

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

A Thesis

by

ZACHARY M. 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, Dr. Jean C. Ragusa
Committee Members, Dr. Jim E. Morel
Dr. Bojan Popov
Head of Department, Dr. Yassin Hassan

May 2017

Major Subject: Nuclear Engineering

Copyright 2017 Zachary M. Prince

ABSTRACT

Transient reactor simulation is 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 involves factorizing neutron flux within a reactor into a time-dependent amplitude and time-space-energy dependent shape. The purpose is to evaluate amplitude and shape on different time scales in order to reduce computational expense, while maintaining solution accuracy. IQS factorization leads to a nonlinear system of equations that requires iteration of shape and amplitude. Additionally, reactor simulation requires multiphysics simulation, where the quasi-static methodology can also be applied.

The objectives of this research is 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 formula (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 to 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

TAMU	Texas A&M University
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
LRA	Laboratorium für Reaktorregelung und Anlagensicherung
DT2	Step Doubling Time Adaptation Method

TABLE OF CONTENTS

	Page
ABSTRACT	ii
DEDICATION	iv
CONTRIBUTORS AND FUNDING SOURCES	v
NOMENCLATURE	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xii
1. INTRODUCTION	1
1.1 Background on Transient Reactor Testing	1
1.2 Time-Dependent Neutron Transport	2
1.2.1 Point Reactor Kinetics	3
1.2.2 Improved Quasi-Static Method	3
1.3 Time-Dependent Reactor Dynamics	4
1.4 Objective	5
2. Improved Quasi-Static Method	8
2.1 Theory	8
2.1.1 Operator Notation and Extension of IQS to Multigroup Transport Equations	11
2.2 Iterative Solution Techniques	12
2.2.1 Shape Convergence	15
2.2.2 Property Convergence	15
2.2.3 Solution Scaling	16
2.2.4 Preconditioned Jacobian-Free Newton-Krylov	16
2.3 Predictor-Corrector version of IQS (IQS P-C)	17
2.4 Temperature Feedback	19
2.4.1 Temperature Evaluation	19
2.4.2 Intermediate Time Scale	20

2.4.3	Dynamical Time Scale Analysis	20
2.4.4	Programming Logic	22
2.5	Time Discretization Schemes	22
2.5.1	Theta Method Time Discretization	22
2.5.2	Backward Difference Formulas	24
2.5.3	Singly-Diagonally-Implicit Runge-Kutta Method	24
2.5.4	Time Adaptation	25
2.6	Delayed Neutron Precursor Updates	28
2.6.1	Precursor Evaluation Using Theta Method Time Discretization . .	28
2.6.2	Analytical Precursor Integration	29
2.7	Rattlesnake Implementation	30
2.7.1	Executioner	30
2.7.2	Action System	31
2.7.3	PRKE coefficients	31
2.7.4	Other Action Systems	32
2.7.5	Predictor-Corrector Modification	33
2.7.6	Input	34
3.	KINETICS EXAMPLES	36
3.1	One Dimensional Prototype Problem	36
3.1.1	IQS Iteration Convergence	37
3.1.2	Time Step Convergence	39
3.2	TWIGL Benchmark	45
3.2.1	TWIGL Convergence Analysis	45
3.2.2	TWIGL with Step Doubling Time Adaptation	46
3.3	C5G7-TD Benchmark	49
3.4	LMW Benchmark?	51
4.	DYNAMICS EXAMPLES	52
4.1	LRA Benchmark	52
4.1.1	LRA Multiphysics Time Scale Results	56
4.1.2	LRA with Time Adaptation	62
4.2	TREAT Transient-15 Problem	63
4.2.1	Transient-15 Temperature Feedback	64
4.2.2	Transient-15 Multiphysics Time Scale Results	66
4.2.3	Transient-15 with Time Adaptation	68
4.3	TREAT M8-CAL Problem	70
	APPENDIX A. FIRST APPENDIX	75

APPENDIX B. A SECOND APPENDIX WHOSE TITLE IS MUCH LONGER THAN THE FIRST	76
B.1 Appendix Section	76
B.2 Second Appendix Section	76

LIST OF FIGURES

FIGURE	Page
2.1 IQS method visualization	11
2.2 Visualization of IQS fixed-point iteration process	14
2.3 Time scales and process of IQS with temperature feedback	21
2.4 Visualization of fixed-point iteration and temperature update process for IQS	23
2.5 Visualization of step doubling process on time-line	27
2.6 Visualization of step doubling process with coding logic	27
3.1 1-D slab region identification	36
3.2 Baseline power profile of 1D slab	38
3.3 Flux profile of 1D slab at various times	38
3.4 # of iterations for various convergence criteria, tolerance= 10^{-11} , max iterations= 20	40
3.5 Final iteration error for K convergence criteria	40
3.6 Final iteration error for K convergence criteria with analytical precursor elimination	41
3.7 Error convergence plots of implicit discretization, IQS, and IQS P-C with various time discretization schemes	43
3.8 Error convergence plots of implicit discretization, IQS, and IQS P-C from Rattlesnake implementation	44
3.9 Error convergence plots of implicit discretization, IQS, and IQS P-C vs. number of GMRES iterations	44
3.10 TWIGL benchmark problem description	45
3.11 Power level comparison of TWIGL Benchmark	47

3.12	TWIGL Benchmark flux/shape comparison at $t = 0.2$	47
3.13	Error convergence comparison of TWIGL Benchmark	48
3.14	Power level comparison of TWIGL Benchmark with time adaptation	49
3.15	Geometry of the two-dimensional C5G7 benchmark problem. Boundary conditions are vacuum (V) or reflective (R).	50
3.16	Geometry of a C5G7 pin-cell.	50
4.1	LRA benchmark geometry with region assignment.	55
4.2	LRA baseline temperature and power profile	57
4.3	LRA baseline spacial power profile at $t = 1.44s$	58
4.4	LRA error convergence plots	58
4.5	Error plot with various temperature updates per macro step	59
4.6	Dynamical time scale for LRA benchmark	62
4.7	LRA power profile with time adaptation of implicit discretization and IQS P-C	63
4.8	Transient 15 159-element small core configuration	64
4.9	Top quarter view of Transient 15 mesh	65
4.10	Transient 15 core power profile at peak power	65
4.11	Transient 15 total power and average temperature profile during transient	67
4.12	Dynamical time scale for the Transient 15 example	69
4.13	Transient-15 step doubling adaptation results with implicit discretization and IQS P-C	69

LIST OF TABLES

TABLE	Page
3.1 1-D slab material properties and problem parameters	37
3.2 1-D slab absorption cross-section at times of interest	37
3.3 TWIGL benchmark material properties and slope perturbation	46
3.4 TWIGL step doubling results	48
3.5 Energy group boundaries for C5G7 MOX benchmark.	51
4.1 LRA benchmark initial two-group constants.	55
4.2 LRA benchmark delayed neutron data.	55
4.3 LRA benchmark scalar values.	56
4.4 LRA baseline verification	57
4.5 Implicit discretization run time results	61
4.6 IQS run time results with $\Delta t = 0.004$	61
4.7 IQS PC run time results with $\Delta t = 0.004$	61
4.8 LRA step doubling adaptation results with implicit discretization and IQS P-C	63
4.9 Transient-15 Error and Runtime Results	66

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) [1]. Currently, two facilities are capable of testing fuels under these extreme conditions: Annular Core Research Reactor (ACRR) at Sandia National Laboratory (SNL) and the Transient Reactor Testing Facility (TREAT) at Idaho National Laboratory (INL) [2]. 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 [3].

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 [4]. 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 at-

tention 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). 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) [5]. 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 [6, 5]. Since the purpose of this research involves temporal dependence, discretization in time is kept general. 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 particularly stiff due

to the speed of neutrons, especially in a pulsing reactor [7]. 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 transient flux while maintaining accuracy.

1.2.1 Point Reactor Kinetics

A common solution process for neutron kinetics is the evaluation of the point reactor kinetics equation (PRKE) [8]. 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 spacial 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 on the initial time step. The rest of the time steps just evaluate one variable: 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 [9, 10, 11, 12, 13]. These components are the flux's amplitude and its shape. Amplitude is only time-dependent, while the shape is both space- and time-dependent. However, the impetus of the method is the assumption that the shape is only weakly dependent on time; therefore, the variable for the diffusion evaluation is less stiff without making the assumption that spatial variance is time-independent, as the PRKE does. 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.

Since shape and amplitude share an independent variable, they are temporally coupled. Ott in [9] was the first publication that investigated the coupling of shape and ampli-

tude in a quasi-static nature, but did not include the time derivative in the shape equation. Ott later in [14] incorporated the time derivative of shape in the equation, yielding better results. The resulting system of equations is heavily nonlinear [10]. Nonlinear problems require an iterative process to evaluate, either fixed-point (Picard) or Newton iteration. Sissaoui et al. [13], Koclas et al. [15], Devooght et al. [11], and Monier [12] all use fixed-point iterative techniques for their quasi-static simulations, the main difference among them is their criteria for convergence. Devooght et al. in [11] also utilizes a Newton iteration technique.

1.3 Time-Dependent Reactor Dynamics

Transient reactor analysis also involves coupling neutronics with other physics like temperature feedback, thermal hydraulics, fuel structure and composition, etc. 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 [16]. 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) [17]. BISON is the application for evaluating fuel temperature and structural properties [18]. RELAP-7 is the main safety analysis application, but focuses on variables involved with hydrodynamic analysis [19]. MAMMOTH is the over-arching ap-

plication that couples all these physics by recomputing cross-sections for Rattlesnake and source terms for BISON and RELAP-7 [20].

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 devised into three objectives:

1. IQS iteration convergence analysis
2. Time step convergence analysis for IQS with high order time discretization schemes
3. IQS application 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.

I think an outline for the introduction should be

1. reactor transient testing. Background, etc... this subsection should conclude on the overarching need of transient testing: computational methods for multiphysics reactor transients. It should also introduce the next 2 subsections, which are both about lit review, one on time-dependent neutronics, one on multiphysics solution techniques for reactor physics.
2. literature review on mostly neutronics time dependent: challenges in computational methods for time-dependent neutron balance equations. the size of the phase-space, the need for implicitness in the temporal treatment due to the stiffness, this should open up to a discussion on alternate methods, such as IQS, ... it is a lit review, so no need for equations, but rather a demonstration that you know the background and have a solid understand of that foundation to lay the ground from your OWN work. As part of this lit, some discussion of Mund's, Dulla's, Ott's, Monier's the Canadian's papers is welcome.
3. now, reactor transients also means coupled to other physics. this is where you discuss some aspect related to multiphysics. you can start with the general discussion that this is typically a nonlinear problem, that it can be solved via Picard or newton. this is maybe a segway into a discussion/introduction about MOOSE. Then you go back on IQS and add the multiphysics aspects. Ott did some work with multiphysics, I gave you the code manual of his own software where he discusses when he decided to do shape updates.

Generally speaking, the rest is real good but you need to spend some time on these 5-8 pages of introduction that set the tone and the background. Everybody needs to understand that you know a lot; that your thoughts are well organized and clear; the rest will come easy (except some results ...).

side note: we need to put the small results about sdirk being poor for stiffness problem. maybe as an appendix. I want to keep that knowledge somewhere.

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).

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

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

where,

ϕ^g	=	Scalar flux in energy group g
C_i	=	Concentration of delayed neutron precursor i
Σ_f^g	=	Fission cross section in energy group g
Σ_r^g	=	Removal cross section in energy group g
$\Sigma_s^{g' \rightarrow g}$	=	Scattering cross section from energy group g' to g
v^g	=	Neutron velocity in energy group g
χ_p^g	=	Fission spectrum of prompt neutrons
$\chi_{d,i}^g$	=	Fission spectrum of delayed neutrons from precursor i
ν^g	=	Total number of neutrons per fission
D^g	=	Diffusion coefficient in energy group g
λ_i	=	Decay constant of precursor i

$$\begin{aligned}\beta_i &= \text{Delayed neutron fraction from precursor } i \\ \beta &= \text{Total delayed neutron fraction } (\beta = \sum_{i=1}^I \beta_i)\end{aligned}$$

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 (φ). The resulting equation is the shape-diffusion equation with precursors represented by Equations (2.2a) and (2.2b).

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

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

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,

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

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 ϕ^{*g} , which can be proven to minimize truncation error [5]. The final expressions are given below:

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

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

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:

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

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

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

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: $\sum_{g=1}^G \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right) = \text{constant}$

[10].

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.

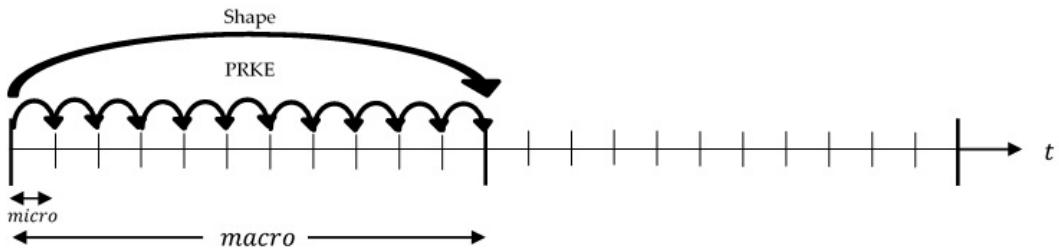


Figure 2.1: IQS method visualization

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 rewrite the neutron conservation equations in operator form, shown in Equation (2.5).

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

where Ψ^g is the multigroup neutron flux (angular flux in the case of transport), $H^{g' \rightarrow g}$ is the scattering operator, $P_p^{g' \rightarrow g}$ 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 ψ^g , leading to the following shape equations, Equation (2.6):

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

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

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

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

where $P_{d,i}^{g' \rightarrow g}$ 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 [13], Koclas et al. from [15], Devooght et al. from [11], and Monier from [12] 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 [13, 15] use fixed point iteration where the criteria is the simply the normalized difference between the last two computed shapes. Monier in [12] also does fixed point iterations with the same criteria, except the solution is scaled by $\frac{\sum_{g=1}^G (\phi^{*g}, \frac{1}{v^g} \varphi^g(t_n))}{\sum_{g=1}^G (\phi^{*g}, \frac{1}{v^g} \varphi^g(t_{n+1}))}$ after each iteration. Devooght in [11] 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 [10] 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

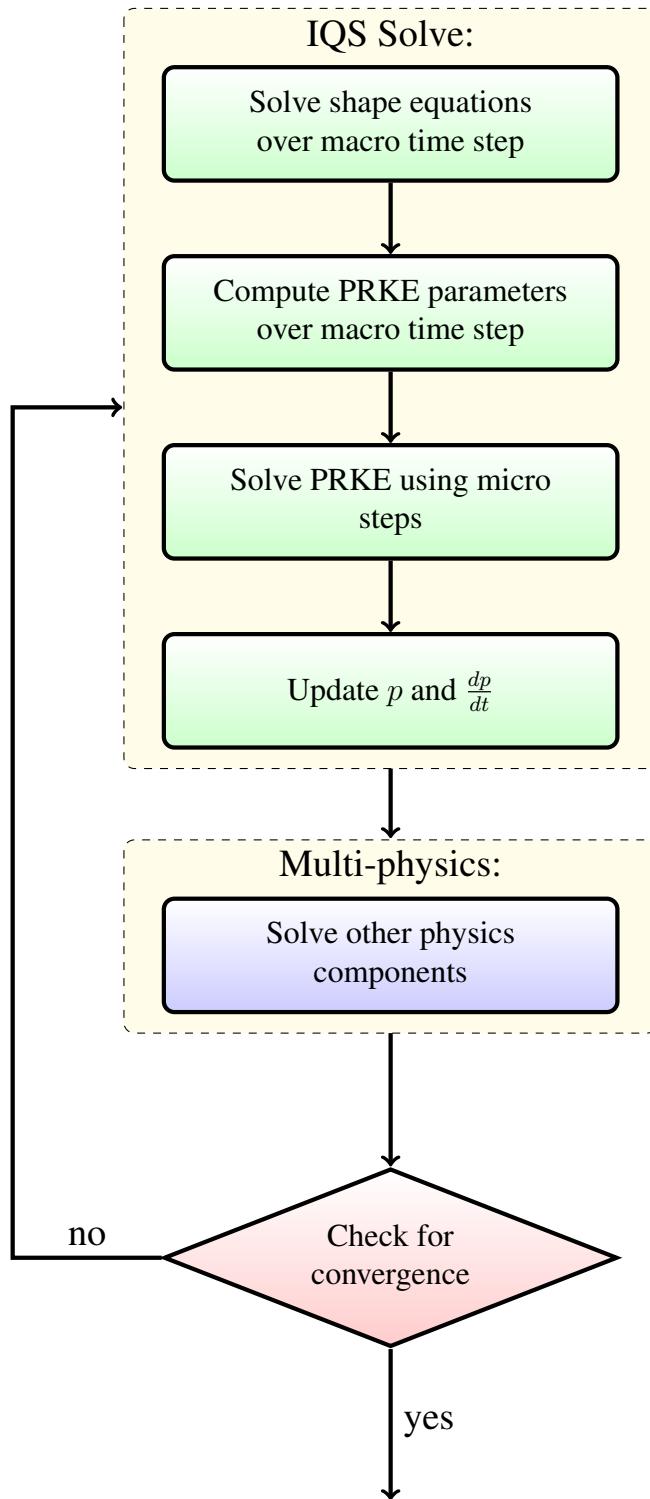


Figure 2.2: Visualization of IQS fixed-point iteration process

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 [12] observes the L^∞ 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).

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

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

Where n is the time step, k is the iteration number, and ϵ_φ is the numerical criteria provided by a user. There is no guarantee that the shape converges, so max number of iterations is typically enforced.

2.2.2 Property Convergence

Monier in [12] 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.

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

$$p_n^{k+1} - p_n^k < \epsilon_p \quad (2.12)$$

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

Where K is the IQS uniqueness expression:

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

ϵ_ρ is the reactivity convergence criteria, ϵ_p is the amplitude convergence criteria, and ϵ_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.

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

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 (GM-RES) 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 [21]. Essentially, the IQS system of equations can be described by Equation (2.16). Where A is a matrix operator, φ is the solution vector, and F is the forcing vector.

$$A(\varphi(p))\varphi = F(\varphi, p, t) \quad (2.16)$$

Applying the residual based Newton-method yields:

$$J\delta\varphi = -R(\varphi, p) \quad (2.17)$$

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:

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

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

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

$$\delta\varphi = P^{-1}w \quad (2.20)$$

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. ϵ 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 version of 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:

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

where the scaling factors are given by

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

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

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:

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

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 [22] and Goluoglu et al. in [23] 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 [10] 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. The most simple example of multi-physics 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 ρ is the material density, c_p is the specific heat, T is temperature, and κ_f is the energy released per fission [24]. The change in temperature of the material mainly affects the thermal macroscopic absorption cross section described by Equation (2.26) [24].

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

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

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).

$$T^{n+1} = T^n + \frac{\kappa_f}{\rho c_p} (a_2 \varphi^{n+1} + a_1 \varphi^n) \quad (2.27)$$

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.

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

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

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 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 (τ) for each physics. A small value τ means the variable is quickly varying in time and consequently needs relatively small time steps for accuracy, vice-versa for large τ . The general definition of τ is defined by Equation (2.30), where θ

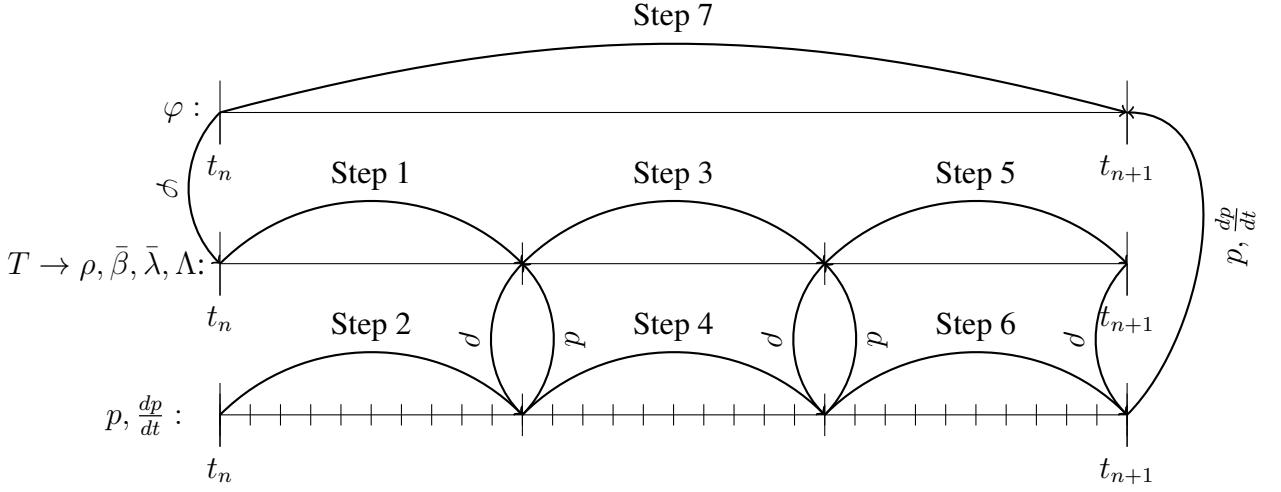


Figure 2.3: Time scales and process of IQS with temperature feedback

is the physic component of interest.

$$\tau = \frac{1}{\left| \frac{1}{\theta} \frac{d\theta}{dt} \right|} \quad (2.30)$$

Since each variable is 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, τ 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). θ represents a summation over groups for flux and shape.

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

According to the a priori hypothesis from previous sections, τ 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 $\Delta t/3$ for the temperature solve is arbitrary and is meant only to match Figure 2.3.

2.5 Time Discretization Schemes

A vital part of the verification and validation for IQS is analyzing error convergence. Since any time discretization scheme is capable of being 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).

$$IV \frac{\partial \varphi}{\partial t} = A\varphi + b \quad (2.32)$$

Where IV is the inverse velocity operator, A contains all the operations on φ 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 θ -scheme ($0 \leq \theta \leq 1$, explicit when $\theta = 0$, implicit when $\theta = 1$, and Crank-Nicholson when $\theta = 1/2$)

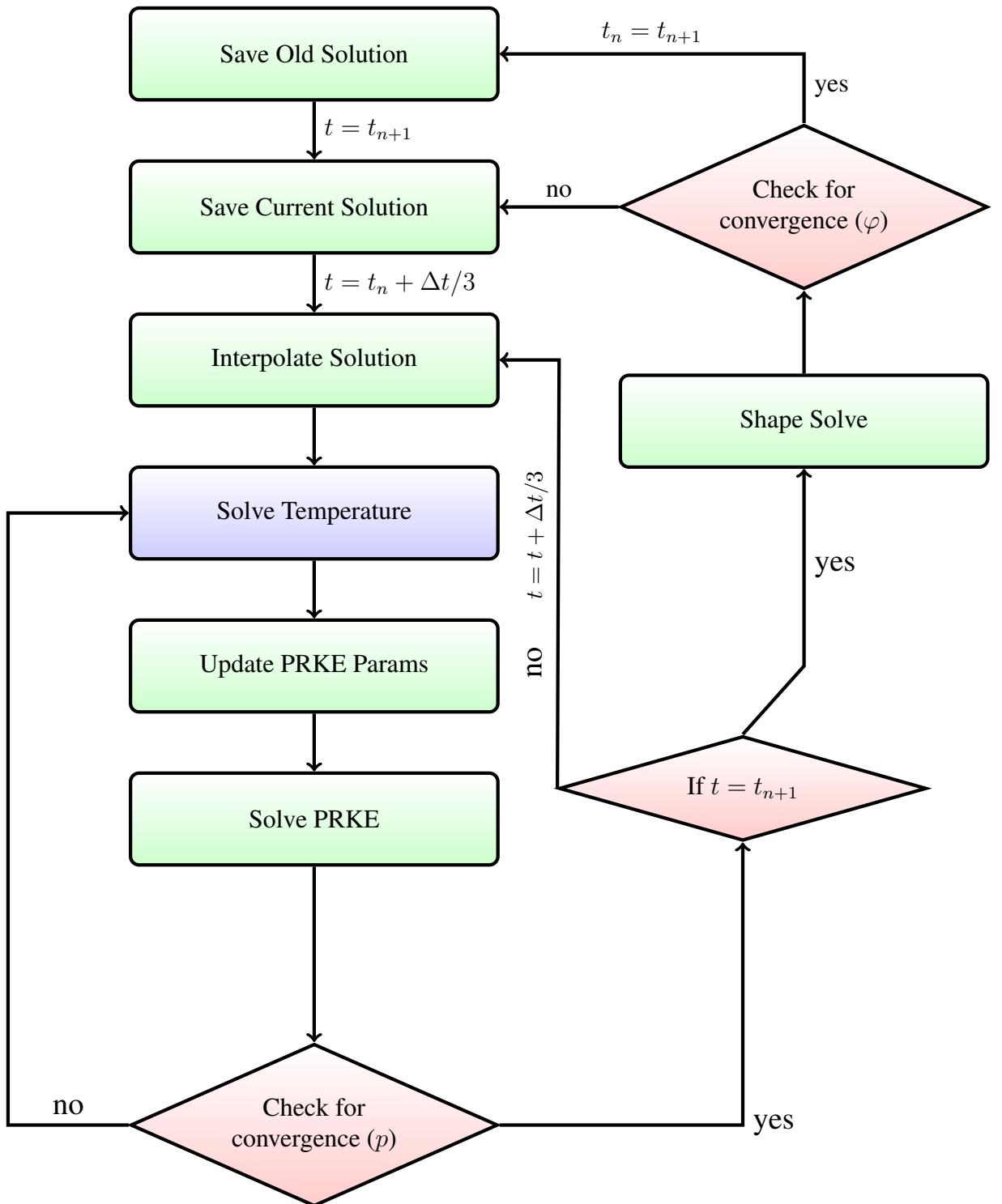


Figure 2.4: Visualization of fixed-point iteration and temperature update process for IQS

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

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

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

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

2.5.2 Backward Difference Formulas

An extension of implicit discretization is the backward difference formulas (BDF) [26]. 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).

$$\varphi_{n+1} = (IV - \Delta t A_{n+1})^{-1} \left[IV \sum_{j=0}^{k-1} \alpha_{jk} \varphi_{n-j} + \Delta t b_{n+1} \right] \quad (2.35)$$

Where k is the order of the error convergence and α_{ij} are coefficients chosen such that the temporal truncation error is minimized. The benefit of using BDF is that any order of convergence can be applied and are all unconditionally stable, 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 [27]. Generally, the method can be depicted by using a system where $dy/dt = f(t, y)$ and

Equation (2.36).

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

Where,

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

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

c_1	a_{11}			
c_2	a_{21}	a_{22}		
\vdots	\vdots	\vdots	\ddots	
c_s	a_{s1}	a_{s2}	\dots	a_{ss}
	b_1	b_2	\dots	b_s

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

λ	λ			
$\frac{1}{2}(1 + \lambda)$	$\frac{1}{2}(1 - \lambda)$		λ	
1	$\frac{1}{4}(-1 + 16\lambda - 6\lambda^2)$	$\frac{1}{4}(5 - 20\lambda + 6\lambda^2)$	λ	
	$\frac{1}{4}(-1 + 16\lambda - 6\lambda^2)$	$\frac{1}{4}(5 - 20\lambda + 6\lambda^2)$	λ	

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 [28]. 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 ($\varphi_{\Delta t}^g$) and a solution with two half steps ($\varphi_{\Delta t/2}^g$).

Note: φ is changed to ϕ for implicit discretization and IQS P-C.

The relative error is computed as follows:

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

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:

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

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.

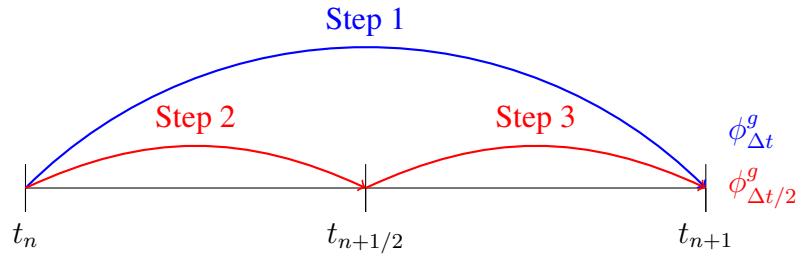


Figure 2.5: Visualization of step doubling process on time-line

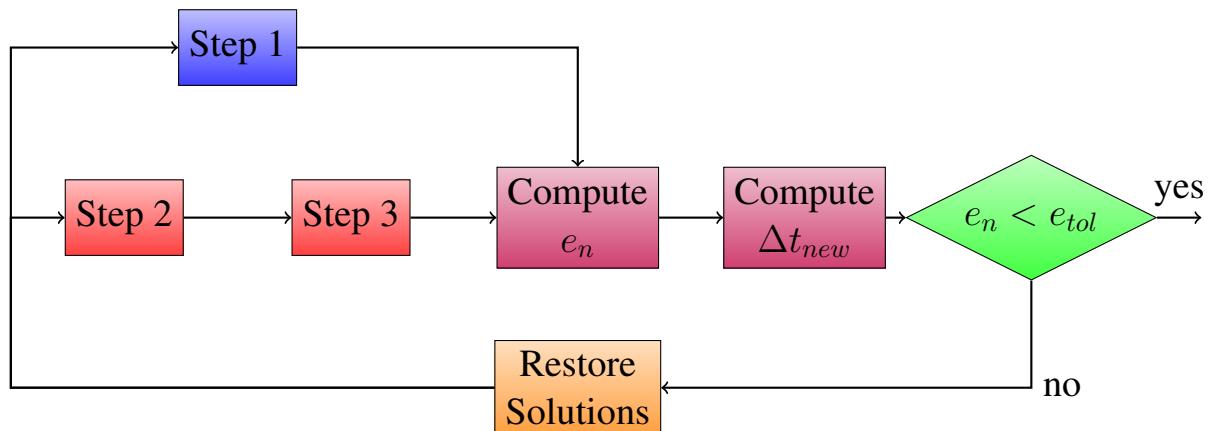


Figure 2.6: Visualization of step doubling process with coding logic

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):

$$\frac{C^{n+1} - C^n}{\Delta t} = (1 - \theta)\beta S_f^n p^n - (1 - \theta)\lambda C^n + \theta\beta S_f^{n+1} p^{n+1} - \theta\lambda C^{n+1} \quad (2.40)$$

Where S_f is the fission source equivalent for shape ($S_f^n = (\nu\Sigma_f)^n \varphi^n$). Rearranging to solve for the precursor at the end of the time step yields

$$C^{n+1} = \frac{1 - (1 - \theta)\Delta t\lambda}{1 + \theta\Delta t\lambda} C^n + \frac{(1 - \theta)\Delta t\beta}{1 + \theta\Delta t\lambda} S_f^n p^n + \frac{\theta\Delta t\beta}{1 + \theta\Delta t\lambda} S_f^{n+1} p^{n+1} \quad (2.41)$$

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

$$C^{n+1} = C^n e^{-\lambda(t_{n+1}-t_n)} + \int_{t_n}^{t_{n+1}} \beta(t') S_f(t') p(t') e^{-\lambda(t_{n+1}-t')} dt' \quad (2.42)$$

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 scheme (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:

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

Applying this to Equation (2.42) yields:

$$C^{n+1} = C^n e^{-\lambda\Delta t} + (\hat{a}_2 S_f^{n+1} + \hat{a}_1 S_f^n) \beta \quad (2.44)$$

With integration coefficients defined as:

$$\hat{a}_1 = \int_{t_n}^{t_{n+1}} \frac{t_{n+1} - t'}{\Delta t} p(t') e^{-\lambda(t_{n+1}-t')} dt' \quad (2.45)$$

$$\hat{a}_2 = \int_{t_n}^{t_{n+1}} \frac{t' - t_n}{\Delta t} p(t') e^{-\lambda(t_{n+1}-t')} dt' \quad (2.46)$$

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:

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

2.7.2 Action System

The IQS implementation mostly require 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.

$$\frac{\partial}{\partial t} \left(\frac{\varphi^g}{v^g} \right) = \underbrace{\frac{\chi_p^g}{k_{eff}} \sum_{g'=1}^G (1 - \beta) \nu^{g'} \sum_f^{g'} \varphi^{g'}}_{FluxKernel} + \underbrace{\sum_{g' \neq g}^G \sum_s^{g' \rightarrow g} \varphi^{g'}}_{FluxKernel} - \underbrace{(-\nabla \cdot D^g \nabla)}_{FluxKernel} \varphi^g - \underbrace{\sum_r^g \varphi^g}_{FluxKernel} \\ - \underbrace{\frac{1}{v^g} \left[\frac{1}{p} \frac{dp}{dt} \right]}_{IQSKernel} \varphi^g + \underbrace{\frac{1}{p} \sum_{i=1}^I \chi_{d,i}^g \lambda_i C_i}_{ModifiedFluxKernel} \quad (2.48)$$

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 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.

IQS will be made available to any action system capable of transient simulation, which includes:

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

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 post-processors 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.

$$\underbrace{\phi_{n+1}^g}_{\text{corrected}} = \underbrace{\phi_{n+1}^g}_{\text{predicted}} \frac{K_0}{K_{n+1}} p_{n+1} \quad (2.49)$$

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 ϕ 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
[...]
[]

```

3. KINETICS EXAMPLES

Neutron dynamics 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 four are from the Argonne National Lab (ANL) Benchmark Problem Book (BPB), and are common problems for testing codes and developing methods [24].

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 Σ_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

Figure 3.2 shows the resulting baseline relative power profile of the one-dimensional

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

$D(cm)$	$\Sigma_a(cm^{-1})$	$\nu\Sigma_f(cm^{-1})$	$v(cm/s)$	β	$\lambda(s^{-1})$
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

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

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}). This baseline computation is used to compute the error of the other time discretization methods. Figure 3.3 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.

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)

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

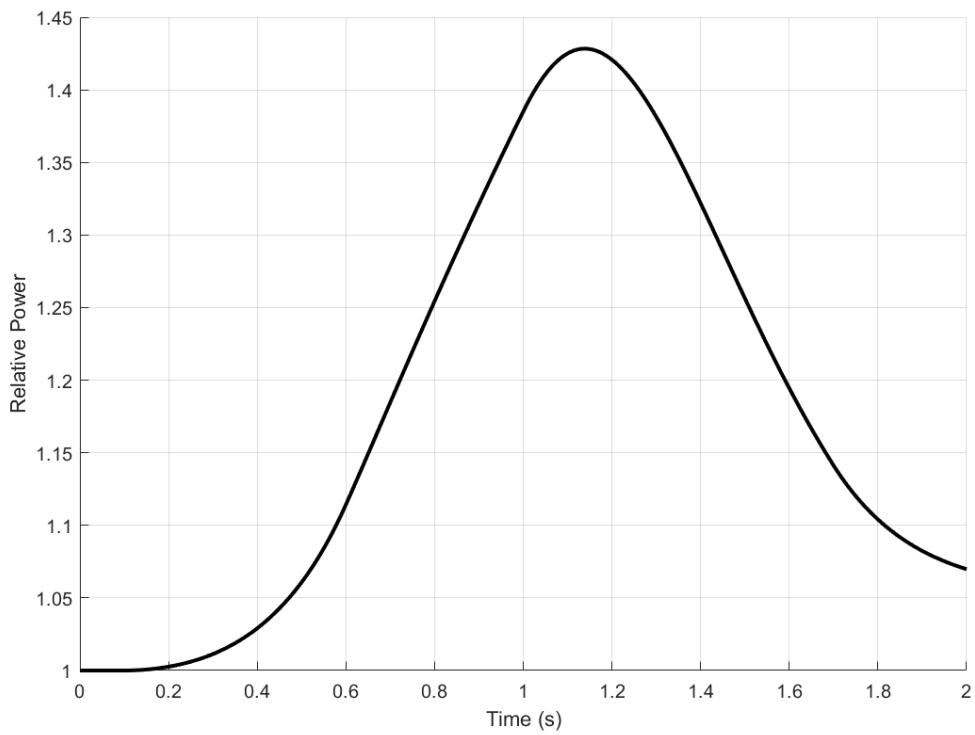


Figure 3.2: Baseline power profile of 1D slab

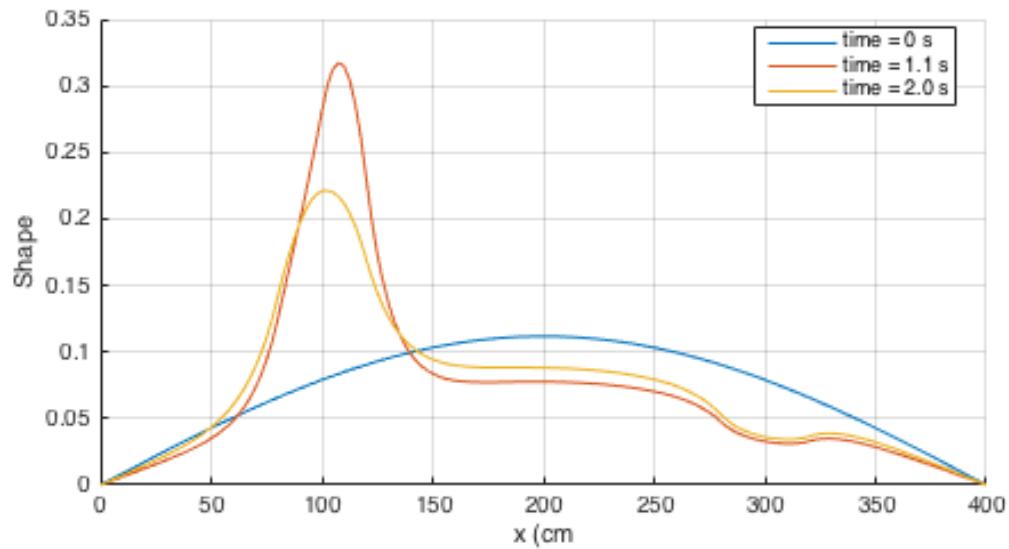


Figure 3.3: Flux profile of 1D slab at various times

3. Reactivity → Equation (2.11)
4. Amplitude → Equation (2.12)
5. K criteria → Equation (2.13)
6. All properties → 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

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 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

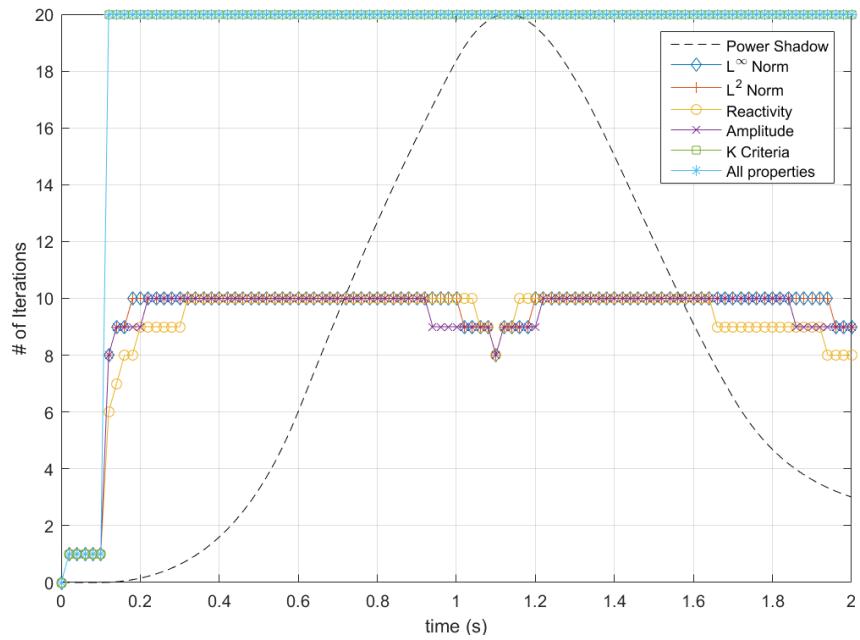


Figure 3.4: # of iterations for various convergence criteria, tolerance= 10^{-11} , max iterations= 20

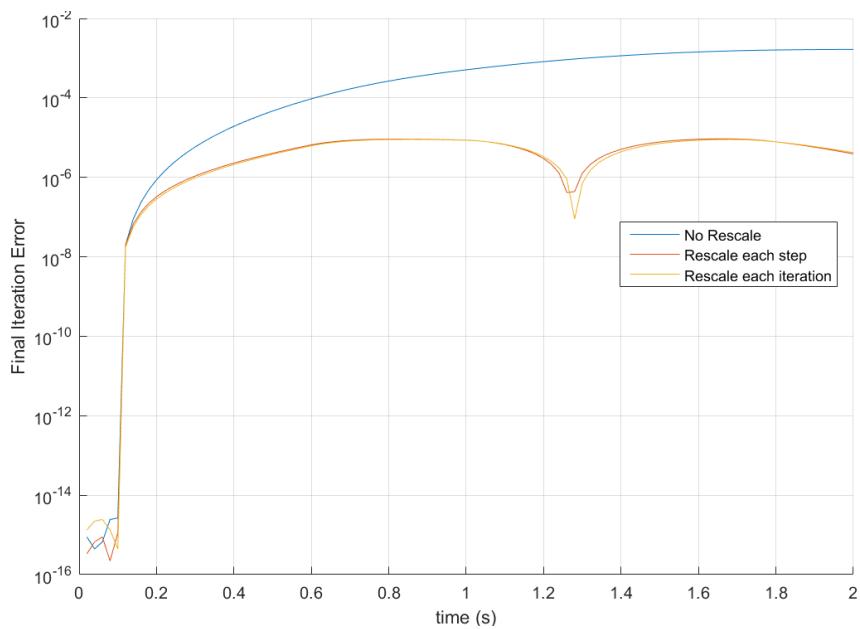


Figure 3.5: Final iteration error for K convergence criteria

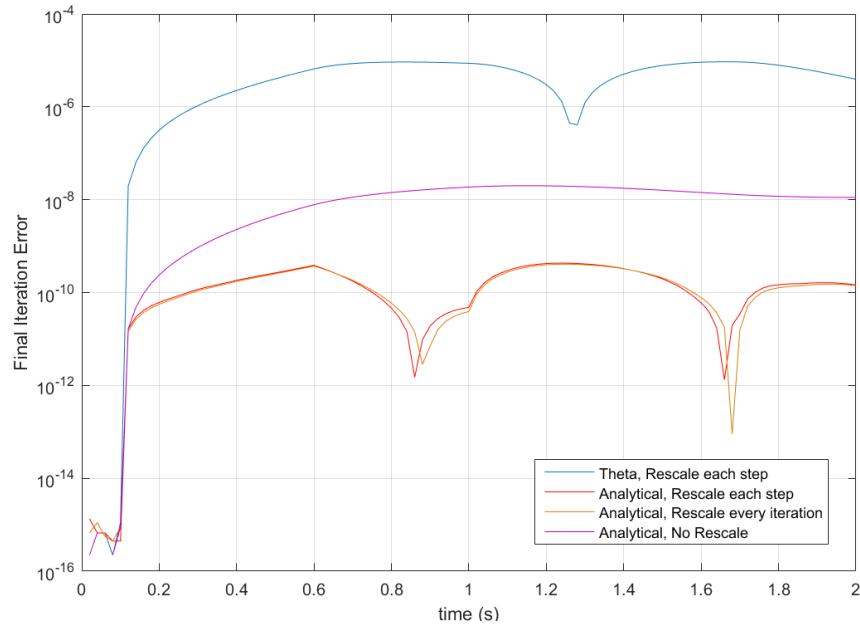


Figure 3.6: Final iteration error for K convergence criteria with analytical precursor elimination

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, 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.

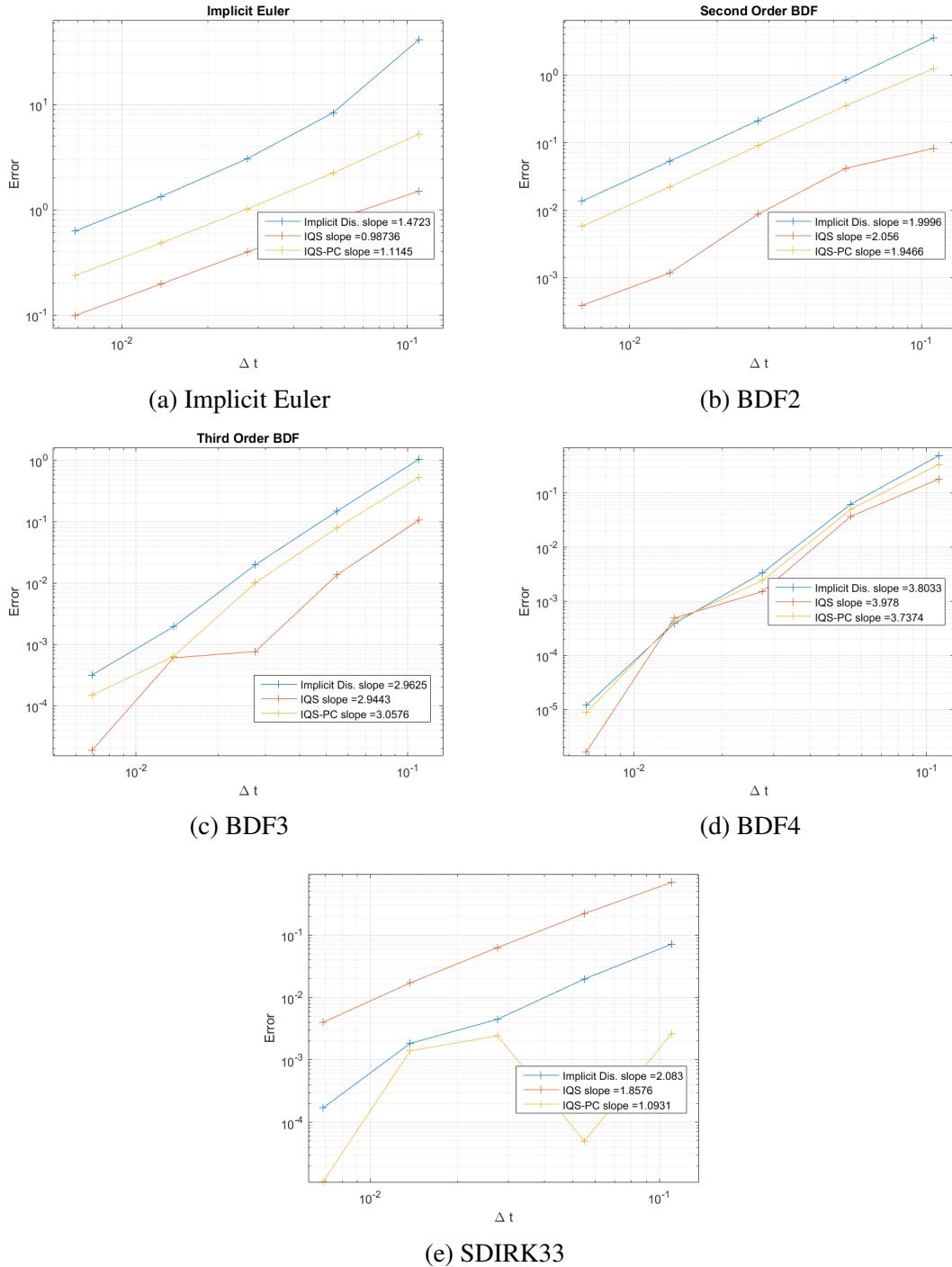


Figure 3.7: Error convergence plots of implicit discretization, IQS, and IQS P-C with various time discretization schemes

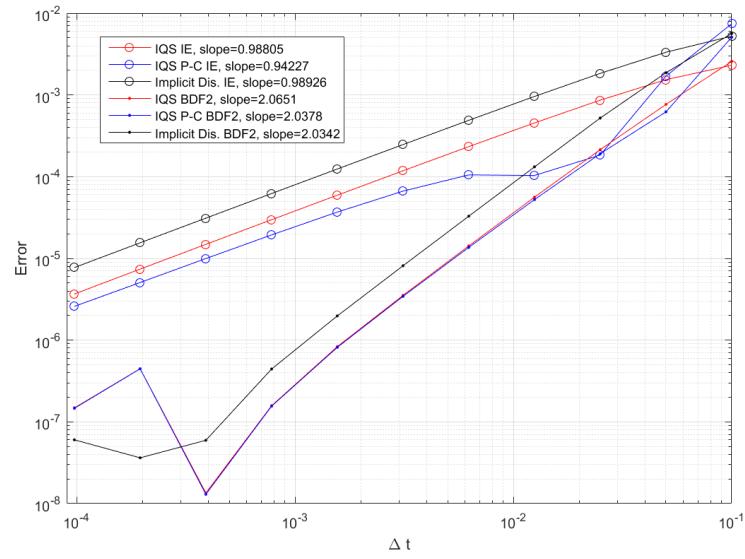


Figure 3.8: Error convergence plots of implicit discretization, IQS, and IQS P-C from Rattlesnake implementation

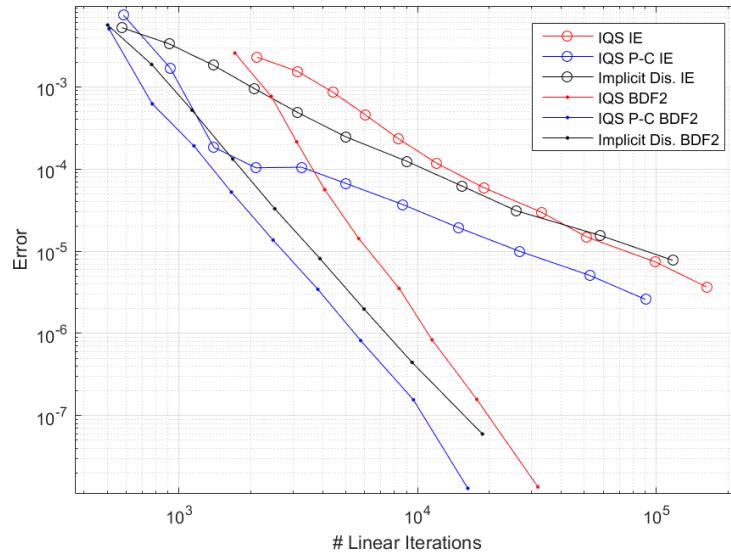


Figure 3.9: Error convergence plots of implicit discretization, IQS, and IQS P-C vs. number of GMRES iterations

3.2 TWIGL Benchmark

This benchmark problem originates from the Argonne National Lab Benchmark Problem Book [24]. It is a 2D, 2-group reactor core model with no reflector region shown in Figure 3.10 [7]. 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.

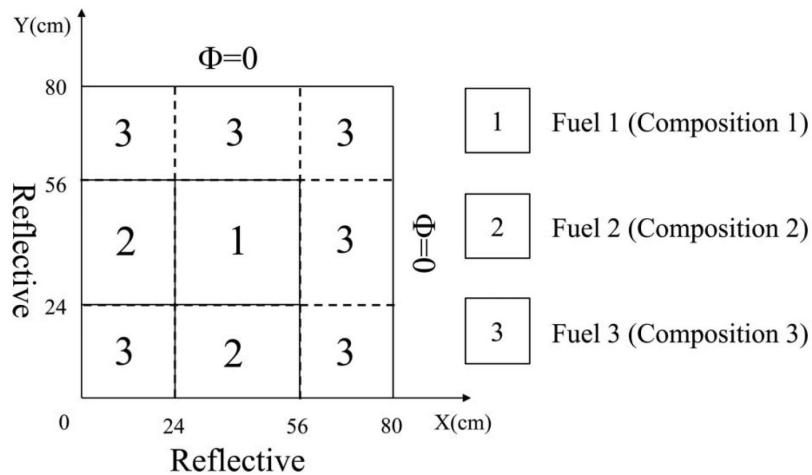


Figure 3.10: TWIGL benchmark problem description

3.2.1 TWIGL Convergence Analysis

Figs. 3.11 and 3.12 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 Brute Force flux plot (Figure 3.12) because the amplitude term is not included, but the gradients

Table 3.3: TWIGL benchmark material properties and slope perturbation

Material	Group	$D(cm)$	$\Sigma_a(cm^{-1})$	$\nu\Sigma_f(cm^{-1})$	χ	$\Sigma_s(cm^{-1})$	
						$g \rightarrow 1$	$g \rightarrow 2$
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
ν		$v_1(cm/s)$	$v_2(cm/s)$	β	$\lambda(1/s)$		
2.43	1.0E7	2.0E5		0.0075	0.08		

Material 1 ramp perturbation:

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

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

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 3.13 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.

3.2.2 TWIGL with Step Doubling Time Adaptation

Table 3.4 and Figure 3.14 show the results for TWIGL with time adaptation. The results show that both IQS methods perform exceptionally well compared to implicit dis-

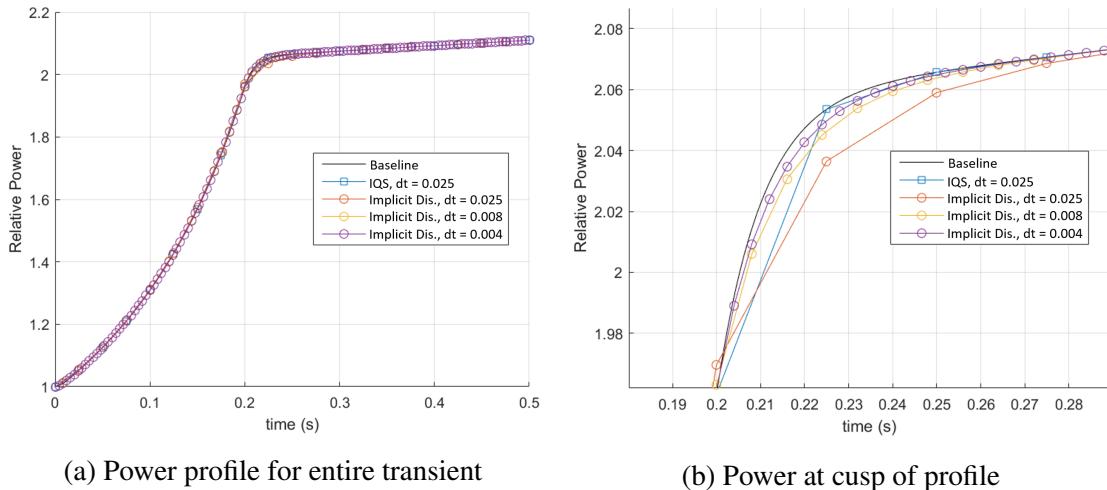


Figure 3.11: Power level comparison of TWIGL Benchmark

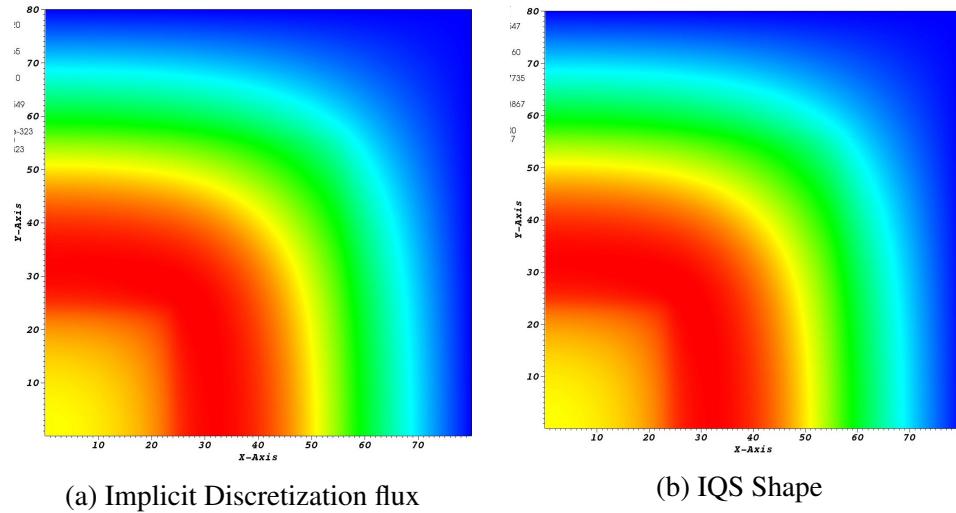


Figure 3.12: TWIGL Benchmark flux/shape comparison at $t = 0.2$

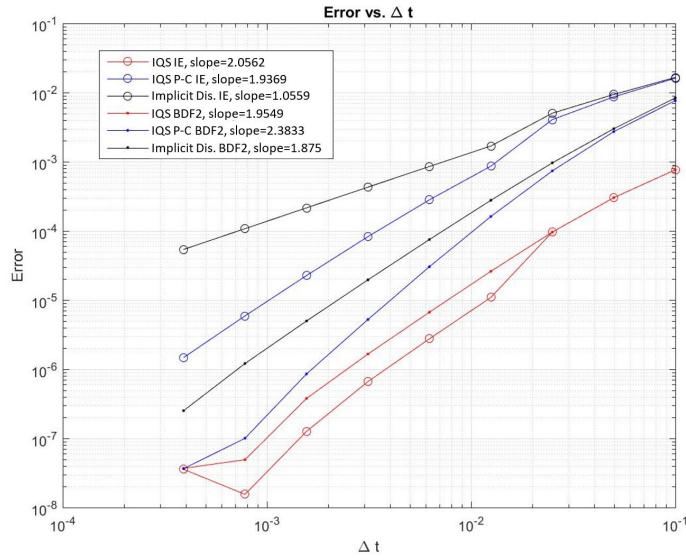


Figure 3.13: Error convergence comparison of TWIGL Benchmark

cretization. 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

Test	e_{tol}	Implicit Discretization			IQS			IQS P-C		
		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

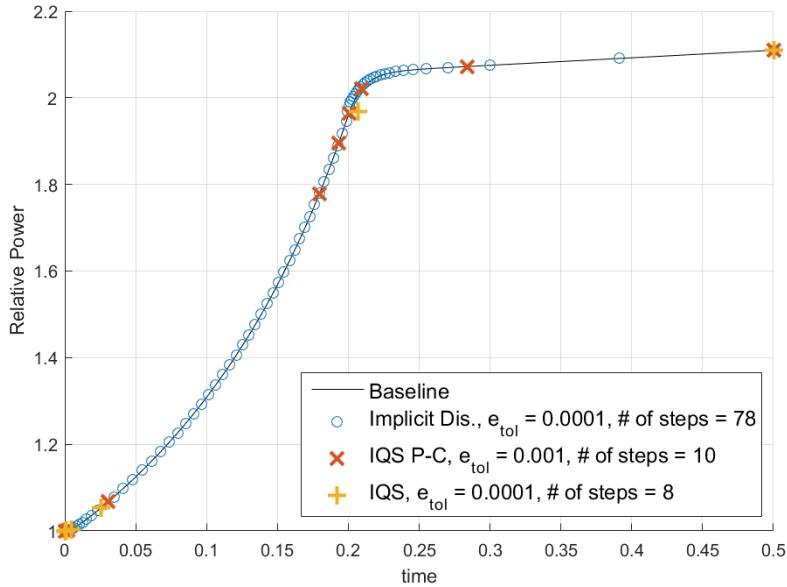


Figure 3.14: Power level comparison of TWIGL Benchmark with time adaptation

3.3 C5G7-TD Benchmark

The C5G7 benchmark is a MOX fueled, pressurized water reactor (PWR) minicore configuration. The C5G7 geometry depicted in Fig. 3.15 comprises a 2-by-2 array of UO₂ and MOX assemblies surrounded by moderator. In Fig. 3.15 vacuum and reflective boundary conditions are denoted by V and R, respectively. The fuel pins are not homogenized, each pin cell is comprised of a fuel pin, fission chamber, or guide tube surrounded by moderator as depicted in Fig. 3.16. The cladding and gap are homogenized into the fuel and are not explicitly modeled. Each assembly is made up of a regular 17-by-17 grid of pin cells. Seven energy-group cross sections for the seven material regions are given in [?]. The energy group boundaries are provided in Table 3.5; three of the seven energy groups are fast, four are thermal.

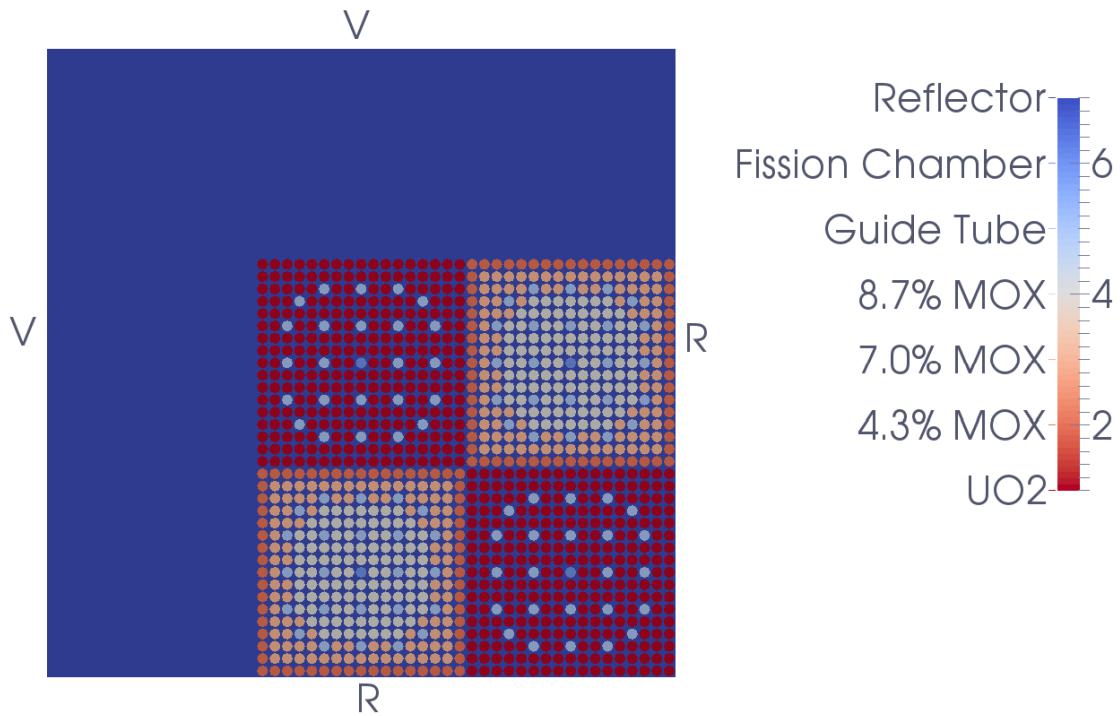


Figure 3.15: Geometry of the two-dimensional C5G7 benchmark problem. Boundary conditions are vacuum (V) or reflective (R).

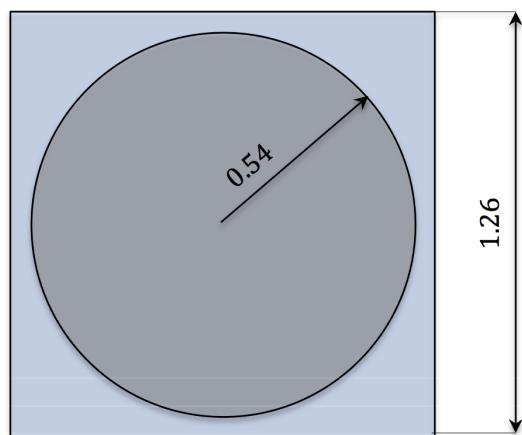


Figure 3.16: Geometry of a C5G7 pin-cell.

Table 3.5: Energy group boundaries for C5G7 MOX benchmark.

Group	Upper Energy
1	20 MeV
2	1 MeV
3	500 keV
4	3 eV
5	0.625 MeV
6	0.1 MeV
71	0.02 MeV

3.4 LMW Benchmark?

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 several dynamics examples, including the LRA benchmark and two TREAT experiments. These examples are of increased complexity from the kinetics examples of Section ???. This section 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 [24]. It is a super prompt-critical transient. To have better understanding on the cross sections given later, we present

the equations here:

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

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

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

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

$$\frac{\partial T}{\partial t} = \alpha f, \quad (4.1e)$$

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

$$P = \kappa S_f, \quad (4.1g)$$

where ϕ_1, ϕ_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; ν is the averaged number of neutrons emitted per fission; β_1, β_2 are the delayed neutron precursor fractions and $\beta = \beta_1 + \beta_2$; C_1, C_2 are the delayed neutron precursor concentrations; λ_1, λ_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; κ is the averaged power released per fission; α is the combination of κ and the specific heat capacity; γ is the Doppler feedback coefficient; $T_0 = T(\vec{r}, t = 0)$. 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

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

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

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

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

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.

Initial two-group constants are presented in Table 4.1. ν 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:

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

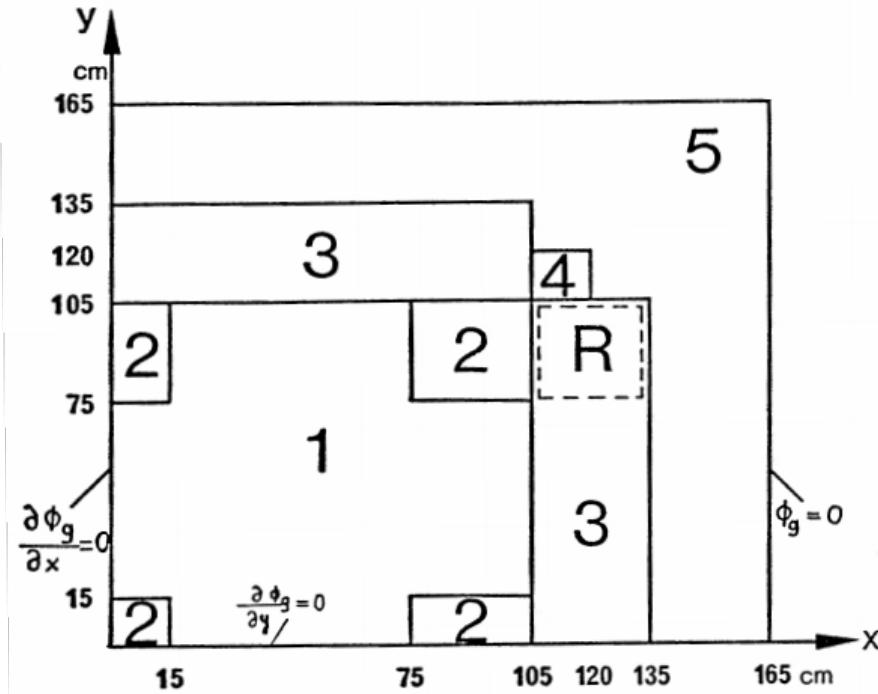


Figure 4.1: LRA benchmark geometry with region assignment.

Table 4.1: LRA benchmark initial two-group constants.

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

Table 4.2: LRA benchmark delayed neutron data.

Group i	β_i	λ_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	$10^{-4} \text{ (cm}^{-2}\text{)}$
Mean number of neutrons per fission	ν	2.43
Conversion factor	α	$3.83 \times 10^{-11} \text{ (K} \cdot \text{cm}^3\text{)}$
Feedback constant	γ	$3.034 \times 10^{-3} \text{ (K}^{1/2}\text{)}$
Energy released per fission	κ	$3.204 \times 10^{-11} \text{ (J/fission)}$
Initial and reference temperature	T_0	300 (K)
Active core volume	V_{core}	17550 (cm^2)

where t is time in seconds.

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 [29] 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.

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

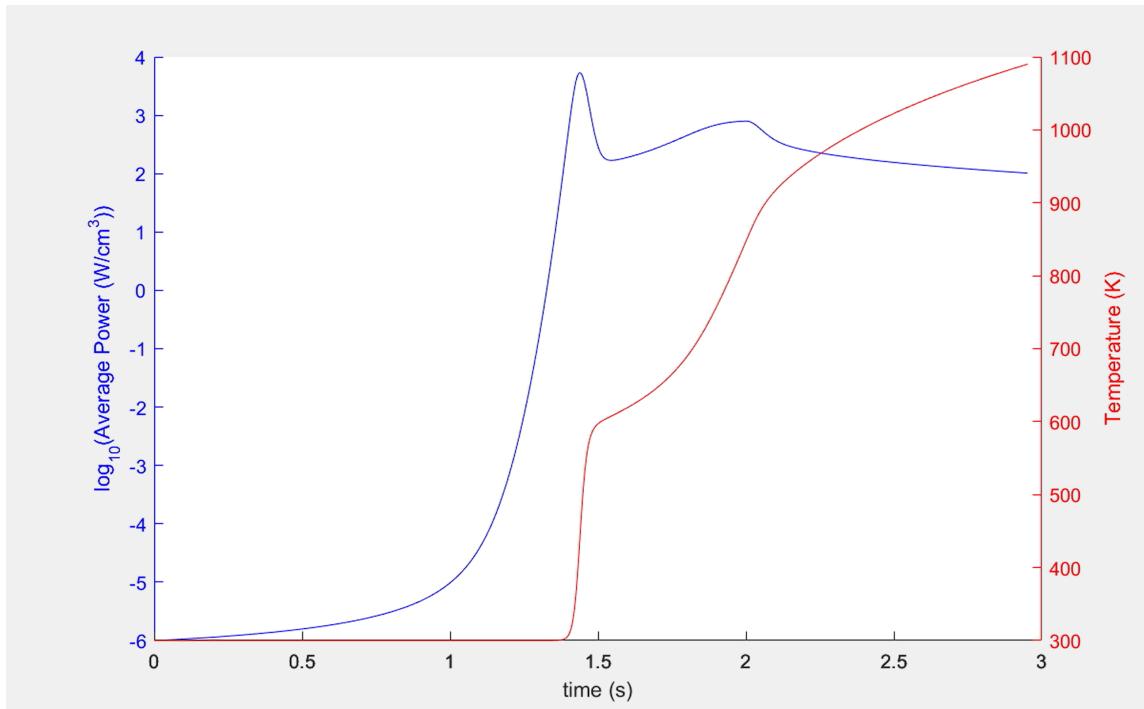


Figure 4.2: LRA baseline temperature and power profile

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

Table 4.4: LRA baseline verification

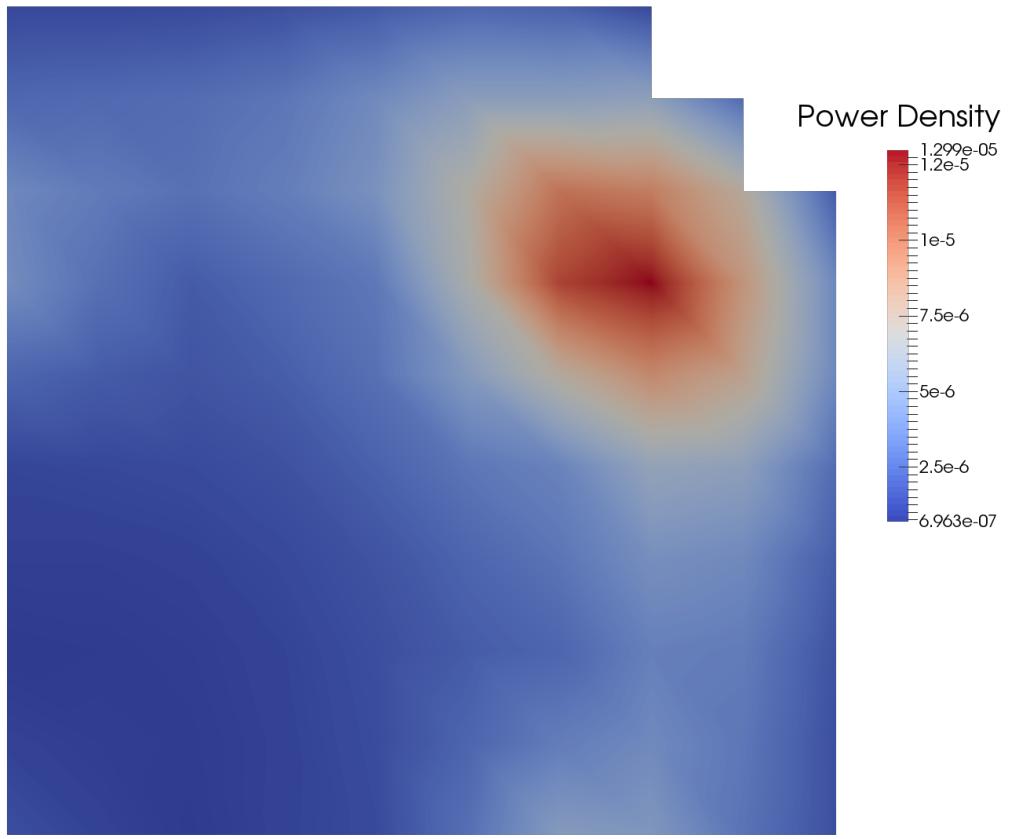
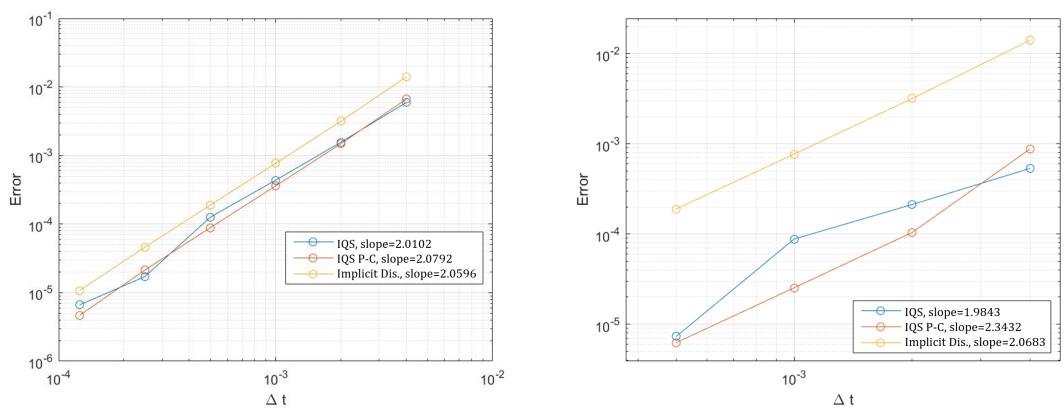


Figure 4.3: LRA baseline spacial power profile at $t = 1.44s$



(a) Only one temperature update per macro step (b) Five temperature updates per macro step

Figure 4.4: LRA error convergence plots

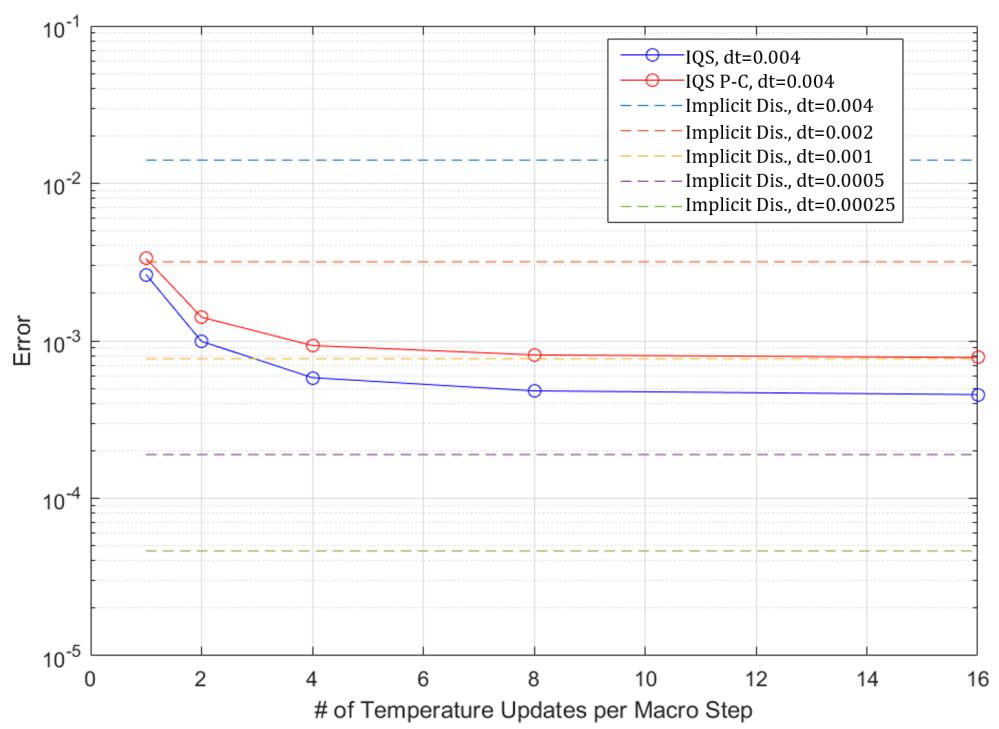


Figure 4.5: Error plot with various temperature updates per macro step

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 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%.

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

Run	Δt	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 4.5: Implicit discretization run time results

Run	Temperature Updates	Error	Runtime (hr)	% Increase 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 4.6: IQS 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

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

than temperature throughout most of the transient. The large τ 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.

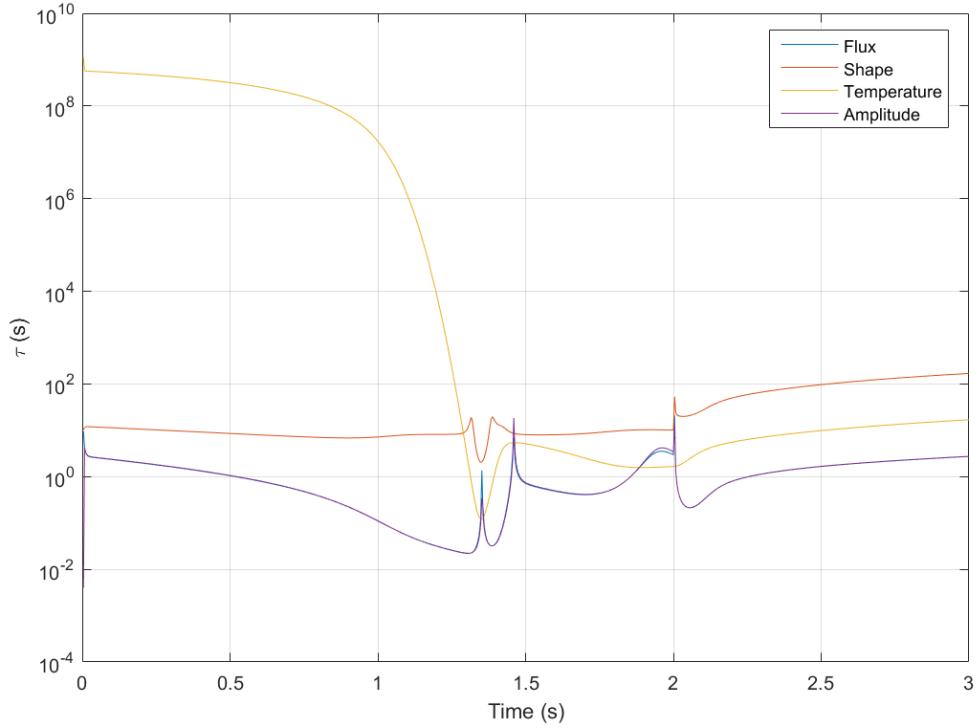


Figure 4.6: Dynamical time scale for LRA benchmark

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 obtained 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.

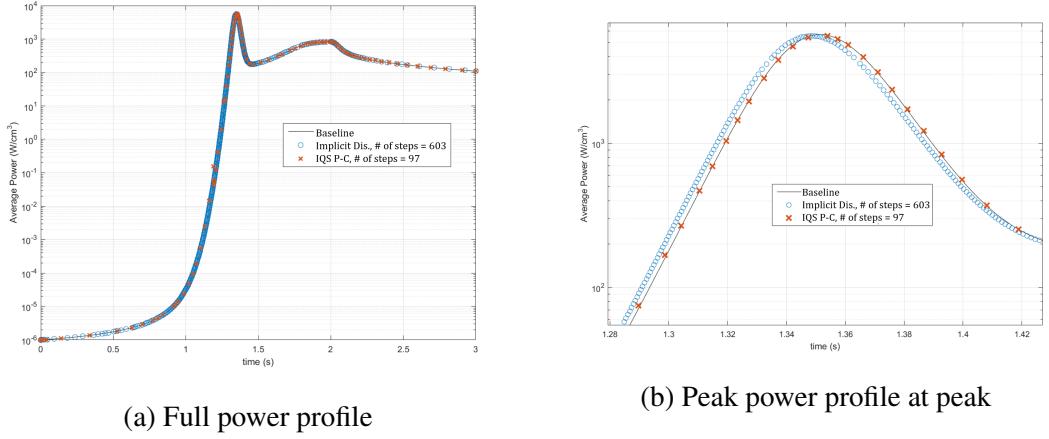


Figure 4.7: LRA power profile with time adaptation of implicit discretization and IQS P-C

Event	Implicit Dis.			IQS P-C		
	Power (W/cm³)	Error	Steps	Power (W/cm³)	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

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

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 [20]. 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 [30]. Figure 4.9 shows the meshing techniques of the model [20]. 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 4.10 shows a visualization of the flux profile within the core, hidden is the massive amount of graphite surrounding the core.

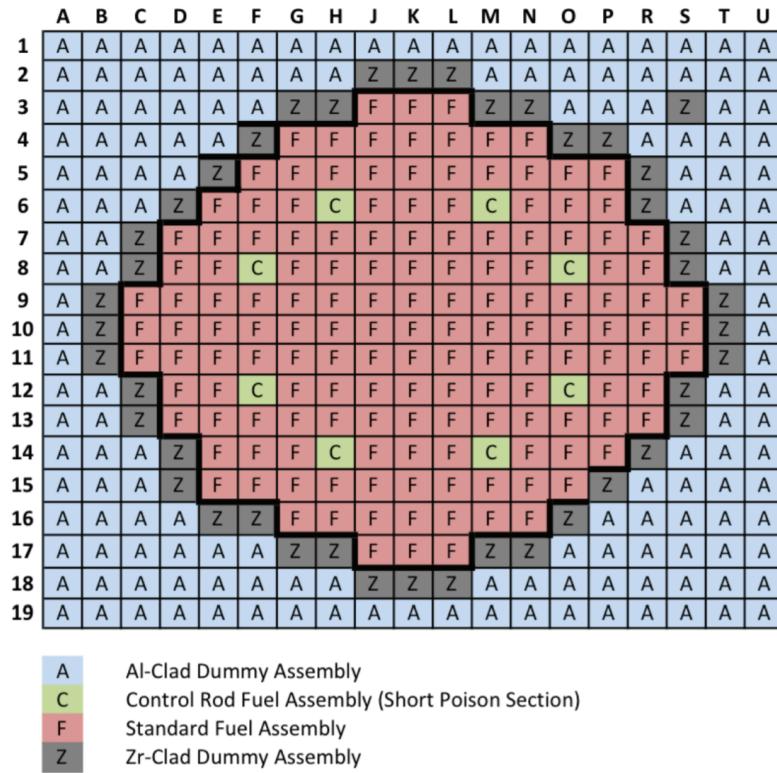


Figure 4.8: Transient 15 159-element small core configuration

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

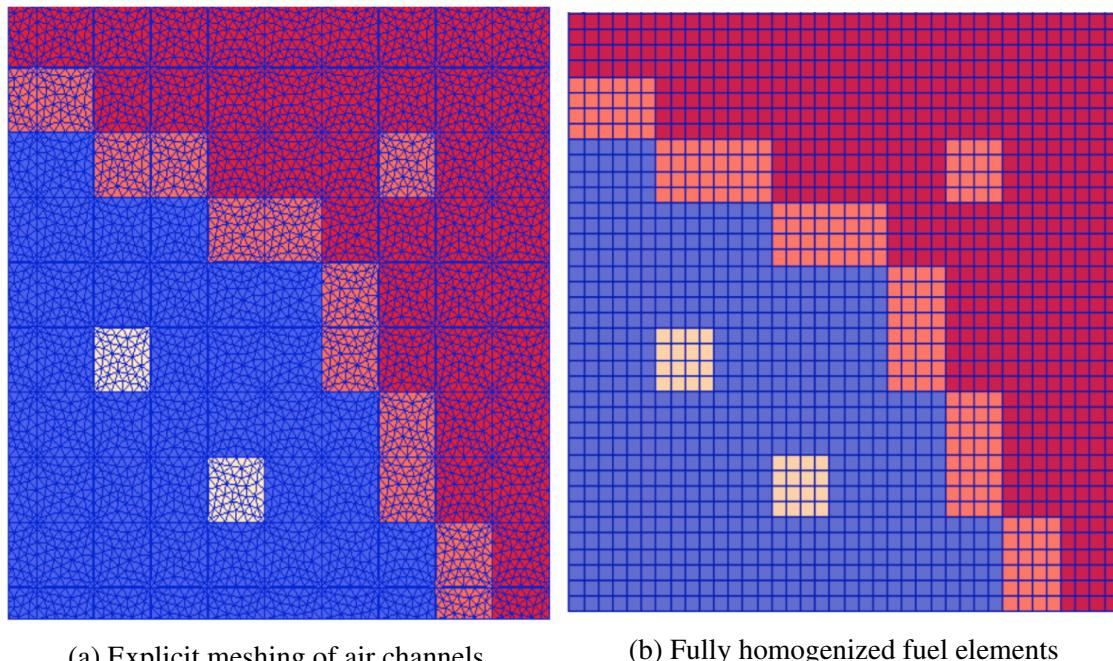


Figure 4.9: Top quarter view of Transient 15 mesh

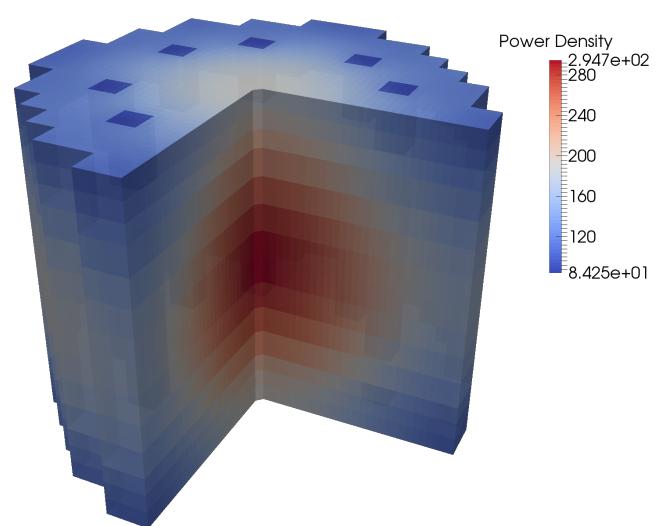


Figure 4.10: Transient 15 core power profile at peak power

similar, except the specific heat is now dependent on temperature 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.

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

$$c_p = -5.8219e-10T^3 - 4.3694e-7T^2 + 2.8369e-3T - 1.009e-2 \text{ (J/g/K)} \quad (4.7)$$

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 4.11 shows the baseline power and temperature profile for the Transient-15 example. Table 4.9 shows the 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

Table 4.9: Transient-15 Error and Runtime Results

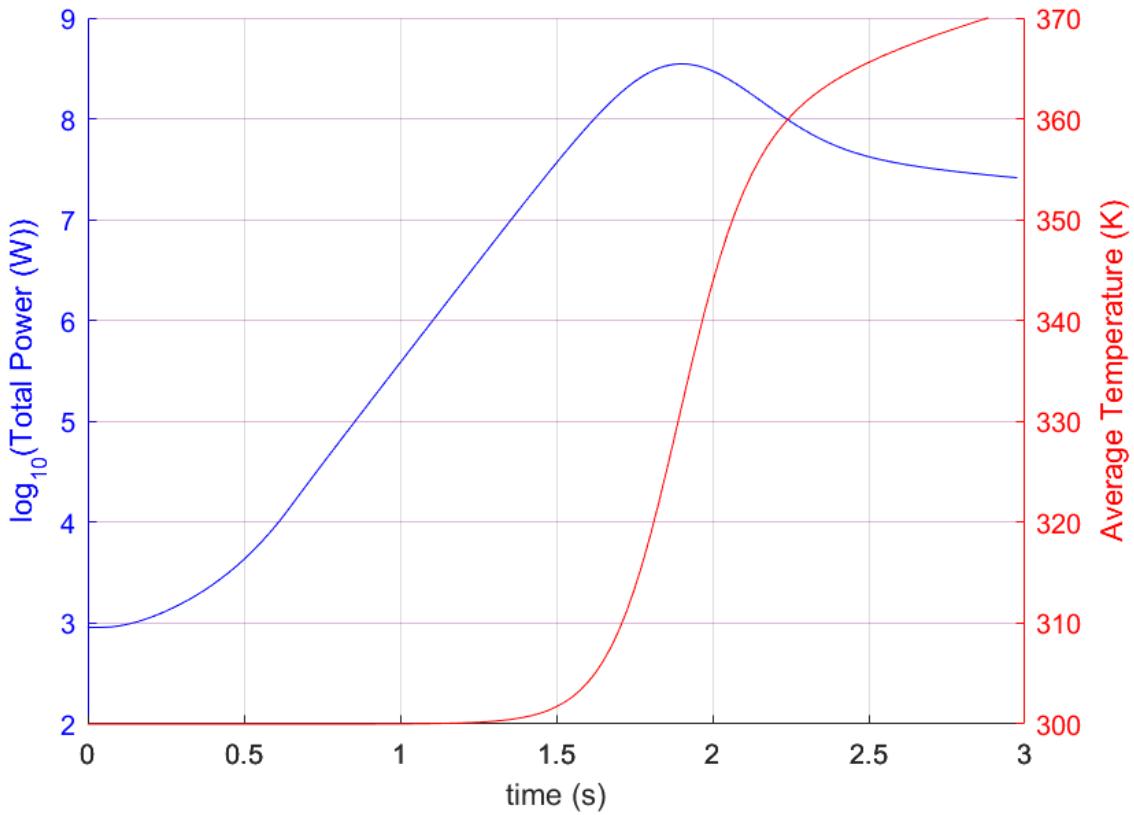


Figure 4.11: Transient 15 total power and average temperature profile during transient

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 measure 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 4.12. 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.

4.2.3 Transient-15 with Time Adaptation

Figure 4.13 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.

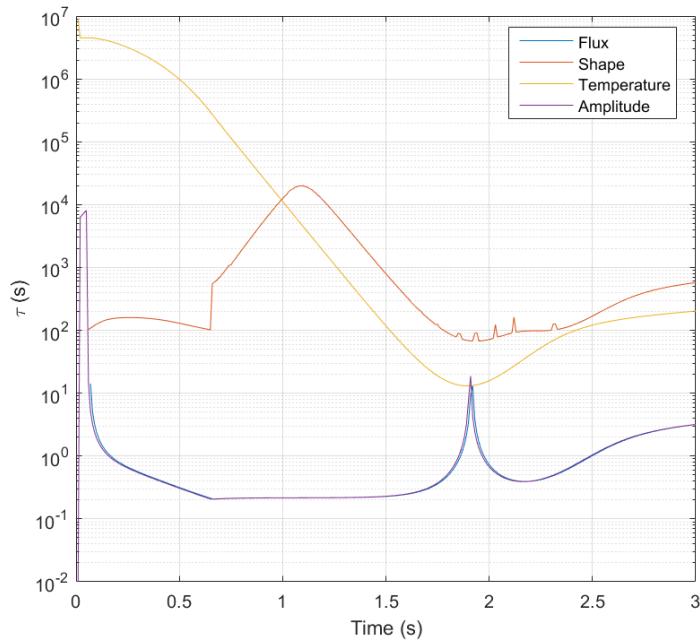


Figure 4.12: Dynamical time scale for the Transient 15 example

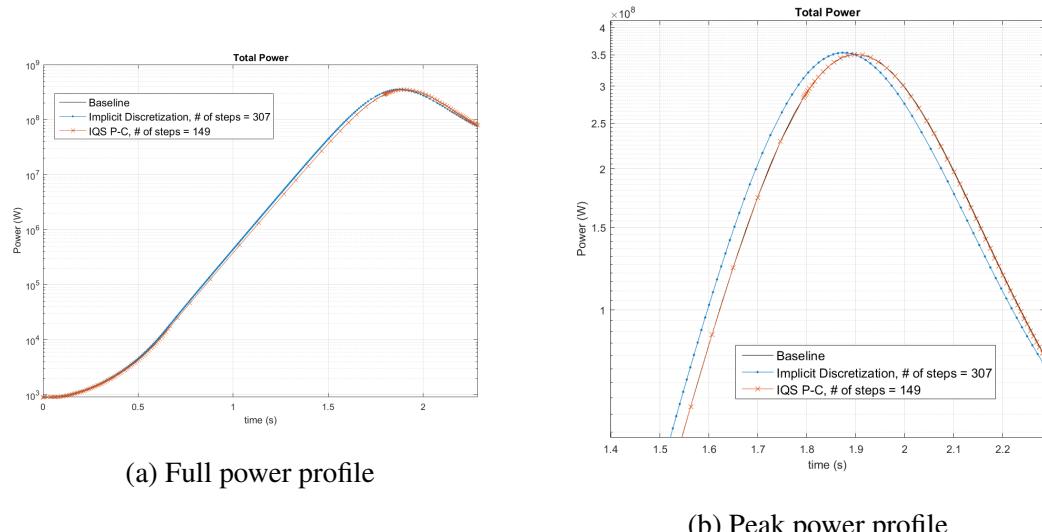


Figure 4.13: Transient-15 step doubling adaptation results with implicit discretization and IQS P-C

4.3 TREAT M8-CAL Problem

REFERENCES

- [1] “Development of light water reactor fuels with enhanced accident tolerance,” tech. rep., United States Department of Energy, Washington, DC 20585, 04 2015.
- [2] W. e. a. Parma, Vernon, “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.
- [3] “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.
- [4] e. a. Fruend, Elias, “Design summary report on the transient reactor test facility (treat), anl-6034,” tech. rep., Agonne National Laboratory, Argonne, IL, 09 1960.
- [5] J. J. Duderstadt and L. J. Hamilton, *Nuclear reactor analysis*. Wiley, 1976.
- [6] O. Zeinkiewicz, R. Taylor, and J. Zhu, *The finite element method: its basis and fundamentals*. SElsevier Butterworth-Heinemann, 2005.
- [7] 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.
- [8] J. Planchard, “On the point-reactor kinetics approximation,” *Progress in Nuclear Energy*, vol. 26, pp. 207–216, 1991.

- [9] K. Ott, “Quasi-static treatment of spatial phenomena in reactor dynamics,” *Nuclear Science and Engineering*, vol. 26, p. 563, 1966.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] D. M. K. Ott, “Accuracy of the quasistatic treatment of spatial reactor kinetics,” *Nuclear Science and Engineering*, vol. 36, pp. 381–419, 1969.
- [15] J. Koclas, M. Sissnoui, 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. 23, pp. 901–907, 1996.
- [16] 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.
- [17] Y. Wang, “Nonlinear diffusion acceleration for multigroup transport equation discretized with sn and continuous fem with rattlesnake,” in *Proc. International Con-*

ference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering, Idaho, 2013.

- [18] 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.
- [19] H. Zhang, H. Zhao, F. N. Gleicher, M. D. DeHart, L. Zou, D. Andrs, and R. C. Martineau, *RELAP-7 Development Updates*. Sep 2015.
- [20] F. N. G. Y. W. A. L. A. J. Ortensi, M. D. DeHart 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.
- [21] D. E. K. D. A. Knoll, “Jacobian-free newton-krylov methods: a survey of approaches and applications,” *Journal fo Computational Physics*, vol. 193, pp. 357–397, 2004.
- [22] 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.
- [23] S. Goluoglu and H. L. Dodds, “A time-dependent, three-dimensional neutron transport methodology,” *Nuclear Science and Engineering*, vol. 139, pp. 248–261, 2001.
- [24] Argonne Code Center, “Benchmark problem book, anl- 7416, suppl. 2,” tech. rep., Argonne National Laboratory, 1977.
- [25] J. H. Ferziger, *Numerical Methods for Engineering Application*. John Wiley and Sons, Inc., New York, NY, 1998.
- [26] B. Gear, “Backward differentiation formulas,” vol. 2, no. 8, p. 3162, 2007. revision 91024.

- [27] L. R. J. M. Franco, I Gómez, “Sdirk methods for stiff odes with oscillating solutions,” *Journal fo Computational and Applied Mathematics*, vol. 81, pp. 197–209, 1997.
- [28] W. T. V. B. P. F. William H. Press, Saul A. Teukolsky, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.
- [29] T. M. Sutton and B. N. Aviles, “Diffusion theory methods for spatial kinetics calculations,” *Progress in Nuclear Energy*, vol. 30, pp. 119–182, 1996.
- [30] H. L. R. C. JF. Kirn, J. Boland, “Reactor physics measurements in treat,” Tech. Rep. ANL-6173, Argonne National Laboratory, Argonne, IL, Oct 1960.

APPENDIX A

FIRST APPENDIX

Text for the Appendix follows.

APPENDIX B

A SECOND APPENDIX WHOSE TITLE IS MUCH LONGER THAN THE FIRST

Text for the Appendix follows.

B.1 Appendix Section

B.2 Second Appendix Section