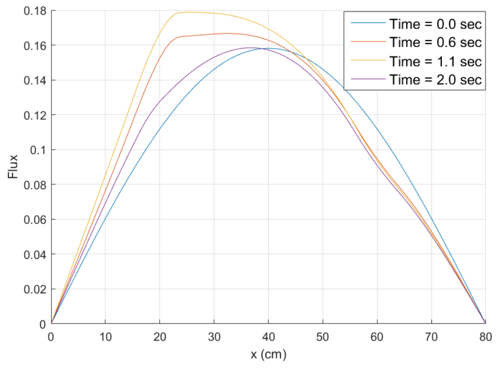
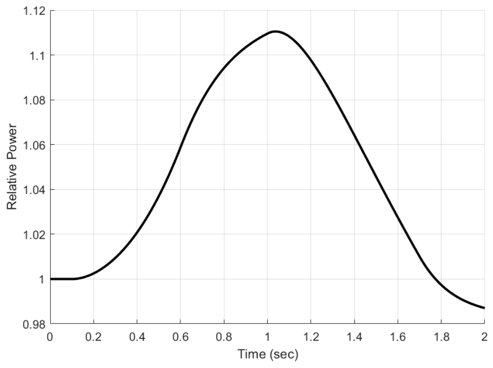


Figure 3.9: Error convergence plots of implicit discretization, IQS, and IQS P-C vs. number of GMRES iterations



(a) Flux profile at various times



(b) Power profile over transient

Figure 3.10: Mini-core baseline flux and power distribution

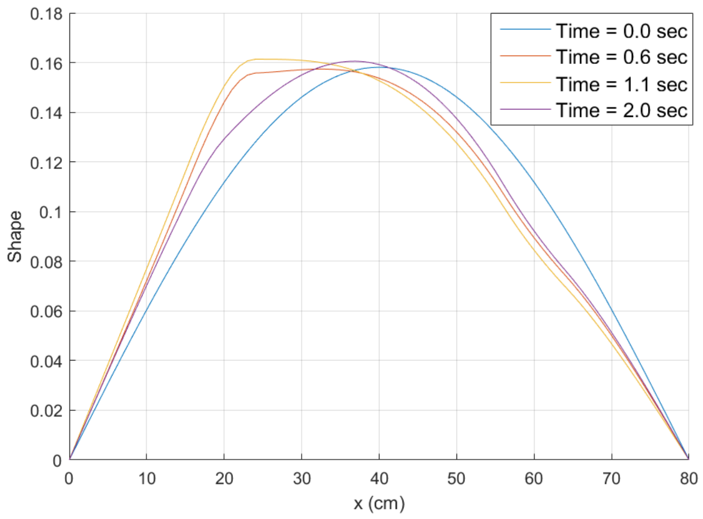


Figure 3.11: Shape profile at various times for one-dimensional mini-core

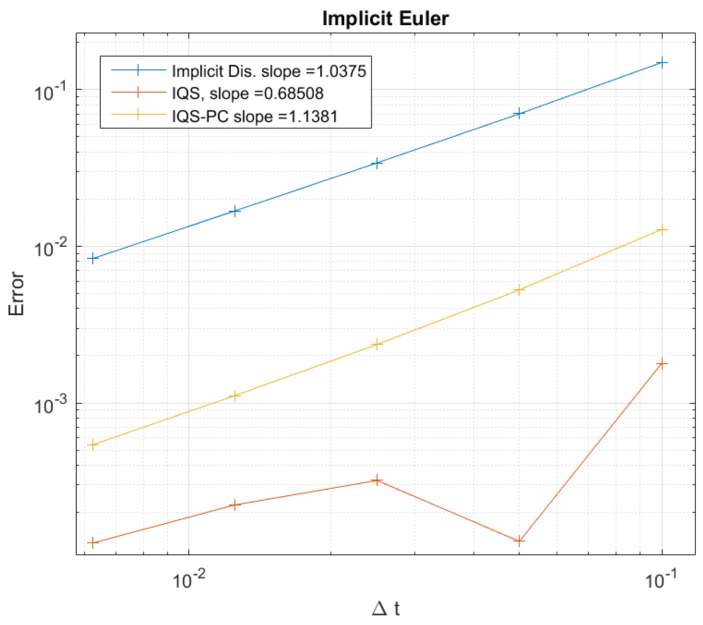


Figure 3.12: Time step convergence for one-dimensional mini-core with implicit Euler discretization

## 3.2 TWIGL Benchmark

This benchmark problem originates from the Argonne National Lab Benchmark Problem Book [3]. It is a 2D, 2-group reactor core model with no reflector region shown in Figure 3.13 [12]. This example is meant to be of progressive complexity from the previous example. The transient of this reactor is very geometrically symmetrical with very little temporal shape change. Therefore, IQS is expected to perform significantly better than the implicit discretization method. Table 3.3 shows the material properties of each fuel region and the ramp perturbation of Material 1.

### 3.2.1 TWIGL Convergence Analysis

Figs. 3.14 and 3.15 show the IQS solution as compared with the implicit discretization solution. It is important to note the IQS shape plot is scaled differently than the

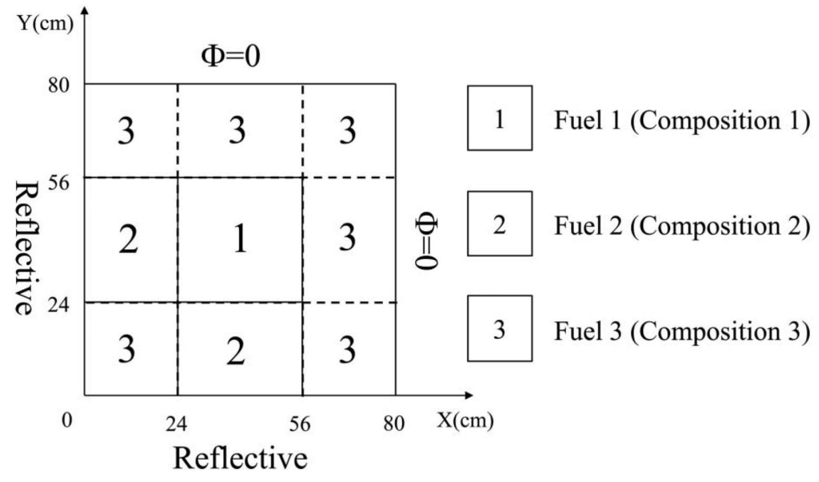


Figure 3.13: TWIGL benchmark problem description [2]

Table 3.3: TWIGL benchmark material properties and slope perturbation [2]

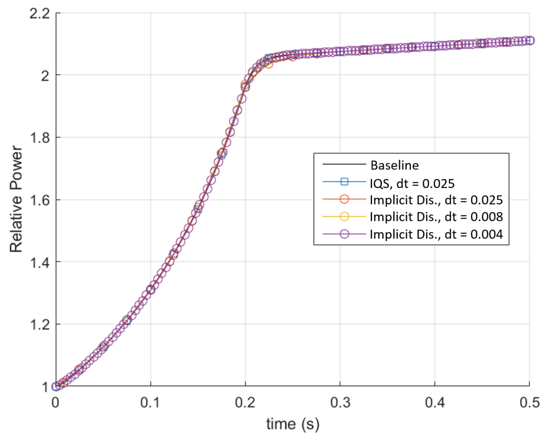
| | | | | | | $\Sigma_s(cm^{-1})$ | |
|---|---|---|---|---|---|---|---|
| Material | Group | $D(cm)$ | $\Sigma_a(cm^{-1})$ | $\nu\Sigma_f(cm^{-1})$ | $\chi$ | $g \to 1$ | $g \to 2$ |
| 1 | 1 | 1.4 | 0.010 | 0.007 | 1.0 | 0.0 | 0.01 |
| | 2 | 0.4 | 0.150 | 0.200 | 0.0 | 0.0 | 0.00 |
| 2 | 1 | 1.4 | 0.010 | 0.007 | 1.0 | 0.0 | 0.01 |
| | 2 | 0.4 | 0.150 | 0.200 | 0.0 | 0.0 | 0.00 |
| 3 | 1 | 1.3 | 0.008 | 0.003 | 1.0 | 0.0 | 0.01 |
| | 2 | 0.5 | 0.050 | 0.060 | 0.0 | 0.0 | 0.00 |
| | $\nu$ | $v_1(cm/s)$ | $v_2(cm/s)$ | $\beta$ | $\lambda(1/s)$ | | |
| | 2.43 | 1.0E7 | 2.0E5 | 0.0075 | 0.08 | | |

Material 1 ramp perturbation:

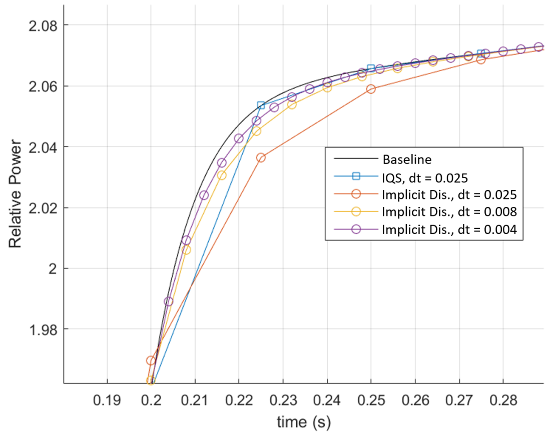
$\Sigma_{a,2}(t) = \Sigma_{a,2}(0) \times (1 - 0.11667t) \quad t \leq 0.2s$

$\Sigma_{a,2}(t) = \Sigma_{a,2}(0) \times (0.97666t) \quad t > 0.2s$

implicit discretization flux plot (Figure 3.15) because the amplitude term is not included, but the gradients of colors is comparable. These plots show that IQS is consistent in more complex, higher dimensional problems in Rattlesnake. These plots also serve to illustrate that IQS has a much more accurate solution, even at a significantly larger time step than the implicit discretization. In order to demonstrate asymptotic convergence of IQS, implicit Euler (IE) and second order BDF (BDF2) were applied to the TWIGL simulation. Figure 3.16 plots the error convergence of IQS and the implicit discretization methods. The curves show the impressive convergence of IQS for the highly transient TWIGL example. The slope indicated in the legend are the linear slope of curves on the log plot, these slopes should be similar to the order of the method (1 for IE and 2 for BDF2). IQS shows a increased order because the PRKE is performing much of accuracy convergence and it is computed using SDIRK33, a third order method.

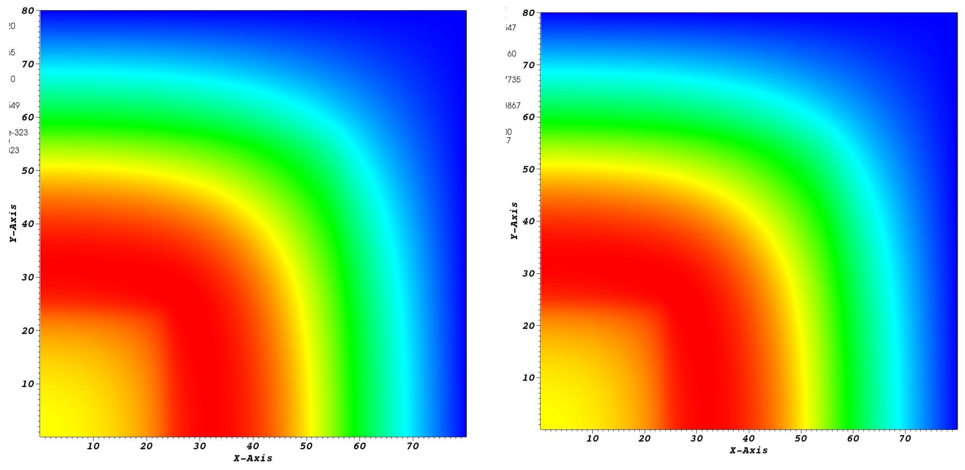


(a) Power profile for entire transient



(b) Power at cusp of profile

Figure 3.14: Power level comparison of TWIGL Benchmark



(a) Implicit Discretization flux

(b) IQS Shape

Figure 3.15: TWIGL Benchmark flux/shape comparison at $t = 0.2$ [1]

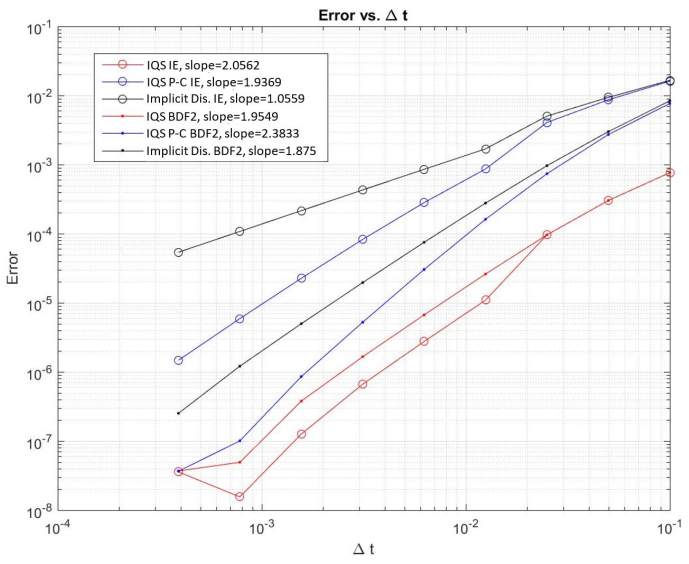


Figure 3.16: Error convergence comparison of TWIGL Benchmark

### 3.2.2 TWIGL with Step Doubling Time Adaptation

Table 3.4 and Figure 3.17 show the results for TWIGL with time adaptation. The results show that both IQS methods perform exceptionally well compared to implicit discretization. It also shows that traditional IQS performed better with large $e_{tol}$, while IQS P-C was better with smaller $e_{tol}$.

Table 3.4: TWIGL step doubling results

| | | Implicit Discretization | | | IQS | | | IQS P-C | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | $e_{tol}$ | Error | Steps | Solves | Error | Steps | Solves | Error | Steps | Solves |
| 1 | 0.05 | 0.00012677 | 9 | 29 | 0.03380433 | 4 | 20 | 0.00323100 | 4 | 9 |
| 2 | 0.01 | 3.5555e-05 | 11 | 35 | 0.00166991 | 5 | 40 | 0.00263068 | 5 | 12 |
| 3 | 0.005 | 4.0364e-05 | 11 | 31 | 0.00886584 | 5 | 40 | 0.00160486 | 6 | 21 |
| 4 | 0.001 | 0.00294822 | 33 | 122 | 0.02976305 | 5 | 36 | 1.7527e-05 | 10 | 35 |
| 5 | 0.0005 | 0.00099778 | 39 | 131 | 0.00143781 | 6 | 55 | 1.4185e-05 | 16 | 74 |
| 6 | 0.0001 | 0.00019510 | 78 | 236 | 0.00016175 | 8 | 65 | 6.2903e-06 | 19 | 78 |
| 7 | 5.0e-05 | 0.00018372 | 112 | 342 | 6.0328e-05 | 12 | 163 | 1.5247e-06 | 24 | 92 |
| 8 | 1.0e-05 | 8.0564e-05 | 263 | 794 | 7.7103e-05 | 379 | 5729 | 9.8321e-07 | 48 | 210 |

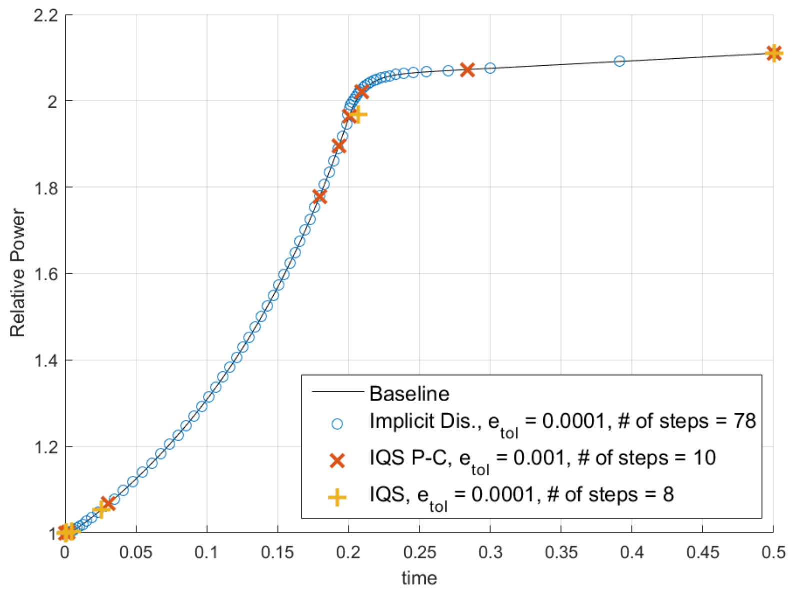


Figure 3.17: Power level comparison of TWIGL Benchmark with time adaptation

# 4. DYNAMICS EXAMPLES

Reactor dynamics is the study of the time-dependent behavior of reactors as an entire system. This study includes the physical nature of neutrons and their feedback with power generation. This feedback includes coupling with other physical properties such as temperature, fluids, material dynamics, etc. This section describes two dynamics examples, including the LRA benchmark and a TREAT experiment. These examples are of increased complexity from the kinetics examples of Chapter 3. This Chapter also analyzes IQS's performance with these, which is vital for verification of IQS in real-world problems.

## 4.1 LRA Benchmark

The LRA benchmark is a two-dimensional, two-group neutron diffusion problem with adiabatic heat-up and Doppler feedback in thermal reactor [3]. It is a super prompt-critical transient. To have better understanding on the cross sections given later, we present the equations here:

$$-\frac{1}{v_1}\frac{\partial \phi_1}{\partial t} = -\nabla\cdot D_1\nabla\phi_1 + (\Sigma_{a,1}+\Sigma_{s,1\to 2})\phi_1 - \nu(1-\beta)S_f - \sum_{i=1}^{2}\lambda_i C_i, \tag{4.1a}$$

$$-\frac{1}{v_2}\frac{\partial \phi_2}{\partial t} = -\nabla\cdot D_2\nabla\phi_2 + \Sigma_{a,2}\phi_2 - \Sigma_{s,1\to 2}\phi_1, \tag{4.1b}$$

$$S_f = \sum_{g=1}^{2}\Sigma_{f,g}\phi_g, \tag{4.1c}$$

$$\frac{\partial C_i}{\partial t} = \nu\beta_i f - \lambda_i C_i, \quad i = 1,2, \tag{4.1d}$$

$$\frac{\partial T}{\partial t} = \alpha f, \tag{4.1e}$$

$$\Sigma_{a,1} = \Sigma_{a,1}(\vec{r}, t=0)\left[1+\gamma\left(\sqrt{T}-\sqrt{T_0}\right)\right], \tag{4.1f}$$

$$P = \kappa S_f, \tag{4.1g}$$

where $\phi_1$, $\phi_2$ are the fast and thermal fluxes; $v_1, v_2$ are the averaged neutron velocities; $\Sigma_{a,1}, \Sigma_{a,2}$ are the absorption cross sections; $\Sigma_{s,1\rightarrow 2}$ is the fast-to-thermal scattering cross section; $\Sigma_{f,1}, \Sigma_{f,2}$ are the fission cross sections; $\nu$ is the averaged number of neutrons emitted per fission; $\beta_1, \beta_2$ are the delayed neutron precursor fractions and $\beta = \beta_1 + \beta_2$; $C_1, C_2$ are the delayed neutron precursor concentrations; $\lambda_1, \lambda_2$ are the decay constants of the delayed neutron precursors; $S_f$ is the fission reaction rate; $P$ is the power density; $T$ is the temperature; $\kappa$ is the averaged power released per fission; $\alpha$ is the combination of $\kappa$ and the specific heat capacity; $\gamma$ is the Doppler feedback coefficient; $T_0(\vec{r}) = T(\vec{r}, t = 0)$. The two-group diffusion equation are solved with zero flux boundary conditions on external surfaces, reflecting conditions at symmetry boundaries and steady state initial conditions which are obtained by solving

$$-\nabla\cdot D_1 \nabla\phi_1 + (\Sigma_{a,1} + \Sigma_{s,1\rightarrow 2})\phi_1 = \frac{1}{k}\sum_{g=1}^{2} \nu\Sigma_{f,g}\phi_g, \tag{4.2}$$

$$-\nabla\cdot D_2 \nabla\phi_2 + \Sigma_{a,2}\phi_2 = \Sigma_{s,1\rightarrow 2}\phi_1. \tag{4.3}$$

The eigenvalue $k$ is used to modify the fission cross section for the transient simulations with $\frac{1}{k}\Sigma_{f,g}, g = 1, 2$. The initial flux distribution shall be normalized such that the averaged power density

$$\bar{P} \equiv \frac{\int_{V_{core}} P(\vec{r}, t = 0) d\vec{r}}{\int_{V_{core}} d\vec{r}}, \tag{4.4}$$

where $V_{core}$ is the core region with fuels, is equal to $10^{-6} W \cdot cm^{-3}$. The initial precursor concentrations are in equilibrium with the initial critical flux distribution.

The geometry is illustrated in Figure 4.1.

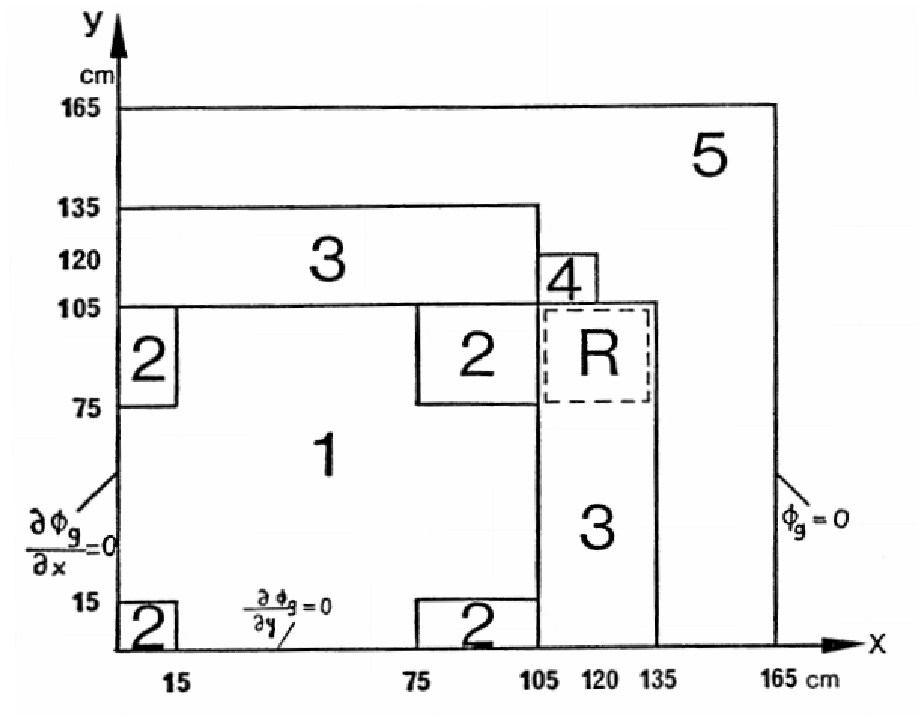

y
cm
165
135
120
105
75
15
5
3
4
2
2
R
1
3
$\frac{\partial \phi_g}{\partial x}=0$
$\frac{\partial \phi_g}{\partial y}=0$
$\phi_g=0$
2
2
15
75
105
120
135
165 cm
x

Figure 4.1: LRA benchmark geometry with region assignment [3]

Initial two-group constants are presented in Table 4.1. $\nu$ is equal to 2.43. Axial bulking $B^2 = 10^{-4}$ is applied for both energy groups. Delayed neutron data are presented in Table 4.2. All fuel materials have the same delayed neutron data. Some scalar data are listed in Table 4.3.

The transient is initiated by changing the thermal absorption cross section as the following:

$$\Sigma_{a,2}(t) = \Sigma_{a,2}(t=0) \begin{cases} 1 - 0.0606184t, & t \leq 2 \\ 0.8787631, & t > 2 \end{cases} \tag{4.5}$$

where $t$ is time in seconds.

Table 4.1: LRA benchmark initial two-group constants [3]

| Region | Material | Group g | $D_g$ (cm) | $\Sigma_{a,g}$ $(cm^{-1})$ | $\nu\Sigma_{f,g}$ $(cm^{-1})$ | $\Sigma_{s,1\to 2}$ $(cm^{-1})$ | $\chi_g$ | $v_g$ $(cm \cdot s^{-1})$ |
|---|---|---|---|---|---|---|---|---|
| 1 | Fuel 1 with rod | 1 | 1.255 | 0.008252 | 0.004602 | | 1 | $3.0 \times 10^7$ |
| | | 2 | 0.211 | 0.1003 | 0.1091 | 0.02533 | 0 | $3.0 \times 10^5$ |
| 2 | Fuel 1 without rod | 1 | 1.268 | 0.007181 | 0.004609 | | 1 | $3.0 \times 10^7$ |
| | | 2 | 0.1902 | 0.07047 | 0.08675 | 0.02767 | 0 | $3.0 \times 10^5$ |
| 3 | Fuel 2 with rod | 1 | 1.259 | 0.008002 | 0.004663 | | 1 | $3.0 \times 10^7$ |
| | | 2 | 0.2091 | 0.08344 | 0.1021 | 0.02617 | 0 | $3.0 \times 10^5$ |
| 4 | Fuel 2 without rod | 1 | 1.259 | 0.008002 | 0.004663 | | 1 | $3.0 \times 10^7$ |
| | | 2 | 0.2091 | 0.073324 | 0.1021 | 0.02617 | 0 | $3.0 \times 10^5$ |
| 5 | Reflector | 1 | 1.257 | 0.0006034 | - | | - | $3.0 \times 10^7$ |
| | | 2 | 0.1592 | 0.01911 | - | 0.04754 | - | $3.0 \times 10^5$ |

Table 4.2: LRA benchmark delayed neutron data

| Group i | $\beta_i$ | $\lambda_i$ $(s^{-1})$ | $\chi_{d,i,1}$ | $\chi_{d,i,2}$ |
|---|---|---|---|---|
| 1 | 0.0054 | 0.0654 | 1 | 0 |
| 2 | 0.001087 | 1.35 | 1 | 0 |

Table 4.3: LRA benchmark scalar values

| Meaning | Notation | value |
|---|---|---|
| Axial buckling for both energy groups | $B_g^2$ | $10^{-4}$ $(cm^{-2})$ |
| Mean number of neutrons per fission | $\nu$ | 2.43 |
| Conversion factor | $\alpha$ | $3.83 \times 10^{-11}$ $(K \cdot cm^3)$ |
| Feedback constant | $\gamma$ | $3.034 \times 10^{-3}$ $(K^{1/2})$ |
| Energy released per fission | $\kappa$ | $3.204 \times 10^{-11}$ $(J/fission)$ |
| Initial and reference temperature | $T_0$ | 300 (K) |
| Active core volume | $V_{core}$ | 17550 $(cm^2)$ |

### 4.1.1 LRA Multiphysics Time Scale Results

Figure 4.2 shows the baseline power and temperature transient profile for the LRA benchmark. Figure 4.3 shows the spacial power distribution at the peak power. The baseline results are compared to the results achieved by Sutton and Aviles in [32] and presented in Table 4.4. The relative difference in the magnitude of the peak power ($t \approx 1.44s$) from the baseline was used for error comparison. Figure 4.4a is an error convergence plot comparing the three techniques where temperature is evaluated only on the macro step (1 temperature update). Figure 4.4b is an error convergence plot comparing the three techniques where temperature is evaluated 5 times within a macro step (5 temperature updates). Finally, Figure 4.5 shows the effect of various temperature updates. The dashed lines correspond to implicit discretization at different flux step sizes, while the IQS macro step size is kept constant.

Table 4.4: LRA baseline verification

| Calculation | Baseline | Sutton (Spandex 1936) |
|---|---|---|
| No. of Spatial Nodes | 3872 | 1936 |
| Eigenvalue | 0.99637 | 0.99637 |
| No. of Time Steps | 6000 | 23,890 |
| Time to Peak Power (s) | 1.441 | 1.441 |
| Peak Power (W/cm$^3$) | 5456 | 5461 |

The convergence plots show that updating temperature and the PRKE parameters within a macro step has a significant effect on the performance of IQS. With only one update, IQS was only slightly better than implicit discretization, implicit discretization required about 150% more time steps than IQS for the same error. While 5 temperature updates showed a much more significant IQS performance, implicit discretization required

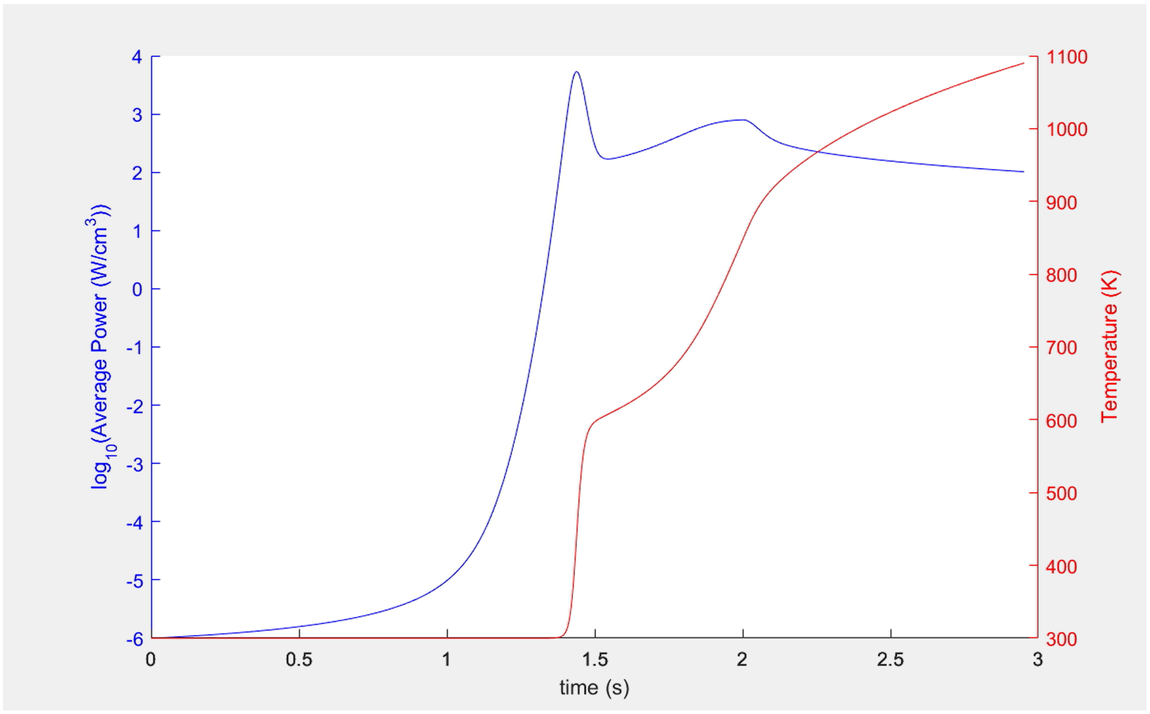


Figure 4.2: LRA baseline temperature and power profile [4]

about 400% more time steps than IQS for the same error. Figure 4.5 shows that error has a convergent behavior for the number of temperature updates. This convergence makes sense because temperature can only be so accurate before the error in shape is dominating. Table 4.5 shows the run time results for the implicit discretization calculations. The number of GMRES linear iterations is included because it is proportional measure of the computational effort. Tables 4.6 and 4.7 present the IQS run-times with various numbers of temperature updates. These run-times are based on total alive time of the execution where the diffusion evaluation is distributed over 24 processors. These run-times show a marginal performance for IQS and impressive performance for IQS P-C. Some of the execution times were able to decrease from implicit discretization with the same number of macro steps because IQS is better equipped to resolve the nonlinearity between temperature and amplitude. Furthermore, there does seem to be an ideal number of temperature

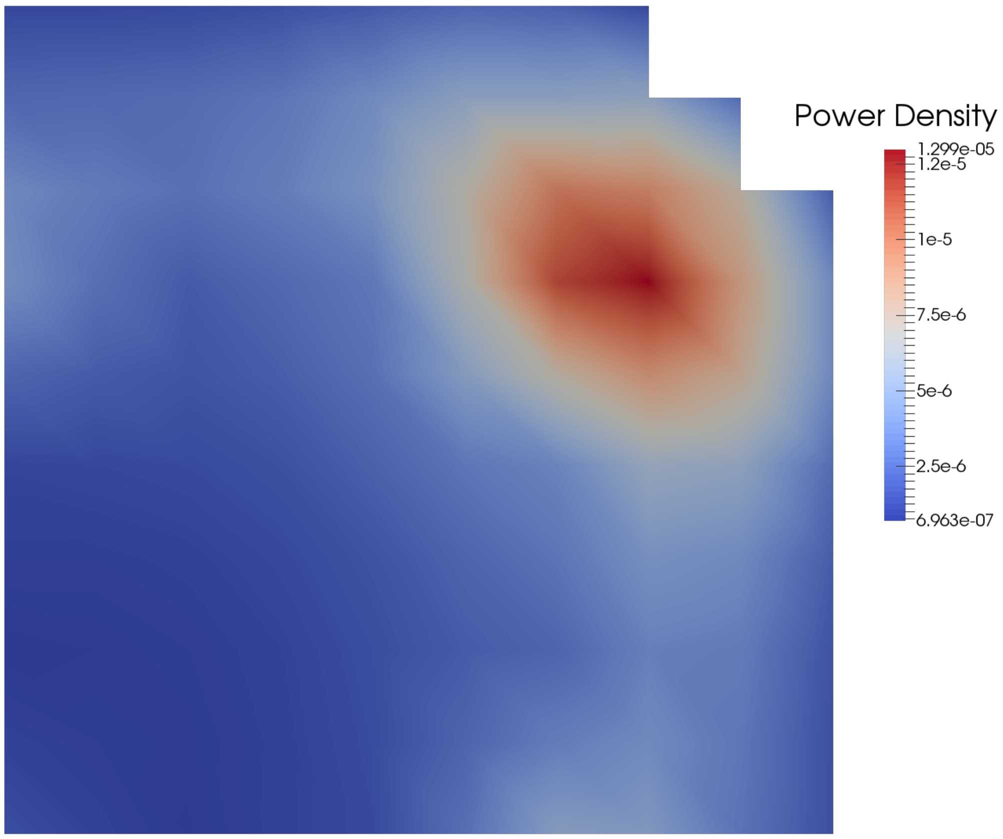
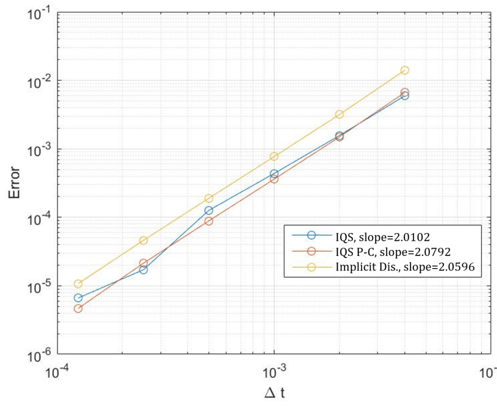
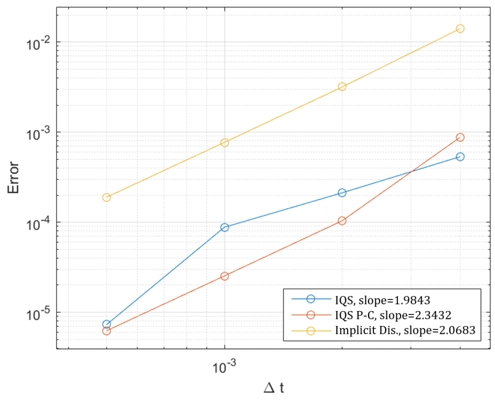


Figure 4.3: LRA baseline spacial power profile at $t = 1.44s$ [4]



(a) Only one temperature update per macro step



(b) Five temperature updates per macro step

Figure 4.4: LRA error convergence plots [4]

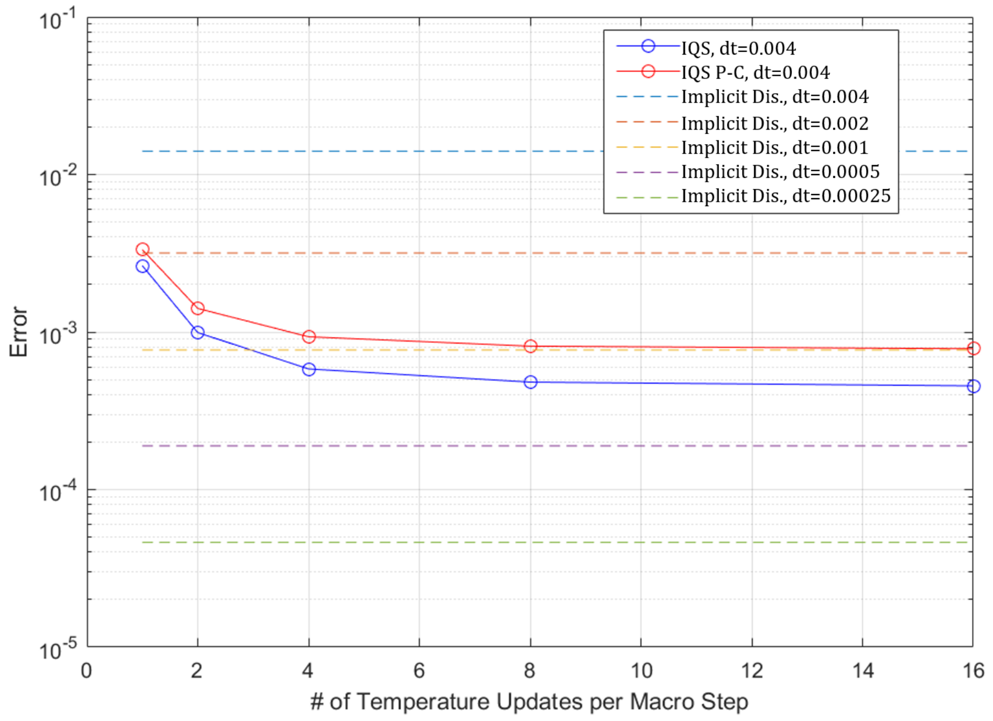


Figure 4.5: Error plot with various temperature updates per macro step [4]

updates to optimize execution time: IQS only needs one and IQS P-C seems to be ideal at 4 updates. This discrepancy in the number of updates shows that a adaptive type implementation of the updates would be ideal, and could enforce a constant error over the transient. It is also important to compare the error of implicit discretization with IQS at one update and IQS P-C at 4 updates. IQS shows an error comparable to implicit discretization at $\Delta t = 0.002$, signifying an actual increase in runtime by -34.1%. IQS P-C shows an error less than implicit discretization at $\Delta t = 0.002$, signifying an actual increase in runtime by <-34.9%.

Table 4.5: Implicit discretization run time results

| Run | $\Delta t$ | Error | Runtime (hr) | Linear Iter. |
|---|---|---|---|---|
| 1 | 4.0e-3 | 1.407e-2 | 4.11 | 7.13e4 |
| 2 | 2.0e-3 | 3.174e-3 | 6.01 | 9.49e4 |
| 3 | 1.0e-3 | 7.690e-4 | 10.38 | 1.45e5 |
| 4 | 5.0e-4 | 1.892e-4 | 21.91 | 2.08e5 |
| 5 | 2.5e-4 | 4.590e-5 | 25.23 | 3.16e5 |

Table 4.6: IQS run time results with $\Delta t = 0.004$

| | Temperature | | Runtime | % Increase |
|---|---|---|---|---|
| Run | Updates | Error | (hr) | in Runtime* |
| 1 | 1 | 2.612e-3 | 3.96 | -3.18% |
| 2 | 2 | 9.893e-4 | 6.02 | 47.1% |
| 3 | 4 | 5.796e-4 | 7.87 | 92.3% |
| 4 | 8 | 4.772e-4 | 12.61 | 207.9% |
| 5 | 16 | 4.516e-4 | 22.14 | 440.7% |

* difference in runtime from $\Delta t = 0.004$ implicit discretization

Table 4.7: IQS PC run time results with $\Delta t = 0.004$

| Run | Temperature Updates | Error | Runtime (hr) | % Increase in Runtime* |
|---|---|---|---|---|
| 1 | 1 | 3.488e-3 | 2.91 | -28.9% |
| 2 | 2 | 1.349e-3 | 3.73 | -9.00% |
| 3 | 4 | 9.161e-4 | 3.97 | -3.04% |
| 4 | 8 | 8.052e-4 | 5.39 | 31.7% |
| 5 | 16 | 7.905e-4 | 8.19 | 100% |

$^*$ difference in runtime from $\Delta t = 0.004$ implicit discretization

The performance of the quasi-statics can also be explained by the computation of the dynamical time scale described by Section 2.4.3. Figure 4.6 shows the time scale profile over the transient, computed using Equation (2.31). This plot shows that in a implicit discretization simulation, the flux dominates the time dependent behavior, while temperature lags in its variance for the majority of the transient. In an IQS simulation, the time scale behavior of amplitude almost exactly matches the flux, while shape is more varying than temperature throughout most of the transient. The large $\tau$ for temperature during the beginning of the transient shows that adaptation of the number of updates is important; computational expense on temperature evaluations is being wasted during this time.

### 4.1.2 LRA with Time Adaptation

Figure 4.7 shows the power profile of the LRA with time adaptation of implicit discretization and IQS P-C, and Table 4.8 compiles the results. These time adaptation results show the significant decrease in macro time steps required for IQS P-C. These profiles were obtaining with only one temperature update per macro step; so based on previous results, the IQS P-C performance would improve even more with more updates.

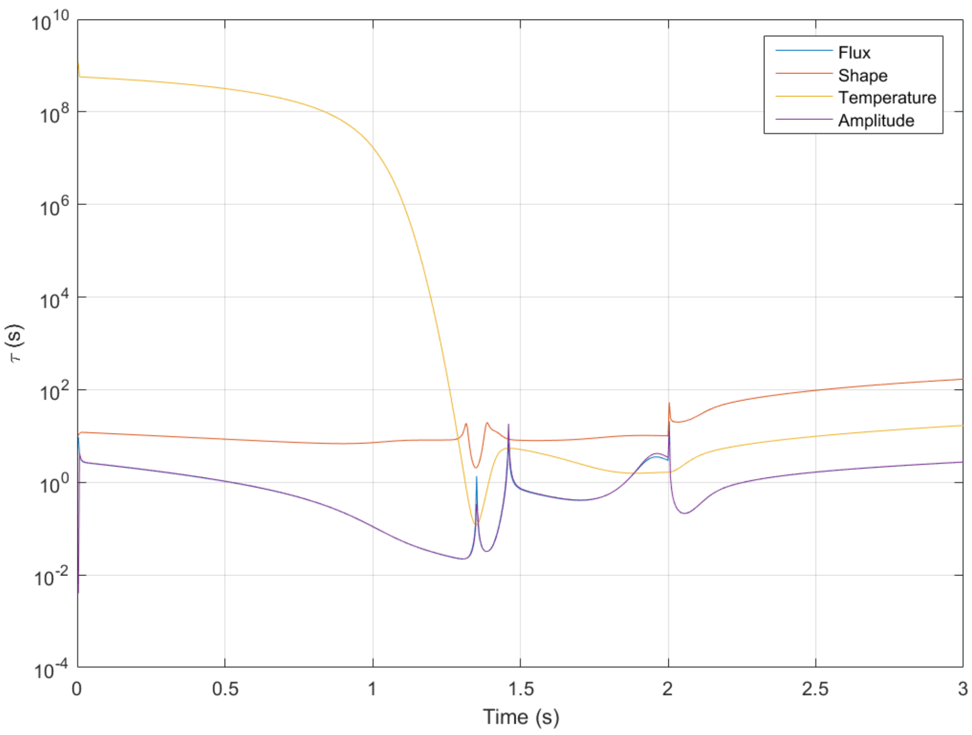
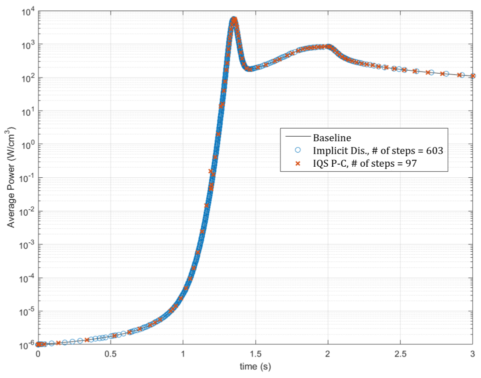
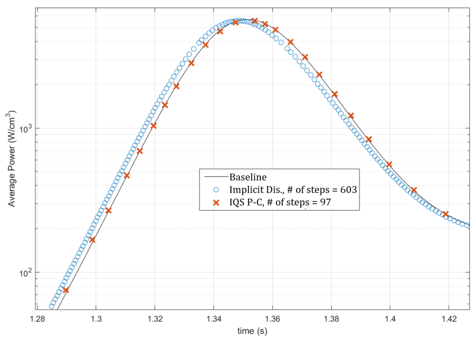


Figure 4.6: Dynamical time scale for LRA benchmark [4]



(a) Full power profile



(b) Peak power profile at peak

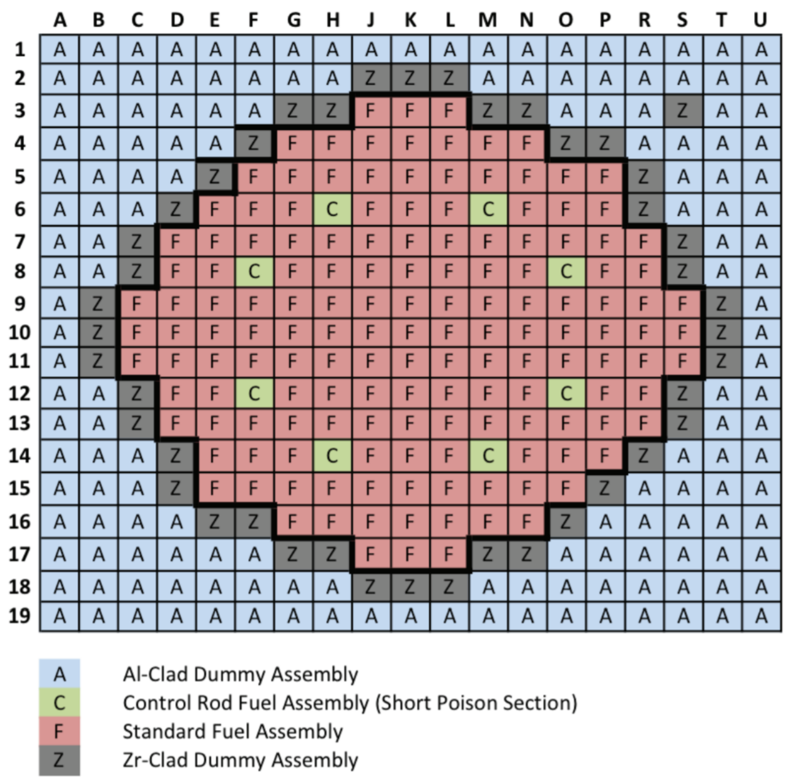
Figure 4.7: LRA power profile with time adaptation of implicit discretization and IQS P-C

Table 4.8: LRA step doubling adaptation results with implicit discretization and IQS P-C

| | Implicit Dis. | | | IQS P-C | | |
|---|---|---|---|---|---|---|
| Event | Power (W/cm$^3$) | Error | Steps | Power (W/cm$^3$) | Error | Steps |
| Max Power | 5567.3 | 0.019454 | 423 | 5568.3 | 0.019274 | 47 |
| End (3 s) | 109.66 | 2.3650e-4 | 603 | 109.65 | 3.0622e-4 | 97 |

## 4.2 TREAT Transient-15 Problem

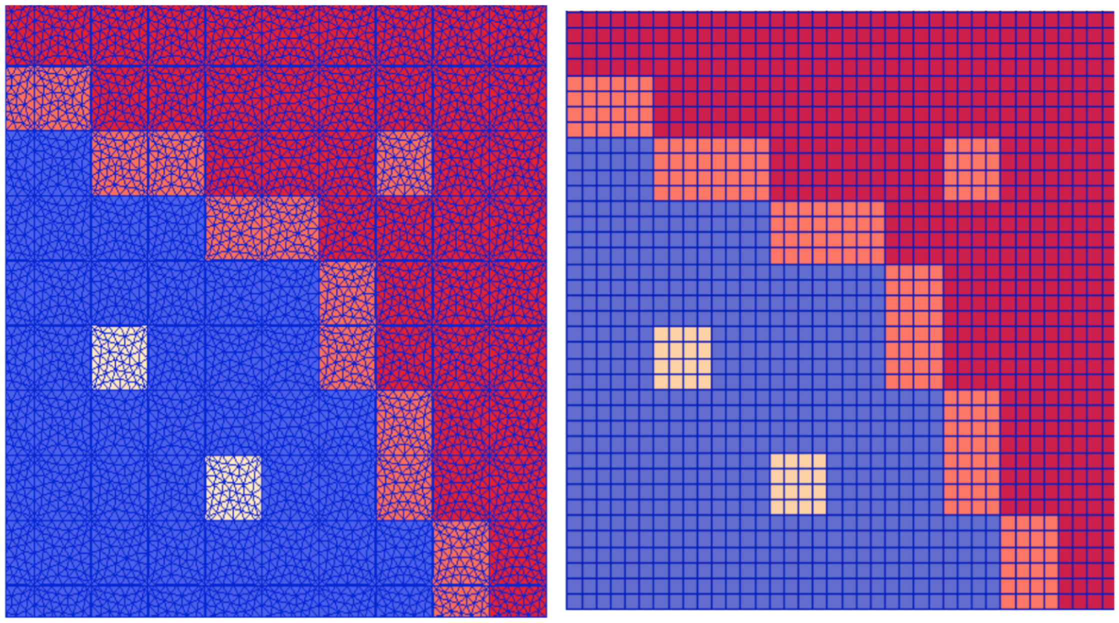
Transient 15 is a test case based on the TREAT core. The preliminary purpose of this model was to match an early test of TREAT; due to lack of experimental data and procedures, model validation was impossible [5]. Therefore, ultimate purpose of this simulation in Rattlesnake is to test the model's fidelity with the thermal feedback of TREAT, but it is not meant to exactly match any previous experiments. Nevertheless, the goal of the following simulations is to test IQS and its time scale based treatment of temperature with a more complex model. The model involves a 159-element "small core" configuration of TREAT, shown in Figure 4.8 [33]. Figure 4.9 shows the meshing techniques of the model [5]. Figure 4.10 shows the single block mesh for each meshing technique, the red area of Figure 4.10a represents the air gap. To ease computation, the fully homogenized version (without air gaps) was used for the following simulations. Transient 15 involves an 11-energy group diffusion approximation and is discretized into $355,712$ hexahedral continuous finite elements totaling $4,109,523$ degrees of freedom. The three-second transient involves a linear ramp decrease in the absorption cross section throughout the control rod region. Figure 4.11 shows a visualization of the flux profile within the core, hidden is the massive amount of graphite surrounding the core.



| | A | B | C | D | E | F | G | H | J | K | L | M | N | O | P | R | S | T | U |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
| **2** | A | A | A | A | A | A | A | A | Z | Z | Z | A | A | A | A | A | A | A | A |
| **3** | A | A | A | A | A | A | Z | Z | F | F | F | Z | Z | A | A | A | Z | A | A |
| **4** | A | A | A | A | A | Z | F | F | F | F | F | F | F | Z | Z | A | A | A | A |
| **5** | A | A | A | A | Z | F | F | F | F | F | F | F | F | F | F | Z | A | A | A |
| **6** | A | A | A | Z | F | F | F | C | F | F | F | C | F | F | F | Z | A | A | A |
| **7** | A | A | Z | F | F | F | F | F | F | F | F | F | F | F | F | F | Z | A | A |
| **8** | A | A | Z | F | F | C | F | F | F | F | F | F | F | C | F | F | Z | A | A |
| **9** | A | Z | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | Z | A |
| **10** | A | Z | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | Z | A |
| **11** | A | Z | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | Z | A |
| **12** | A | A | Z | F | F | C | F | F | F | F | F | F | F | C | F | F | Z | A | A |
| **13** | A | A | Z | F | F | F | F | F | F | F | F | F | F | F | F | F | Z | A | A |
| **14** | A | A | A | Z | F | F | F | C | F | F | F | C | F | F | F | Z | A | A | A |
| **15** | A | A | A | Z | F | F | F | F | F | F | F | F | F | F | Z | A | A | A | A |
| **16** | A | A | A | A | Z | Z | F | F | F | F | F | F | F | Z | A | A | A | A | A |
| **17** | A | A | A | A | A | A | Z | Z | F | F | F | Z | Z | A | A | A | A | A | A |
| **18** | A | A | A | A | A | A | A | A | Z | Z | Z | A | A | A | A | A | A | A | A |
| **19** | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |

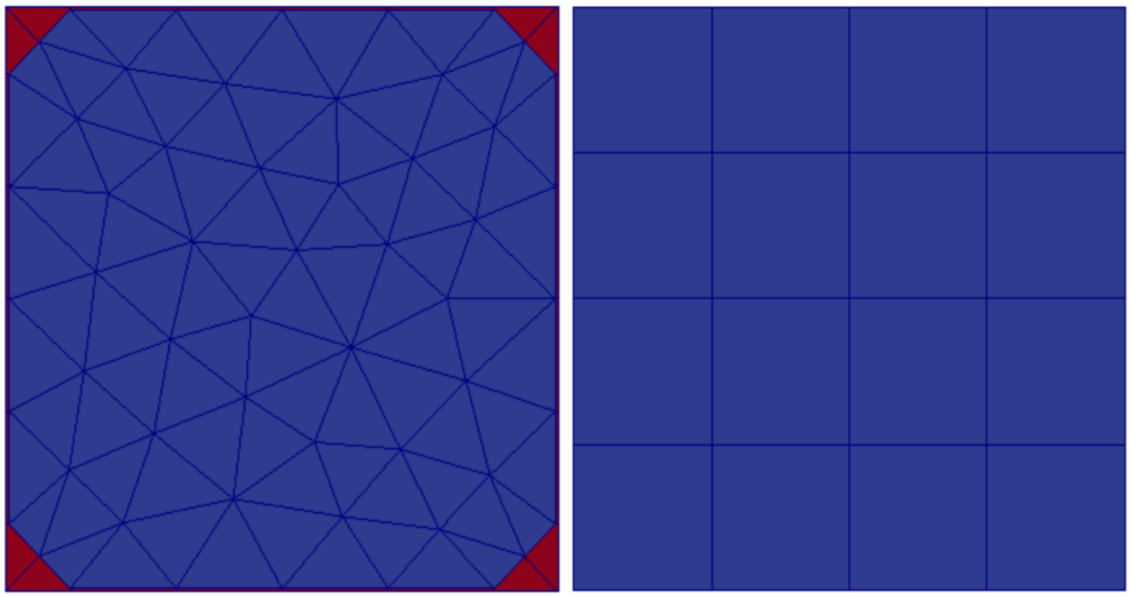
| | |
|---|---|
| A | Al-Clad Dummy Assembly |
| C | Control Rod Fuel Assembly (Short Poison Section) |
| F | Standard Fuel Assembly |
| Z | Zr-Clad Dummy Assembly |

Figure 4.8: Transient-15 159-element small core configuration [5]



(a) Explicit meshing of air channels (b) Fully homogenized fuel elements

Figure 4.9: Top quarter view of Transient-15 mesh [5]



(a) Explicit meshing of air channels (b) Fully homogenized fuel elements

Figure 4.10: Top single block view of Transient-15 mesh [5]

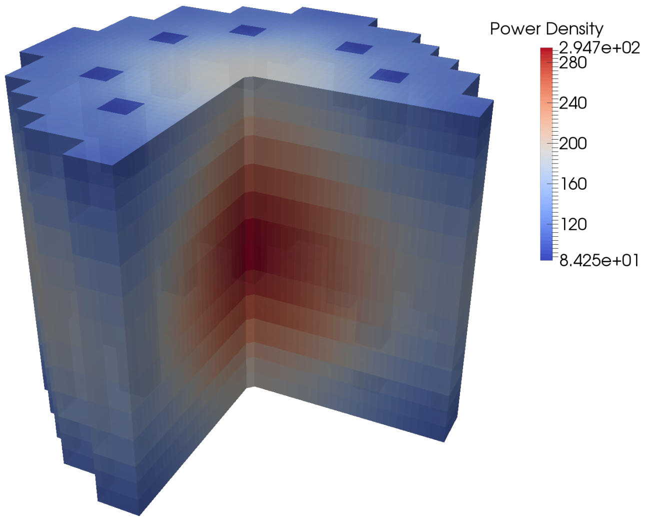


Figure 4.11: Transient-15 core power profile at peak power [4]

### 4.2.1 Transient-15 Temperature Feedback

The Transient-15 model uses a adiabatic temperature feedback mechanism, similar to the one explored by the LRA. Equation (4.6) describes the heat up of the fuel. It is very similar, except the specific heat is now dependent on temperature, which is described by Equation (4.7). The temperature evaluation is identical to the one described in LRA section, except a Newton iteration process is employed to resolve the nonlinearity from the specific heat term. The feedback to the cross-sections are applied using linear interpolation of tabular data provided by INL.

$$\rho c_p(T)\frac{\partial T(\vec{r},t)}{\partial t} = \kappa_f \sum_{g=1}^{G} \Sigma_f^g \phi^g(\vec{r},t) \tag{4.6}$$

$$\begin{aligned} c_p = &-5.8219 \times 10^{-10} T^3 - 4.3694 \times 10^{-7} T^2 \\ &+ 2.8369 \times 10^{-3} T - 1.009 \times 10^{-2} \ (J/g/K) \end{aligned} \tag{4.7}$$

### 4.2.2 Transient-15 Multiphysics Time Scale Results

In order to test the temperature feedback treatment, six different scenarios were run: a baseline with a very small time step, implicit discretization, IQS with one and 5 temperature updates per macro step, and IQS P-C with one and 5 updates. Figure 4.12 shows the baseline power and temperature profile for the Transient-15 example. Table 4.9 shows the error and runtime results.

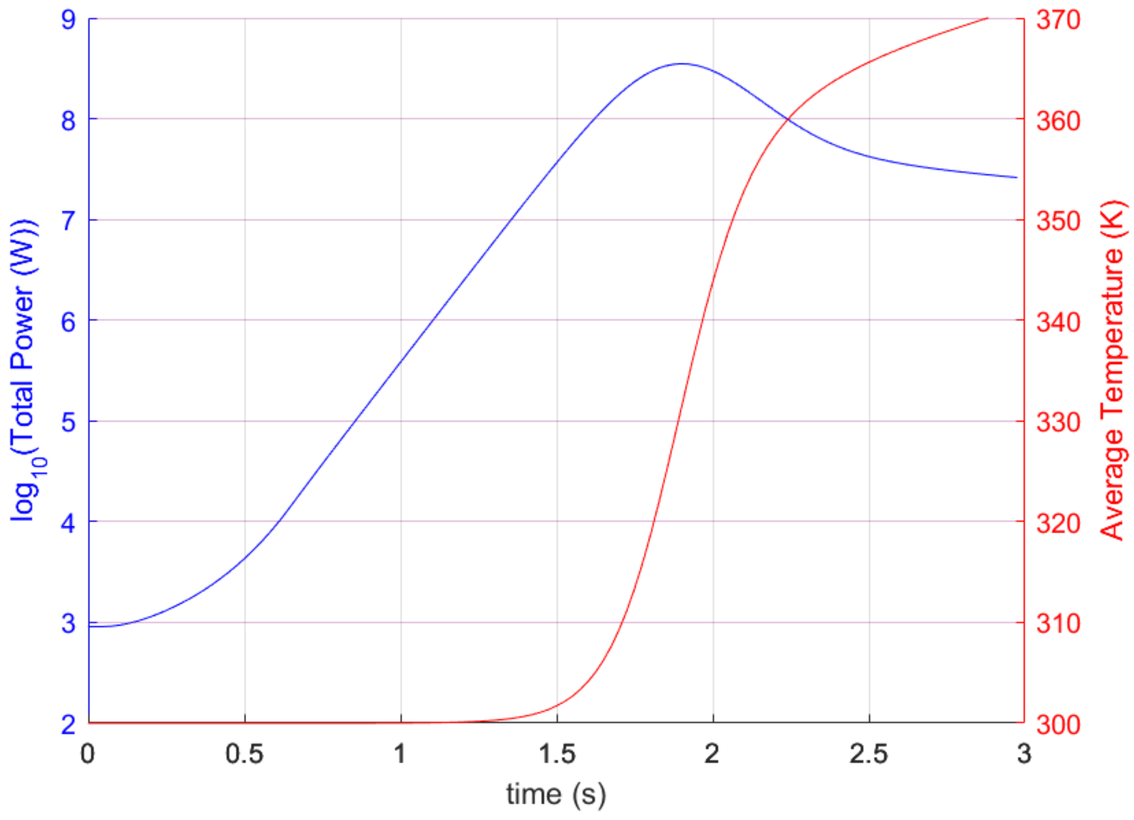


Figure 4.12: Transient-15 total power and average temperature profile during transient [4]

The results from Table 4.9 show similar performance of IQS with the temperature updates as the LRA. Again, the number of linear GMRES iterations is shown as a mea-

Table 4.9: Transient-15 error and runtime results

| Method | No. of Steps | Max Power (W) | Time at Max Power (s) | Max Average Temperature (K) | % Increase Runtime* | Max Power Error | Linear Iterations |
|---|---|---|---|---|---|---|---|
| Baseline | 3000 | 3.5039e+08 | 1.901 | 371 | — | — | — |
| Implicit Dis. | 300 | 3.5011e+08 | 1.90 | 371 | — | 7.875e-4 | 41020 |
| IQS | 300 | 3.5036e+08 | 1.90 | 371 | -11.9% | 8.385e-5 | 23949 |
| IQS (5 updates) | 300 | 3.5040e+08 | 1.90 | 371 | 49.7% | 3.687e-5 | 24035 |
| IQS P-C | 300 | 3.5065e+08 | 1.90 | 371 | -2.1% | 7.527e-4 | 39020 |
| IQS P-C (5 updates) | 300 | 3.5043e+08 | 1.90 | 371 | 26.5% | 1.227e-4 | 37866 |

* difference in runtime from implicit discretization

sure of computational expense. However, these iterations do not consider the temperature updates, so the iterations of the simulations with multiple updates should be taken with a grain of salt. IQS with 1 temperature update shows a performance that reduces the error to approximately a tenth of the implicit discretization error, and reduces the execution time by about 12%. This shows that IQS was able to resolve the nonlinearity between flux and temperature with significantly fewer diffusion evaluations. Having IQS with 5 updates significantly increased the execution time for the same time step, but the error was reduced. Comparing this error to a similar implicit discretization error at a smaller time step could show that the runtime was reduced. IQS P-C performed not nearly as well as it did with the LRA benchmark, but still proved to be effective. Having 5 updates for IQS P-C increased the runtime marginally, but decreased the error significantly. The transient profile of the variables' dynamical time scales is shown in Figure 4.13. This plot exhibits a similar response to that of the LRA. The response of temperature shows that the updates are a computational frugal treatment of the feedback and adaptation of the number of updates is vital for optimization.

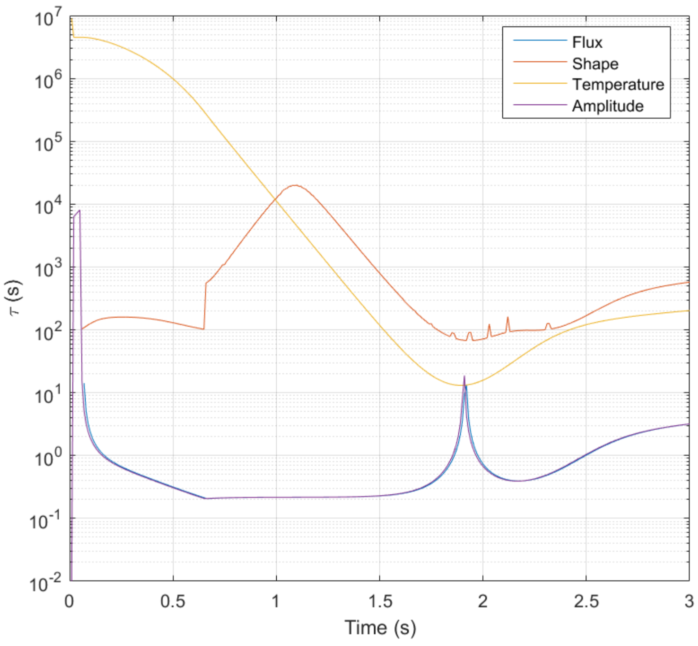
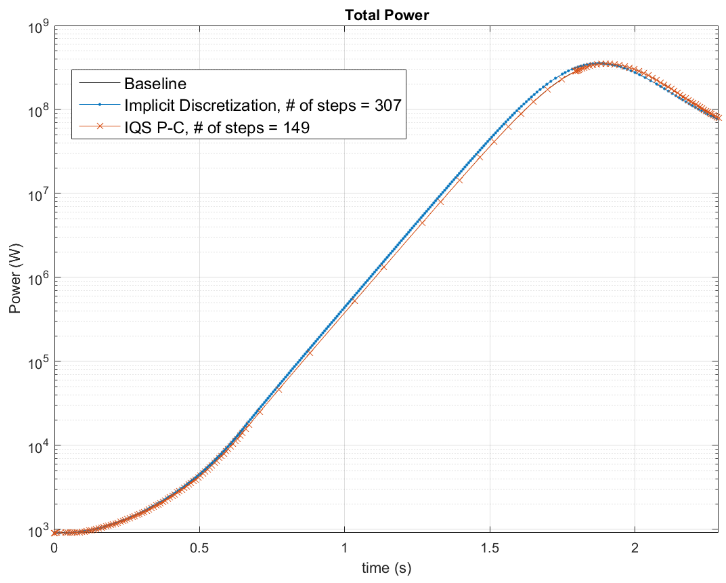


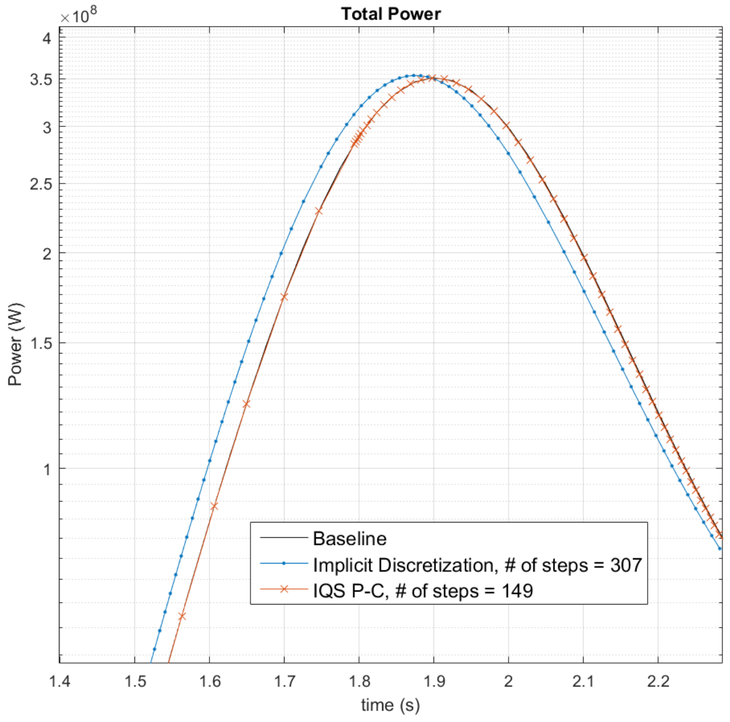
Figure 4.13: Dynamical time scale for the Transient-15 example [4]

### 4.2.3 Transient-15 with Time Adaptation

Figure 4.14 shows the power profile of the LRA with time adaptation of implicit discretization and IQS P-C. These plots show that IQS P-C requires marginally fewer macro time steps than implicit discretization, but is qualitatively much closer to the baseline profile. Like the LRA step doubling results, IQS P-C only performs one temperature update per macro step. Adding more updates would most likely improve the error, but increase the computation time significantly.



(a) Full power profile



(b) Peak power profile

Figure 4.14: Transient-15 step doubling adaptation results with implicit discretization and IQS P-C

# 5. CONCLUSIONS

The goal of this thesis research was to continue the investigation and development of the improved quasi-static method for the optimization of transient reactor simulations. In pursuit of this goal, three objectives were formulated: establish IQS behavior for various iteration techniques, validate time step convergence for IQS, and apply IQS to multiphysics simulations. The following three sections describe the purpose of each objective, as well as conclusions on their results.

## 5.1 Iteration Convergence Analysis

IQS is a nonlinear system of equations; therefore, either fixed-point or Newton iteration were needed to evaluate shape and amplitude. Through literary review, most IQS applications used fixed-point iteration with various convergence criteria. Previous application of the Newton method is not discussed in detail, nor is any analysis of its convergence presented. Investigating these iteration techniques is important for understanding the behavior of IQS, as well as determining the most appropriate technique for optimal performance. The iteration techniques were applied to a one-dimensional prototype problem for testing. For fixed-point iteration, five different criteria were tested:

1. $L^\infty$ norm of the change in shape between iterations
2. $L^2$ norm of the change in shape between iterations
3. Difference in reactivity between iterations
4. Difference in amplitude between iterations
5. IQS uniqueness consistency criteria

The results showed that criteria 1-4 had relatively equivalent convergence behavior. However, iteration with the criteria 5 could not converge without a highly-accurate analytical treatment of the precursor equation. This criteria proved to be the most rigorous and thorough convergence criteria and is recommended for any application of IQS.

## 5.2 Time Step Convergence Analysis

Validating IQS by demonstrating proper error convergence is essential for predicting model error and using time adaptation techniques. Implicit Euler and BDF discretization schemes were applied to the one-dimensional prototype problem. IQS showed proper error convergence up through fourth order BDF schemes. Implicit Euler and second order BDF schemes were applied to the TWIGL benchmark, where IQS showed normal error convergence. Step doubling adaptation was also applied to TWIGL, where IQS performed impressively, reducing the number of diffusion evaluations considerably. The Crank-Nicholson scheme was applied to the LRA benchmark with IQS. IQS again showed expected second order convergence and was able to reduce the number of diffusion evaluations with time adaptation by more than a factor of 6. The results of the TWIGL and LRA benchmarks proved that IQS is capable of proper convergence and time adaptation behavior for more complex problems.

## 5.3 IQS Application to Multiphysics

Full transient reactor simulation requires the implementation of multiphysics solution methods. In order to test IQS with these types of simulations, it was implemented into the MOOSE/Rattlesnake framework and executed using MAMMOTH. The LRA benchmark and a full core TREAT model were evaluated as test cases for multiphysics treatment. These examples involve a adiabatic heat up of the fuel with Doppler broadening feedback of cross-sections. The evaluation of temperature was integrated in the quasi-static process by introducing an intermediate time scale for temperature and PRKE parameter evaluation. The results of the LRA benchmark showed that quasi-static approach to temperature greatly improved the accuracy for a given time step size. However, the increase in computation time due to the extra temperature evaluations was significant. Therefore, for this benchmark, IQS with one update and IQS P-C with four updates is optimal. The results of the TREAT example showed similar behavior, except the updates only produced a marginal increase in accuracy. These examples' results and the dynamical time scale analysis show that performing a variant number of updates during the transient could further improve the performance of this quasi-static process. In conclusion, IQS and the quasi-static treatment of temperature can drastically improve the computational efficiency for burdensome full transient reactor simulations.

## 5.4 Recommendations for Further Research

This research intended to investigate IQS performance for transient reactor simulation and develop means for maximizing computational efficiency for these simulations. Further improvement and testing of IQS can be done in the following areas:

1. This thesis only applied IQS to fuel temperature feedback models. It is recommended that IQS be applied and tested with other feedback mechanisms such as thermal hydraulic and structural feedback, especially within the quasi-static process.

2. Application of IQS to neutron transport was hinted at in Section 2.1.1, but diffusion is typically sufficient for reactor simulation. It is recommended that IQS be applied to neutron transport with multiphysics and tested to see if similar performance is observed.

3. The large systems of full reactor models require a linear iteration process, such as GMRES, to evaluate efficiently. These processes usually implement a preconditioner to minimize the number of linear iterations and improve computation time.

Investigating of an optimal preconditioner would further improve IQS performance in terms of computational efficiency.

REFERENCES

[1] Z. M. Prince, J. C. Ragusa, and Y. Wang, "Implementation of the improved quasi-static method in Rattlesnake/MOOSE for time-dependent radiation transport modelling," in *Physics of Reactors 2016 (PHYSOR 2016): Unifying Theory and Experiments in the 21st Century*, 2016.

[2] Y. Ban, T. Endo, and A. Yamamoto, "A unified approach for numerical calculation of space-dependent kinetic equation," *Journal of Nuclear Science and Technology*, vol. 49, pp. 496–515, 2012.

[3] Argonne Code Center, "Benchmark problem book, ANL-7416, Suppl. 2," tech. rep., Argonne National Laboratory, 1977.

[4] Z. M. Prince and J. C. Ragusa, "Multiphysics core dynamics simulation using the improved quasi-static method," in *Proc. International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea*, 2017.

[5] J. Ortensi, M. D. DeHart, F. N. Gleicher, Y. Wang, A. L. Alberti, and T. S. Palmer, "Full core TREAT kinetics demonstration using Rattlesnake/BISON coupling within MAMMOTH," Tech. Rep. INL/EXT-15-36268, Idaho National Laboratory, Idaho Falls, Idaho, Sep 2015.

[6] "Development of light water reactor fuels with enhanced accident tolerance," tech. rep., United States Department of Energy, Washington, DC 20585, 04 2015.

[7] E. J. Parma, M. E. Vernon, *et al.*, "Global nuclear energy partnership fuels transient testing at the Sandia National Laboratories nuclear facilities: Planning and facility

infrastructure options," tech. rep., Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US), 2007.

[8] "Environmental assessment for the resumption of transient testing of nuclear fuels and materials," tech. rep., Department of Energy, Idaho Opoerations Office, Idaho Falls, ID 83415, 02 2014.

[9] E. Fruend *et al.*, "Design summary report on the Transient Reactor Test Facility (TREAT), ANL-6034," tech. rep., Agonne National Laboratory, Argonne, IL, 09 1960.

[10] J. J. Duderstadt and L. J. Hamilton, *Nuclear reactor analysis*. Wiley, 1976.

[11] O. Zeinkiewicz, R. Taylor, and J. Zhu, *The finite element method: its basis and fundamentals*. Selsevier Butterworth-Heinemann, 2005.

[12] L. Hageman and J. Yasinsky, "Comparison of alternating direction time differencing method with other implicit method for the solution of the neutron group diffusion equations," *Nucl. Sci. Eng.*, vol. 38, pp. 8 – 32, 1969.

[13] J. Planchard, "On the point-reactor kinetics approximation," *Progress in Nuclear Energy*, vol. 26, pp. 207–216, 1991.

[14] K. Ott, "Quasi-static treatment of spatial phenomena in reactor dynamics," *Nuclear Science and Engineering*, vol. 26, p. 563, 1966.

[15] S. Dulla, E. H. Mund, and P. Ravetto, "The quasi-static method revisited," *Progress in Nuclear Energy*, vol. 50, no. 8, pp. 908 – 920, 2008.

[16] J. Devooght, B. Arien, E. H. Mund, and A. Siebertz, "Fast reactor transient analysis using the generalized quasi-static approximation," *Nuclear Science and Engineering*, vol. 88, pp. 191–199, 1984.

[17] A. Monier, *Application of the Collocation Technique to the Spatial Discretization of the Generalized Quasistatic Method for Nuclear Reactors*. PhD thesis, Université de Montréal, 1991.

[18] M. Sissnoui, J. Koclas, and A. Hébert, "Solution of the improved and generalized quasistatic methods by Kaps and Rentrop integration scheme with stepsize control," *Annals of Nuclear Energy*, vol. 22, pp. 763–774, 1995.

[19] K. Ott and D. Meneley, "Accuracy of the quasistatic treatment of spatial reactor kinetics," *Nuclear Science and Engineering*, vol. 36, pp. 381–419, 1969.

[20] J. Koclas, M. Sissnoui, and A. Hébert, "Solution of the improved and generalized quasistatic methods using an analytic calculation or a semi-implicit scheme to compute the precursor equations," *Annals of Nuclear Energy*, vol. 23, pp. 901–907, 1996.

[21] D. Gaston, C. Newman, G. Hansen, , and D. Lebrun-Grandie, "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nucl. Engrg. Design*, vol. 239, pp. 1768 – 1778, 2009.

[22] Y. Wang, "Nonlinear diffusion acceleration for multigroup transport equation discretized with SN and continuous FEM with Rattlesnake," in *Proc. International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering, Idaho*, 2013.

[23] R. Williamson, J. Hales, S. Novascone, M. Tonks, D. Gaston, C. Permann, D. Andrs, and R. Martineau, "Multidimensional multiphysics simulation of nuclear fuel behavior," *Journal of Nuclear Materials*, vol. 423, no. 1-3, pp. 149 – 163, 2012.

[24] H. Zhang, H. Zhao, F. N. Gleicher, M. D. DeHart, L. Zou, D. Andrs, and R. C. Martineau, *RELAP-7 Development Updates*. Sep 2015.

[25] D. A. Knoll and D. E. Keyes, “Jacobian-free Newton-Krylov methods: a survey of approaches and applications,” *Journal of Computational Physics*, vol. 193, pp. 357–397, 2004.

[26] H. Ikeda and T. Takeda, “Development and verification of an efficient spatial neutron kinetics method for reactivity-initiated event analyses,” *Journal of Nuclear Science and Technology*, vol. 38, pp. 496–515, 2001.

[27] S. Goluoglu and H. L. Dodds, “A time-dependent, three-dimensional neutron transport methodology,” *Nuclear Science and Engineering*, vol. 139, pp. 248–261, 2001.

[28] J. H. Ferziger, *Numerical Methods for Engineering Application*. John Wiley and Sons, Inc., New York, NY, 1998.

[29] B. Gear, “Backward differentiation formulas,” vol. 2, no. 8, p. 3162, 2007. revision 91024.

[30] J. M. Franco, I. Gómez, and L. Rández, “SDIRK methods for stiff ODEs with oscillating solutions,” *Journal fo Computational and Applied Mathematics*, vol. 81, pp. 197–209, 1997.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[32] T. M. Sutton and B. N. Aviles, “Diffusion theory methods for spatial kinetics calculations,” *Progress in Nuclear Energy*, vol. 30, pp. 119–182, 1996.

[33] J. F. Kirn, J. Boland, H. Lawroski, and R. Cook, “Reactor physics measurements in TREAT,” Tech. Rep. ANL-6173, Agonne National Laboratory, Argonne, IL, Oct 1960.

# APPENDIX A

# DERIVATIONS

## A.1 Derivation of IQS with CFEM Diffusion

Multigroup diffusion equation with delayed neutron precrusors:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{\phi^g}{v^g}\right) =&\chi_p^g \sum_{g'=1}^{G}(1-\beta)\frac{\nu^{g'}\Sigma_f^{g'}}{k_{\mathit{eff}}}\phi^{g'} - \left(-\nabla\cdot D^g\nabla + \Sigma_r^g\right)\phi^g \\ &+ \sum_{g'\neq g}^{G}\Sigma_s^{g'\to g}\phi^{g'} + \sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i\ , \quad 1\le g\le G\end{aligned} \tag{A.1}$$

$$\frac{dC_i}{dt} = \beta_i\sum_{g=1}^{G}\frac{\nu^{g'}\Sigma_f^{g'}}{k_{\mathit{eff}}}\phi^g - \lambda_i C_i\ , \quad 1\le i\le I \tag{A.2}$$

With reflecting and vacuum boundary conditions:

$$\begin{cases}\nabla\phi^g = 0, & \vec{r}\in\partial\mathcal{D}_2 \\ D\nabla\phi^g - \frac{1}{2}\phi^g = 0, & \vec{r}\in\partial\mathcal{D}_3\end{cases} \tag{A.3}$$

$$\text{with } \partial\mathcal{D} = \partial\mathcal{D}_2\cup\partial\mathcal{D}_3$$

Multiplying Equation A.1 by the adjoint solution $\phi^{*g}$ and Equation A.2 by $C^* = \sum_{g=1}^{G}\phi^{*g}\chi_{d,i}^g$ for test functions and integrating in space:

$$\frac{d}{dt}\left(\phi^{*g}, \frac{1}{v^g}\phi^g\right)_{\mathcal{D}} = \left(\phi^{*g}\chi_p^g(1-\beta), \sum_{g'=1}^{G}\frac{\nu^{g'}\Sigma_f^{g'}}{k_{\mathit{eff}}}\phi^{g'}\right)_{\mathcal{D}} + \left(\nabla\phi^{*g}, D^g\nabla\phi^g\right)_{\mathcal{D}}$$

$$-\frac{1}{2}\left\langle \phi^{*g}, \phi^{g}\right\rangle_{\partial\mathcal{D}_3} - \left(\phi^{*g}, \Sigma_r^g \phi^g\right)_{\mathcal{D}} + \sum_{g'\neq g}^{G}\left(\phi^{*g}, \Sigma_s^{g'\to g}\phi^{g'}\right)_{\mathcal{D}} + \sum_{i=1}^{I}\left(\phi^{*g}\chi_{d,i}^{g}, \lambda_i C_i\right)_{\mathcal{D}} \tag{A.4}$$

$$\frac{d}{dt}\left(C^*, C_i\right)_{\mathcal{D}} = \left(C^*\beta_i, \sum_{g'=1}^{G}\frac{\nu^{g'}\Sigma_f^{g'}}{k_{\mathit{eff}}}\varphi^{g'}\right)_{\mathcal{D}} - \left(C^*, \lambda_i C_i\right)_{\mathcal{D}} \tag{A.5}$$

Summing over groups on both sides of Equation A.4:

$$\begin{aligned}\frac{d}{dt}\left(\sum_{g=1}^{G}\left(\phi^{*g}, \frac{1}{v^g}\phi^g\right)_{\mathcal{D}}\right) &= \left(\sum_{g=1}^{G}\phi^{*g}\chi_p^g(1-\beta), \sum_{g'=1}^{G}\frac{\nu^{g'}\Sigma_f^{g'}}{k_{\mathit{eff}}}\phi^{g'}\right)_{\mathcal{D}} \\ &+ \sum_{g=1}^{G}\left(\nabla\phi^{*g}, D^g\nabla\phi^g\right)_{\mathcal{D}} - \frac{1}{2}\sum_{g=1}^{G}\left\langle\phi^{*g}, \phi^g\right\rangle_{\partial\mathcal{D}_3} - \sum_{g=1}^{G}\left(\phi^{*g}, \Sigma_r^g\phi^g\right)_{\mathcal{D}} \\ &+ \sum_{g=1}^{G}\sum_{g'\neq g}^{G}\left(\phi^{*g}, \Sigma_s^{g'\to g}\phi^{g'}\right)_{\mathcal{D}} + \sum_{i=1}^{I}\left(\sum_{g=1}^{G}\phi^{*g}\chi_{d,i}^g, \lambda_i C_i\right)_{\mathcal{D}}\end{aligned} \tag{A.6}$$

Defining bilinear functions:

$$T(\phi^*, \phi) = \sum_{g=1}^{G}\left(\phi^{*g}, \frac{1}{v^g}\phi^g\right)_{\mathcal{D}} \tag{A.7}$$

$$C_i(C^*, C_i) = \left(C^*, C_i\right)_{\mathcal{D}} \tag{A.8}$$

$$F(\phi^*, \phi) = \left(\sum_{g=1}^{G}\phi^{*g}\left(\chi_p^g(1-\beta) + \sum_{i=1}^{I}\chi_{d,i}^g\beta_i\right), \sum_{g'=1}^{G}\frac{\nu^{g'}\Sigma_f^{g'}}{k_{\mathit{eff}}}\phi^{g'}\right)_{\mathcal{D}} \tag{A.9}$$

$$L(\phi^*, \phi) = \sum_{g=1}^{G}\left(-\left(\nabla\phi^{*g}, D^g\nabla\phi^g\right)_{\mathcal{D}} + \frac{1}{2}\left\langle\phi^{*g}, \phi^g\right\rangle_{\partial\mathcal{D}_3} + \left(\phi^{*g}, \Sigma_r^g\phi^g\right)_{\mathcal{D}}\right) \tag{A.10}$$

$$S(\phi^*, \phi) = \sum_{g=1}^{G}\sum_{g'\neq g}^{G}\left(\phi^{*g}, \Sigma_s^{g'\to g}\phi^{g'}\right)_{\mathcal{D}} \tag{A.11}$$

$$F_{d,i}(\phi^*, \phi) = \left(\beta_i\sum_{g=1}^{G}\phi^{*g}\chi_{d,i}^g, \sum_{g'=1}^{G}\frac{\nu^{g'}\Sigma_f^{g'}}{k_{\mathit{eff}}}\phi^{g'}\right)_{\mathcal{D}} \tag{A.12}$$

$$S_{d,i}(\phi^*,\phi) = \left(\sum_{g=1}^{G} \phi^{*g}\chi_{d,i}^{g}, \lambda_i C_i\right)_{\mathcal{D}} \tag{A.13}$$

$$P_i(C^*,\phi) = \left(C^*\beta_i, \sum_{g'=1}^{G} \frac{\nu^{g'}\Sigma_f^{g'}}{k_{\mathit{eff}}}\phi^{g'}\right)_{\mathcal{D}} \tag{A.14}$$

$$D_i(C^*,C_i) = (C^*, \lambda_i C_i)_{\mathcal{D}} \tag{A.15}$$

Rewriting the flux and precursor equations:

$$\frac{d}{dt}T(\phi^*,\phi) = F(\phi^*,\phi) - \sum_{i=1}^{I} F_{d,i}(\phi^*,\phi) - L(\phi^*,\phi) + S(\phi^*,\phi) + \sum_{i=1}^{I} S_{d,i}(\phi^*,\phi) \tag{A.16}$$

$$\frac{d}{dt}C_i(C^*,C_i) = P_i(C^*,\phi) - D_i(C^*,C_i) \tag{A.17}$$

Doing IQS factorization:

$$\begin{aligned}\frac{d}{dt}T(\phi^*,\varphi) = F(\phi^*,\varphi) - \sum_{i=1}^{I} F_{d,i}(\phi^*,\varphi) - L(\phi^*,\varphi) - \frac{1}{p}\frac{dp}{dt}T(\phi^*,\varphi)\\ + S(\phi^*,\varphi) + \frac{1}{p}\sum_{i=1}^{I} S_{d,i}(\phi^*,\phi)\end{aligned} \tag{A.18}$$

$$\frac{\partial}{\partial t}C_i(C^*,C_i) = P_i(C^*,\varphi)p - D_i(C^*,C_i) \tag{A.19}$$

Defining PRKE parameters:

$$\rho = \frac{F(\phi^*,\varphi) - L(\phi^*,\varphi) + S(\phi^*,\varphi)}{F(\phi^*,\varphi)} \tag{A.20}$$

$$\bar{\beta} = \sum_{i=1}^{I} \bar{\beta}_i = \sum_{i=1}^{I} \frac{F_{d,i}(\phi^*,\varphi)}{F(\phi^*,\varphi)} = \sum_{i=1}^{I} \frac{P_i(C^*,\varphi)}{F(\phi^*,\varphi)} \tag{A.21}$$

$$\Lambda = \frac{T(\phi^*,\varphi)}{F(\phi^*,\varphi)} \tag{A.22}$$

$$\bar{\lambda}_i = \frac{S_{d,i}(\phi^*, \varphi)}{C_i(C^*, C_i)} = \frac{D_i(C^*, C_i)}{C_i(C^*, C_i)} \tag{A.23}$$

$$\xi_i = \frac{C_i(C^*, C_i)}{T(\phi^*, \varphi)} \tag{A.24}$$

Writing implementing parameters:

$$\frac{dp}{dt} = \left[\frac{\rho - \bar{\beta}}{\Lambda}\right] p + \sum_{i=1}^{I} \bar{\lambda}_i \xi_i + p \frac{1}{T(\phi^*, \varphi)} \frac{\partial}{\partial t} T(\phi^*, \varphi) \tag{A.25}$$

$$\frac{d\xi_i}{dt} = \frac{\bar{\beta}_i}{\Lambda} p - \bar{\lambda}_i \xi_i \quad 1 \leq i \leq I \tag{A.26}$$

Assuming $\frac{\partial}{\partial t} T(\phi^*, \varphi) = 0$:

$$\frac{dp}{dt} = \left[\frac{\rho - \bar{\beta}}{\Lambda}\right] p + \sum_{i=1}^{I} \bar{\lambda}_i \xi_i \tag{A.27}$$

$$\frac{d\xi_i}{dt} = \frac{\bar{\beta}_i}{\Lambda} p - \bar{\lambda}_i \xi_i \quad 1 \leq i \leq I \tag{A.28}$$

## A.2 Proof of Flux-Shape Solution Inequality

There has been a reoccurring question that if shape and amplitude have same time step and integration scheme, the solution should be equal to the full flux solution. This section is meant to prove that this notion is not true. For simplicity, the integration scheme chosen was backward-Euler. Writing the shape equation in time discretized form:

$$\begin{aligned} \frac{T^{n+1}(\phi^*, \varphi) - T^n(\phi^*, \varphi)}{\Delta t} &= F^{n+1}(\phi^*, \varphi) - \sum_{i=1}^{I} F_{d,i}^{n+1}(\phi^*, \varphi) - L^{n+1}(\phi^*, \varphi) - \\ &- \frac{1}{p^{n+1}} \frac{p^{n+1} - p^n}{\Delta t} T^{n+1}(\phi^*, \varphi) + S^{n+1}(\phi^*, \varphi) + \frac{1}{p^{n+1}} \sum_{i=1}^{I} S_{d,i}^{n+1}(\phi^*, C_i) \end{aligned} \tag{A.29}$$

where $n$ is the index for the previous time step and $n+1$ is the next one. Multiplying both sides by $p^{n+1}$:

$$p^{n+1}\frac{T^{n+1}(\phi^*,\varphi)-T^{n}(\phi^*,\varphi)}{\Delta t} = F^{n+1}(\phi^*,\varphi)p^{n+1} - \sum_{i=1}^{I} F_{d,i}^{n+1}(\phi^*,\varphi)p^{n+1}$$
$$- L^{n+1}(\phi^*,\varphi)p^{n+1} - \frac{p^{n+1}-p^{n}}{\Delta t}T^{n+1}(\phi^*,\varphi) + S^{n+1}(\phi^*,\varphi)p^{n+1} + \sum_{i=1}^{I} S_{d,i}^{n+1}(\phi^*,C_i) \tag{A.30}$$

Notice that for any bilinear form with $\varphi$, when multiplied by $p^{n+1}$ is the same form, but with $\phi$ instead. So:

$$p^{n+1}\frac{T^{n+1}(\phi^*,\varphi)-T^{n}(\phi^*,\varphi)}{\Delta t} + \frac{p^{n+1}-p^{n}}{\Delta t}T^{n+1}(\phi^*,\varphi) = F^{n+1}(\phi^*,\phi)$$
$$- \sum_{i=1}^{I} F_{d,i}^{n+1}(\phi^*,\phi) - L^{n+1}(\phi^*,\phi) + S^{n+1}(\phi^*,\phi) + \sum_{i=1}^{I} S_{d,i}^{n+1}(\phi^*,C_i) \tag{A.31}$$

Now the full flux equation is discretized the same way:

$$\frac{T^{n+1}(\phi^*,\phi)-T^{n}(\phi^*,\phi)}{\Delta t} = F^{n+1}(\phi^*,\phi) - \sum_{i=1}^{I} F_{d,i}^{n+1}(\phi^*,\phi)$$
$$- L^{n+1}(\phi^*,\phi) + S^{n+1}(\phi^*,\phi) + \sum_{i=1}^{I} S_{d,i}^{n+1}(\phi^*,C_i) \tag{A.32}$$

Notice that Equations A.31 and A.32 have the same right hand side. So the left hand side of each equation can be equated, while converting the left hand side of Equation A.32 to amplitude and flux:

$$\frac{p^{n+1}T^{n+1}(\phi^*,\varphi)-p^{n}T^{n}(\phi^*,\varphi)}{\Delta t} = p^{n+1}\frac{T^{n+1}(\phi^*,\varphi)-T^{n}(\phi^*,\varphi)}{\Delta t}$$

$$+\frac{p^{n+1}-p^n}{\Delta t}T^{n+1}(\phi^*,\varphi) \tag{A.33}$$

From Equation A.27:

$$\frac{p^{n+1}-p^n}{\Delta t} = \left[\frac{\rho^{n+1}-\bar{\beta}^{n+1}}{\Lambda^{n+1}}\right]p^{n+1} + \sum_{i=1}^{I}\bar{\lambda}_i^{n+1}\xi_i^{n+1} \tag{A.34}$$

Substituting the PRKE parameter definitions:

$$\begin{aligned}\frac{p^{n+1}-p^n}{\Delta t}T^{n+1}(\phi^*,\varphi) =& F^{n+1}(\phi^*,\phi) - \sum_{i=1}^{I}F_{d,i}^{n+1}(\phi^*,\phi)\\ &- L^{n+1}(\phi^*,\phi) + S^{n+1}(\phi^*,\phi) + \sum_{i=1}^{I}S_{d,i}^{n+1}(\phi^*,C_i)\end{aligned} \tag{A.35}$$

Notice how the right hand side of Equation A.35 is equal to the right hand side of Equation A.32 so:

$$\frac{p^{n+1}-p^n}{\Delta t}T^{n+1}(\phi^*,\varphi) = \frac{p^{n+1}T^{n+1}(\phi^*,\varphi)-p^nT^n(\phi^*,\varphi)}{\Delta t} \tag{A.36}$$

Substituting this into the last term on the right hand side of Equation A.33:

$$\begin{aligned}\frac{p^{n+1}T^{n+1}(\phi^*,\varphi)-p^nT^n(\phi^*,\varphi)}{\Delta t} &= p^{n+1}\frac{T^{n+1}(\phi^*,\varphi)-T^n(\phi^*,\varphi)}{\Delta t}\\ &+ \frac{p^{n+1}T^{n+1}(\phi^*,\varphi)-p^nT^n(\phi^*,\varphi)}{\Delta t}\end{aligned} \tag{A.37}$$

Which can be written as:

$$\frac{T^{n+1}(\phi^*,\varphi)-T^n(\phi^*,\varphi)}{\Delta t} = 0 \tag{A.38}$$

Table A.1: Numerical results for inequality proof where the compared value is $(1, \phi)$

| Time(s) | Full Flux Solve | IQS Solve | Relative Difference |
|---|---|---|---|
| 0.0 | 0.90903993 | 0.90903993 | 0 |
| 0.1 | 0.91253395 | 0.90903993 | 0.00383 |
| 0.2 | 0.92068212 | 0.91341921 | 0.00789 |
| 0.3 | 0.93564944 | 0.92419479 | 0.01224 |
| 0.4 | 0.96072564 | 0.94502984 | 0.01634 |
| 0.5 | 1.00132530 | 0.98193047 | 0.01937 |
| 0.6 | 1.06731452 | 1.04589052 | 0.02007 |
| 0.7 | 1.13375988 | 1.11085442 | 0.0202 |
| 0.8 | 1.19865931 | 1.17468271 | 0.02 |
| 0.9 | 1.26133943 | 1.23659429 | 0.01962 |
| 1.0 | 1.32173672 | 1.29643586 | 0.01914 |
| 1.1 | 1.33970601 | 1.31604696 | 0.01766 |
| 1.2 | 1.32132462 | 1.30051903 | 0.01575 |
| 1.3 | 1.27831729 | 1.26067046 | 0.0138 |
| 1.4 | 1.22290272 | 1.20822518 | 0.012 |
| 1.5 | 1.16481963 | 1.15274011 | 0.01037 |
| 1.6 | 1.11035980 | 1.10047097 | 0.00891 |
| 1.7 | 1.06269788 | 1.05460529 | 0.00762 |
| 1.8 | 1.03149816 | 1.02465437 | 0.00663 |
| 1.9 | 1.01107465 | 1.00511526 | 0.00589 |
| 2.0 | 0.99776339 | 0.99243083 | 0.00534 |

However, this equation is not necessarily true. The assumption is made that $\frac{dT}{dt} = 0$ when deriving Equation A.27, but in practice this doesn't always hold up. Therefore, putting the shape and amplitude equations on the same discretization, does not necessarily mean the IQS solution will produce the same solution as the full flux equation.

To show this result numerically, the discretized scheme was applied to the one-group, one-dimensional example and five Picard iterations were done each time step for IQS. The compiled results are shown in Table A.1 by comparing the difference in the volumetric integral of the flux. The results show that there is an average $1\%$ difference between the full flux solution and IQS.

# APPENDIX B

# OTHER TESTING

## B.1 SDIRK Convergence Problems

In order to figure what is causing convergence problems for SDIRK, a manufactured solution was created in the one-dimensional prototype code, described by Equation (B.1). A second order finite element basis function was used, so the spatial profile was known exactly with only one degree of freedom. The ODE describing this degree of freedom is shown by Equation (B.2). The coefficients represented in this equation are defined by Equation (B.3).

$$\phi(x,t) = x(1-x)(1+t)^4 \qquad 0 \leq x \leq 1 \tag{B.1}$$

$$\frac{1}{v}\frac{\partial \phi}{\partial t} = \alpha\phi + \frac{\beta}{v}4(1+t)^3 + \gamma(1+t)^4 \tag{B.2}$$

$$\alpha = -D\int_0^1 (4(1-2x))^2 dx + (\nu\Sigma_f - \Sigma_a)\int_0^1 (4x(1-x))^2 dx = -5.28 \tag{B.3a}$$

$$\beta = \frac{1}{4}\int_0^1 (4x(1-x))^2 dx = 0.1333 \tag{B.3b}$$

$$\gamma = (\Sigma_a - \nu\Sigma_f)\beta + 2D\int_0^1 4x(1-x)dx = 1.32 \tag{B.3c}$$

The ODE was then evaluated using SDIRK and BDF schemes with varying velocities. Figure B.1 compares SDIRK and BDF convergence with a relatively large $v$. This plot shows the SDIRK is unable to establish a proper error convergence for such a stiff problem. Figure B.2 compares SDIRK convergence lines with different velocities. This plot shows that only at relatively small values of $v$, is SDIRK able to show proper convergence.

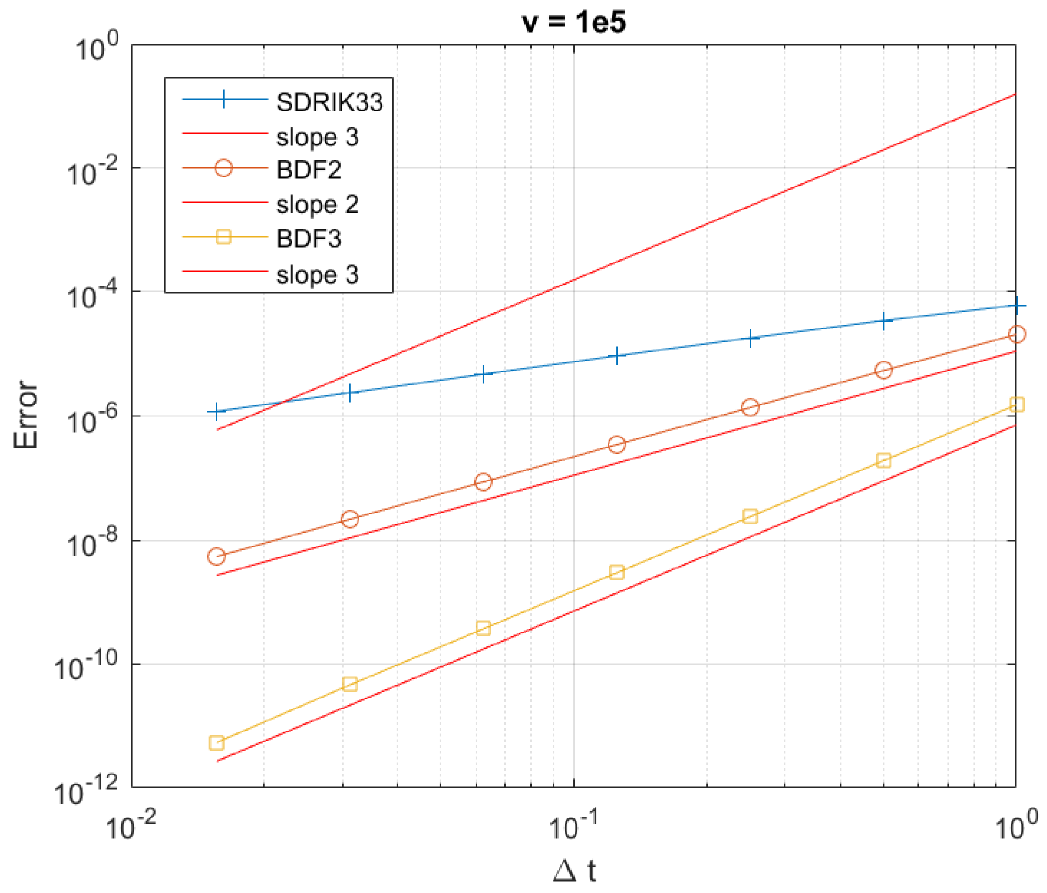


Figure B.1: SDIRK33 vs. BDF convergence with $v = 1e5$

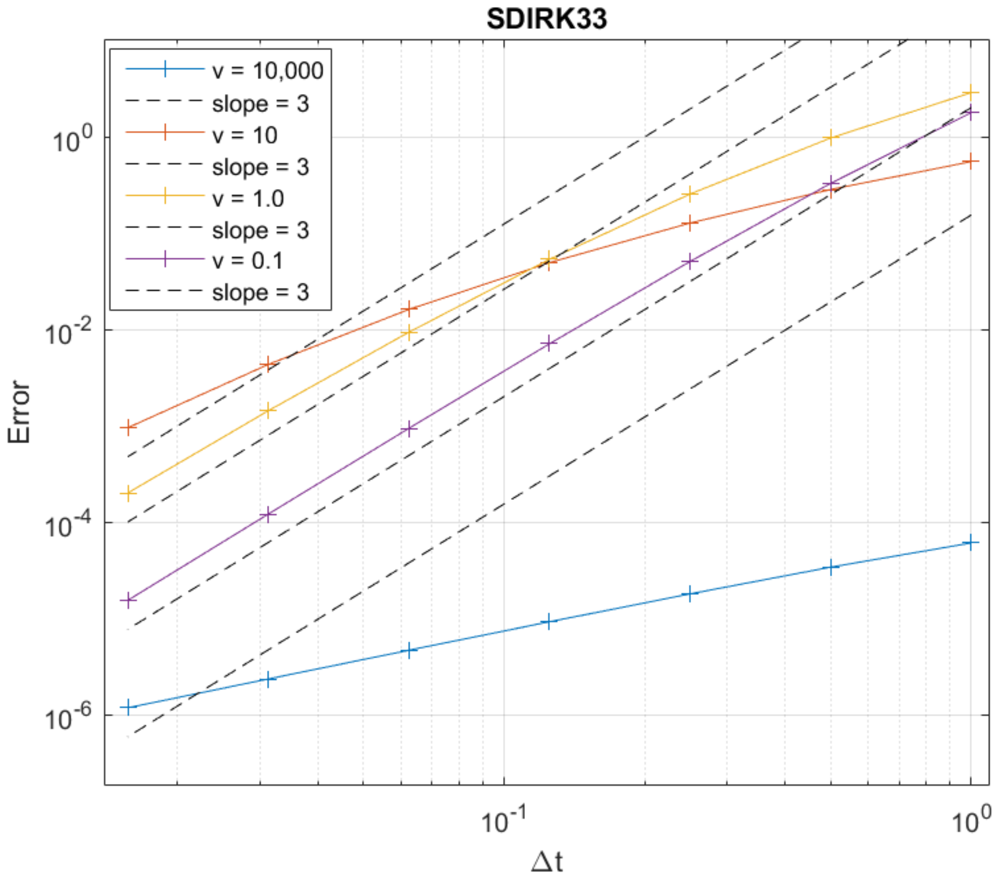


Figure B.2: SDIRK33 convergence with different velocities