Improved Quasi-Static Methods for Time-Dependent Neutron Diffusion and Implementation in Rattlesnake

Zachary M. Prince

Committee
Dr. Jean Ragusa (Chair)
Dr. Jim Morel
Dr. Bojan Popov

Final Examination for Partial Fulfillment of a Masters of Science
Department of Nuclear Engineering, Texas A&M University, College Station, TX

March 6, 2017

email: zachmprince@tamu.edu

Objective

Purpose Statement

Investigate and develop the improved quasi-static method (IQS) for minimizing computation expense in transient reactor simulations, while maintaining accuracy.

What is IQS?

- Transient neutronics method meant to minimize expensive diffusion evaluations
- Non-approximately separates spatial variance and temporal power
- Impetus is spatial variance doesn't change much in time, compared to powerb

Objectives

- Establish IQS performance for various nonlinear iteration techniques applied to transient neutronics
- Validate time step convergence in transient neutronics problems for IQS with high order discretization schemes
- Apply IQS to multiphysics simulations

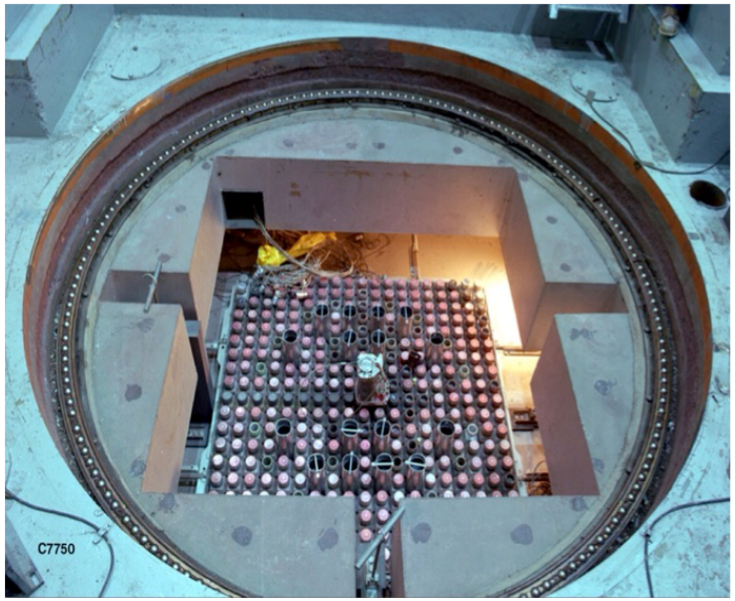
Outline

Transient Testing

- Fukushima sparked a systematic demand for accident tolerant reactor system designs
- New designs are significantly different and require experimental testing
- Tesing requires specialized facilities like ACRR at SNL and TREAT at INL

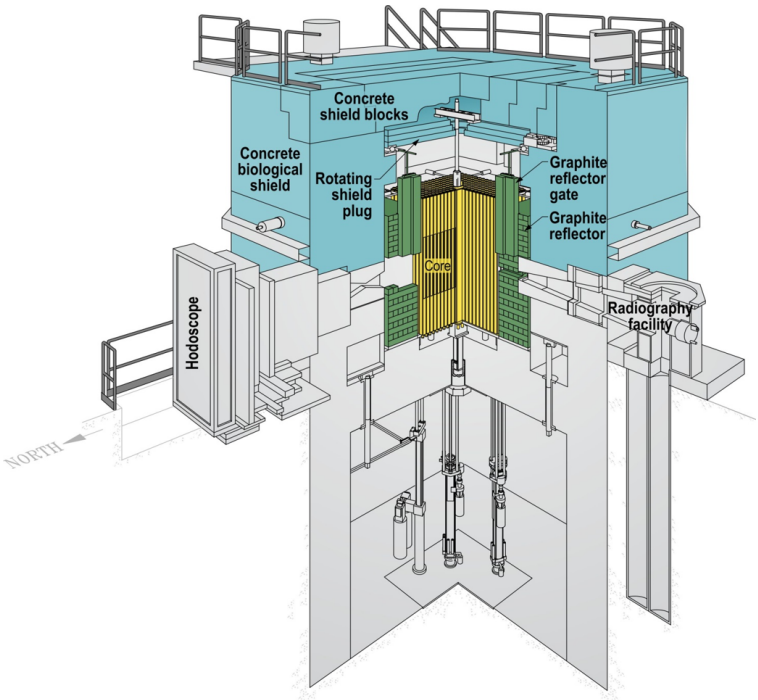
Figure: Hole believed to be caused by fuel leaking out of the pressure vessel at Fukushima Daiichi nuclear power plant

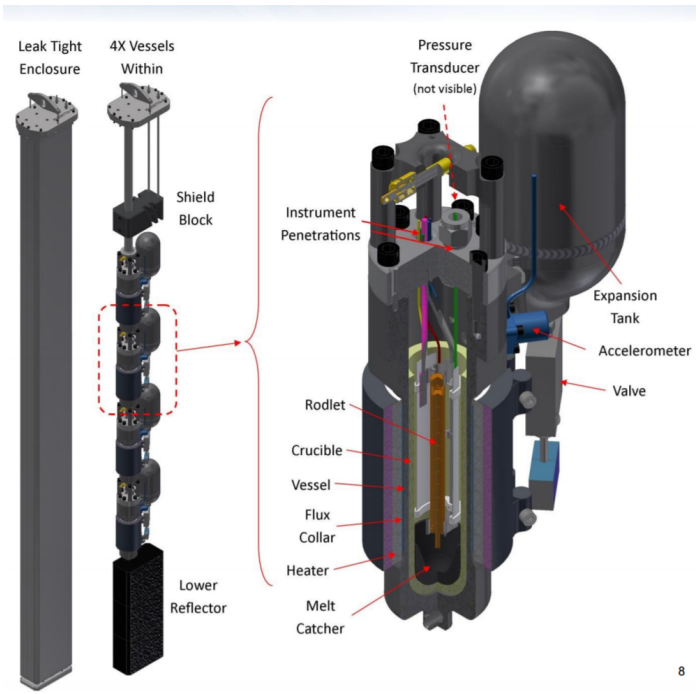
Transient Reactor Testing Facility (TREAT)

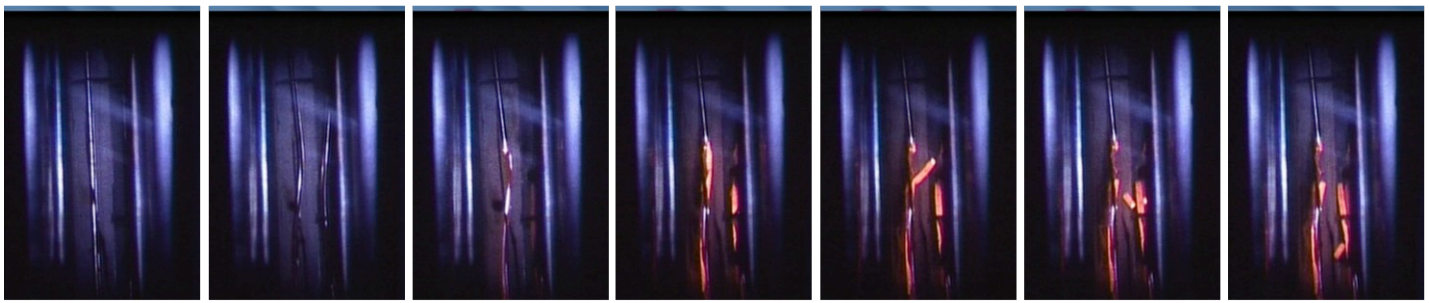


- Operation started in 1959, stand-by status in 1994, expected restart by 2020
- Designed to induce accident-like scenarios to fuel and other reactor components
- Air-cooled, graphite moderated, 100 kW steady-state, up to 19 GW peak transients

TREAT Experiments

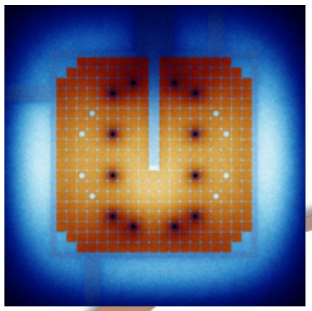




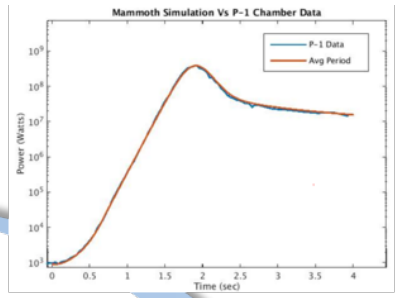


TIME

Modeling and experimentation are mutualistic

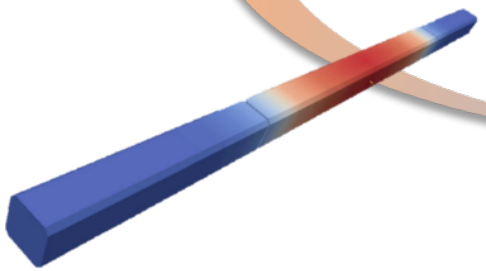


Validation

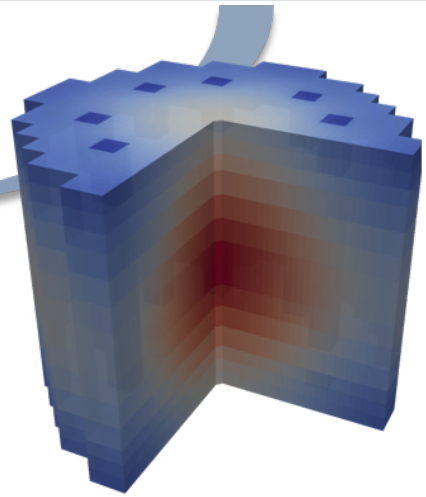


Experiments

Modelling & Simulation



Prediction



Transient reactor simulation is difficult

Neutronics

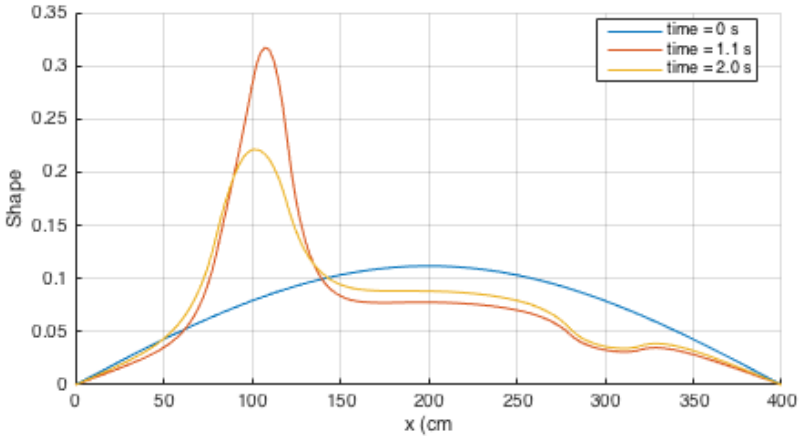
- Transient behavior of neutrons is typically evaluated using deterministic neutron transport to determine flux
- Neutron transport contains 7 independent variables: time (t), position ($\vec{r}$), energy (E), direction ($\vec{\Omega}$)
- Neutron diffusion carefully eliminates direction to reduce to 5 variables
- These equations result in a stiff system, so expensive implicit solvers are required for stability
- Each variable needs to be discretized: FEM for space, multigroup for energy, and time will be discussed later

Multiphysics

- Reactor physics involve more than the behavior of neutrons, including: fuel temperature, thermal hydraulics, fuel structure, depletion etc.
- All of these variables are dependent on one another, or coupled
- Each variable have very different solution methods
- The exponential increase in computational expense for each variable and communication between different solvers make multiphysics a daunting task

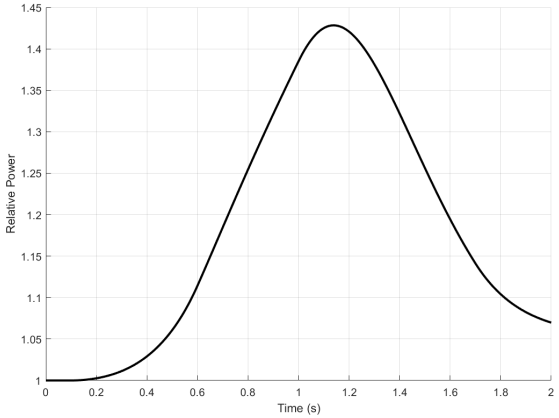
IQS mitigates neutronics expense

- IQS involves factorizing flux into space-time-energy-dependent space and time-dependent amplitude
- Shape maintains the difficulty of flux to evaluate, but amplitude is much easier
- The impetus of IQS is that shape is weakly dependent on time
- Shape and amplitude can be evaluated on different time scales to maximize efficiency



Shape($\vec{r}, E, t$)

$\times$



Amplitude(t)

Time-dependent Multigroup Diffusion

Group Fluxes ϕ^g $(1 \leq g \leq G)$ with Precursors C_i $(1 \leq i \leq I)$

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

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

- Direct time discretization of these equations is termed "implicit discretization"
- These equations are particularly stiff due to the large value of v
- Implicit schemes are necessary with many time steps

Point Reactor Kinetics Equation (PRKE)

Factorization

Point reactor kinetics involve factorizing flux into space and time dependent components:

$$\phi^g(\vec{r}, t) = \Phi^g(\vec{r}) \times p(t)$$

PRKE

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

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

- Large assumption of time-independent spatial variance
- Very inexpensive to evaluate after initial flux calculation
- "Back-of-the-envelope" calculation and postprocessing

Improved Quasi-Static Method (IQS)

IQS Factorization

Decomposition of the multigroup flux into the product of a time-dependent amplitude (p) and a space-/time-dependent multigroup shape (φ^g):

$$\phi^g(\vec{r},t) = p(t)\varphi^g(\vec{r},t)$$

- Factorization is **not** an approximation.
- Note that $p(t)$ and $\varphi^g(\vec{r},t)$ are not unique.
- Impetus is that $\varphi^g(\vec{r},t)$ is much slower varying than $\phi^g(\vec{r},t)$ and $p(t)$
- Equations for $\varphi^g(\vec{r},t)$ and $p(t)$ need to be derived

IQS Shape Equations

Shape Equations

Implementing factorization and solving for φ^g:

$$\frac{1}{v^g}\frac{\partial \varphi^g}{\partial t} = \frac{\chi_p^g}{k_{eff}} \sum_{g'=1}^{G} (1-\beta)\nu^{g'}\Sigma_f^{g'}\varphi^{g'} - \left(-\vec{\nabla}\cdot D^g\vec{\nabla} + \Sigma_r^g + \frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}\right)\varphi^g$$

$$+ \sum_{g'\neq g}^{G} \Sigma_s^{g'\to g}\varphi^{g'} + \frac{1}{p}\sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i \ , \quad 1 \leq g \leq G$$

$$\frac{dC_i}{dt} = p\sum_{g=1}^{G}\nu_{d,i}\Sigma_f^g\varphi^g - \lambda_i C_i, \quad 1 \leq i \leq I$$

Differences with original transport equation

1. An additional removal term based on $\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}\varphi^g$
2. Delayed neutron source term scaled by $\frac{1}{p}$
3. The delayed fission source in the precursor equation scaled by p

Amplitude equations (PRKE)

Principle

To obtain the amplitude equation, we multiply the shape equations with a weighting function (initial adjoint flux, ϕ^{*g}), then integrate over domain.

Notation

For brevity, the adjoint flux product and integration over domain will be represented with parenthetical notation:

$$\int_D \phi^{*g}(\vec{r}) f(\vec{r}) dr^3 = (\phi^{*g}, f)$$

Uniqueness of the factorization

In order to impose uniqueness of the factorization, one requires:

$$K_0 = \sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right) = constant$$

PRKE for IQS

PRKE

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

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

PRKE Coefficients

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

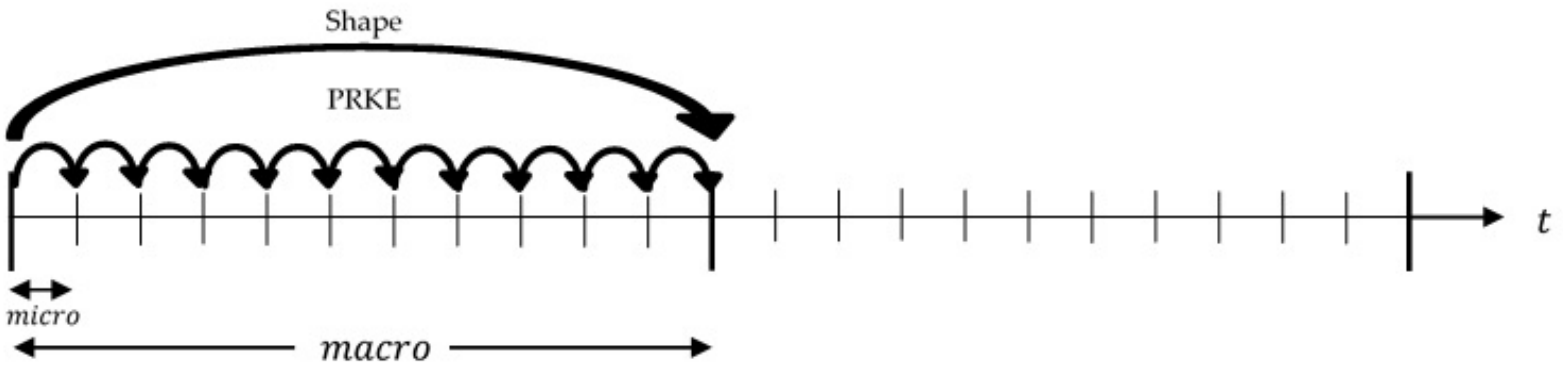
$$\frac{\bar{\beta}}{\Lambda} = \sum_{i=1}^{I} \frac{\bar{\beta}_i}{\Lambda} = \frac{1}{k_{eff}} \frac{\sum_{i=1}^{I} \sum_{g=1}^{G} (\phi^{*g}, \beta_i \nu^g \Sigma_f^g \varphi^g)}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right)}$$

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

Quasi-Static Process

Time scales and IQS solution process

Because solving for the shape can be expensive, especially in two or three dimensions, it is attractive to make the assumption that the shape is weakly time-dependent so the shape can be computed after a multitude of PRKE calculations:

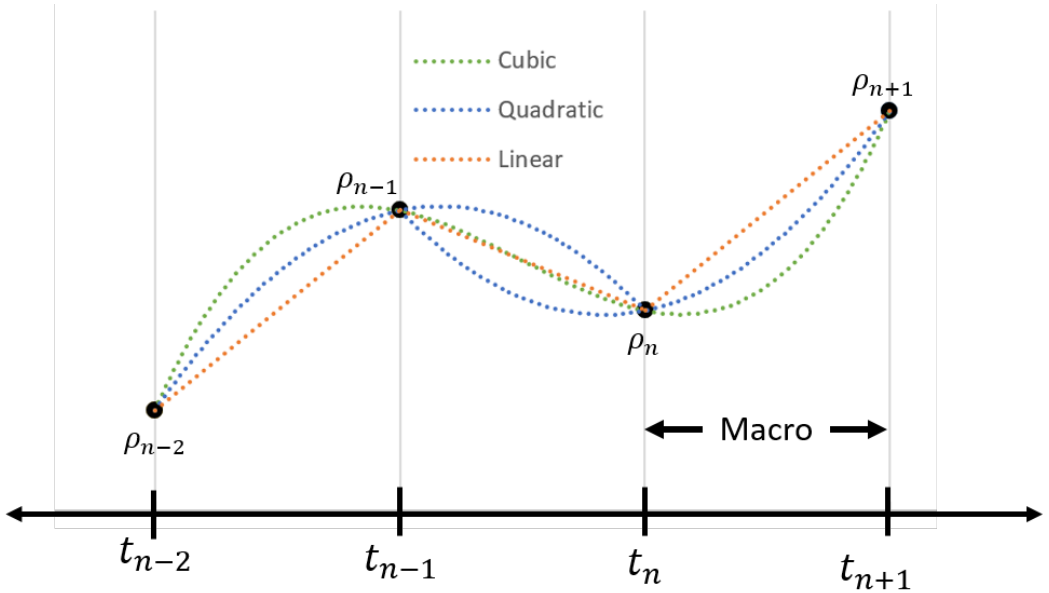


Interpolation of PRKE Parameters

Lagrange Interpolation

- Parameters are evaluated at each macro step using the relevant shape
- Parameters are interpolated between macro steps for PRKE
- Most applications use linear interpolation
- Higher order Lagrange is possible:

$$\rho(t) = \sum_{j=0}^{k} \rho_{n-j+1} \prod_{m \neq j}^{k} \frac{t - t_{n-m+1}}{t_{n-j+1} - t_{n-m+1}}$$



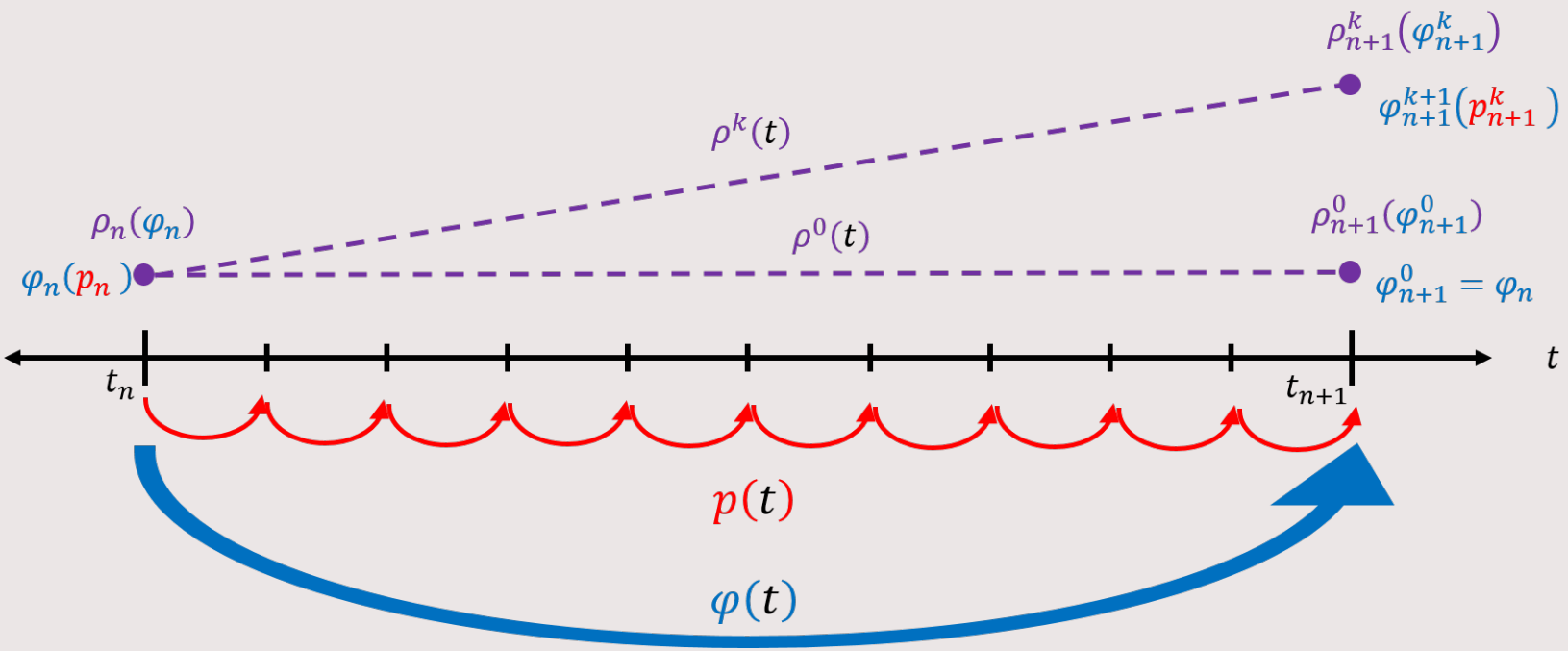
IQS is nonlinear

Nonlinear systems need an iterative solution process

There are two general iteration processes:

1. Fixed-point (Picard): back and forth corrections between amplitude and shape with relevant convergence criteria
2. Newton: residual-Jacobian based approach on shape

Fixed-Point Process



Relevant Picard convergence criteria

IQS convergence criteria

- Shape based convergence:

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

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

- Reactivity based:

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

- Amplitude based:

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

- Uniqueness consistency:

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

IQS Predictor-Corrector

IQS P-C Linearizes the System

IQS P-C linearizes the system and avoids iterations on the shape:

1. Evaluate multigroup diffusion equation to get predicted flux $\phi_{n+1}^{g,pred}$
2. Scale predicted flux to obtain shape:

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

3. Compute PRKE parameters at t_{n+1}
4. Evaluate PRKE along micro step using interpolated parameters to obtain p_{n+1}
5. Scale φ_{n+1}^{g} to obtain corrected flux:

$$\phi_{n+1}^{g,corr} = p_{n+1} \times \varphi_{n+1}^{g}$$

Advantage: No IQS nonlinear iteration is necessary
Disadvantage: Assumes $\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g}\varphi_{n+1}^{g}\right)$ is inherently constant

Time Discretization

Time Discretizations

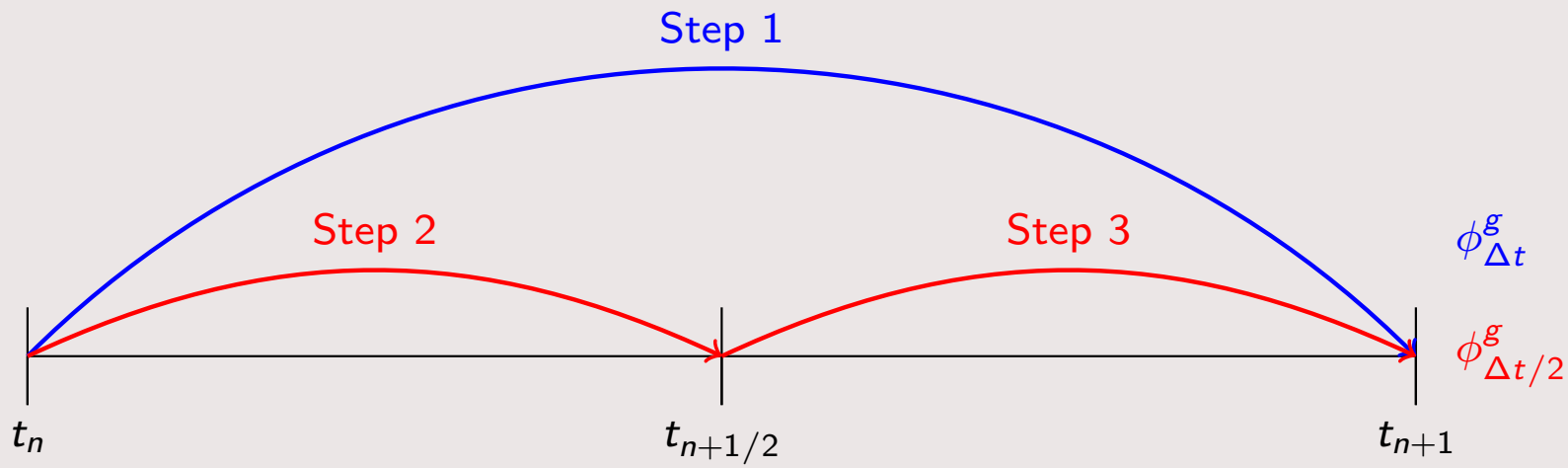
- Implicit Euler - 1st Order
- Crank-Nicolson - 2nd Order
- Backward Difference Formulae (BDF) - 1st, 2nd, 3rd, 4th Order
- Singly-Diagonally-Implicit Runge-Kutta (SDIRK) - 3rd Order (SDIRK33)

Time Step Adaptation

- Determining appropriate time step is difficult without prior knowledge of the solution
- Some areas of the transient may require smaller time steps for constant accuracy
- Time adaptation predicts the local truncation error and computes a time step appropriate for a given tolerance
- Examples:
  - Step doubling
  - Embedded Runge-Kutta

Step doubling time adaption was chosen for testing

Step Doubling Solution Process

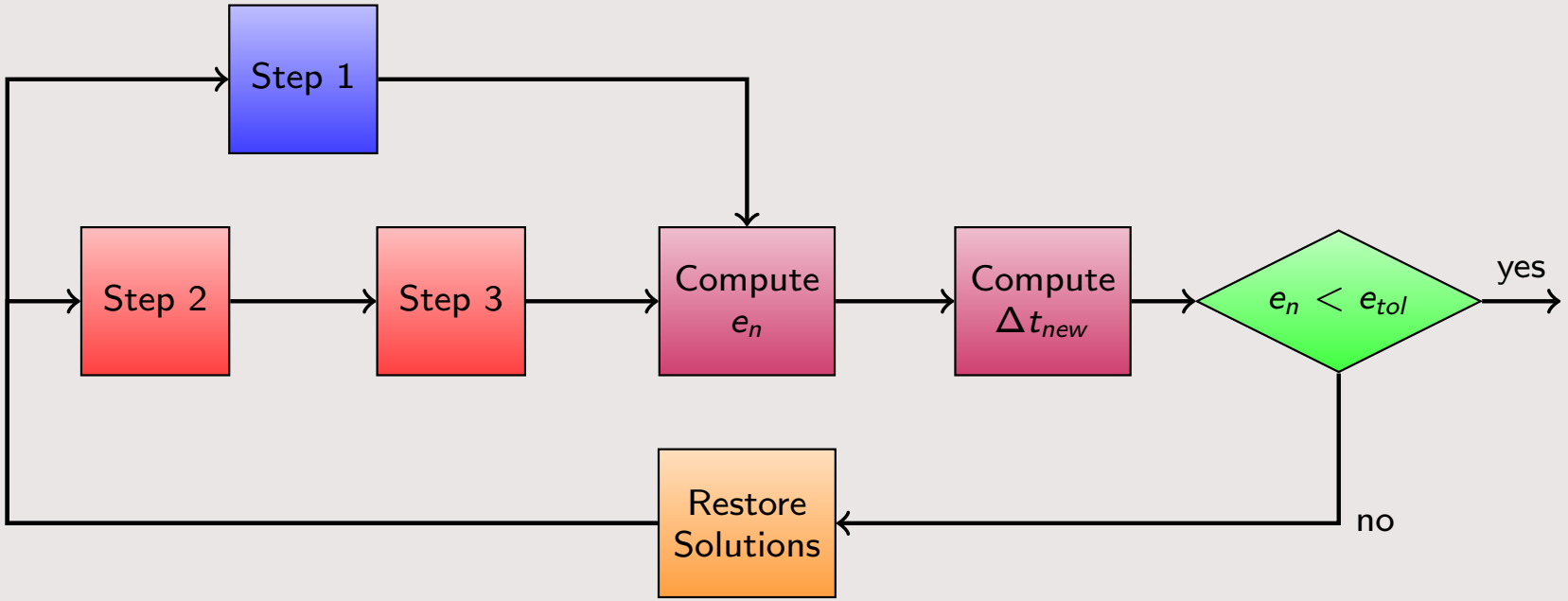


$$e_n = \frac{\left\| \sum_{g=1}^{G} \left(\phi^g_{\Delta t/2} - \phi^g_{\Delta t} \right) \right\|_{L^2}}{\max\left(\left\| \sum_{g=1}^{G} \phi^g_{\Delta t/2} \right\|_{L^2}, \left\| \sum_{g=1}^{G} \phi^g_{\Delta t} \right\|_{L^2} \right)}$$

$$\Delta t_{new} = S_f \Delta t \left[\frac{e_{tol}}{e_n} \right]^{1/(p+1)}$$

Step Doubling Solution Process

Programming Visualization



Each Step undergoes:

- Shape evaluation
- PRKE evaluations
- Multiphysics evaluations
- Iterations for convergence of amplitude, shape, and multiphysics

Treatment of delayed neutron precursors

Delayed neutron precursor equation is an ODE

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

Time Integration of Precursors

- Trapezoid Rule:

$$C_i^{n+1} = \frac{1-0.5\Delta t\lambda_i}{1+0.5\Delta t\lambda_i}C_i^n + \frac{0.5\Delta t\beta_i}{1+0.5\Delta t\lambda_i}\sum_{g=1}^{G}\left[(\nu\Sigma_f^g\varphi^g)^n p^n + (\nu\Sigma_f^g\varphi^g)^{n+1}p^{n+1}\right]$$

- Analytical Integration:

$$C_i^{n+1} = C_i^n e^{-\lambda_i\Delta t} + \sum_{g=1}^{G}\left(\hat{a}_{2,i}(\nu\Sigma_f^g\varphi^g)^{n+1} + \hat{a}_{1,i}(\nu\Sigma_f^g\varphi^g)^n\right)\beta_i$$

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

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

Multiphysics: Adiabatic heat-up with absorption cross-section feedback

Implemented Form

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

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

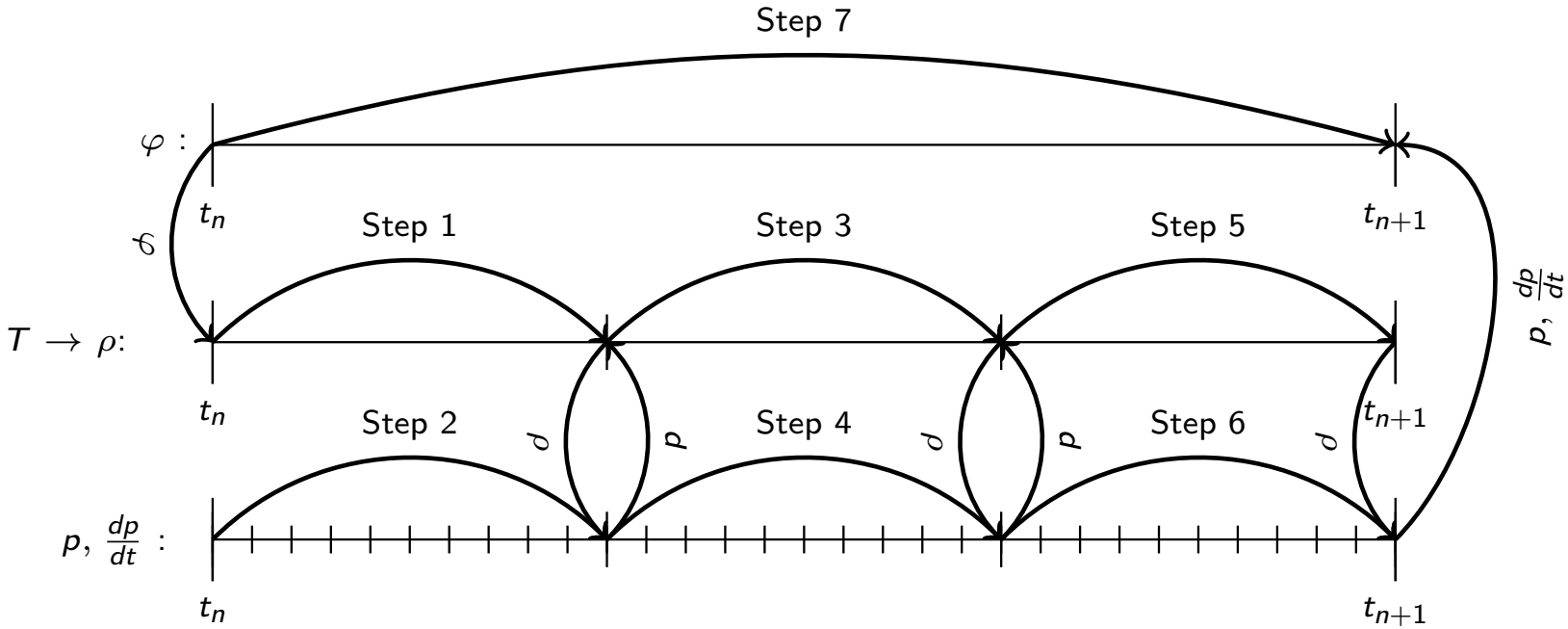
Analytical temperature integration with IQS

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

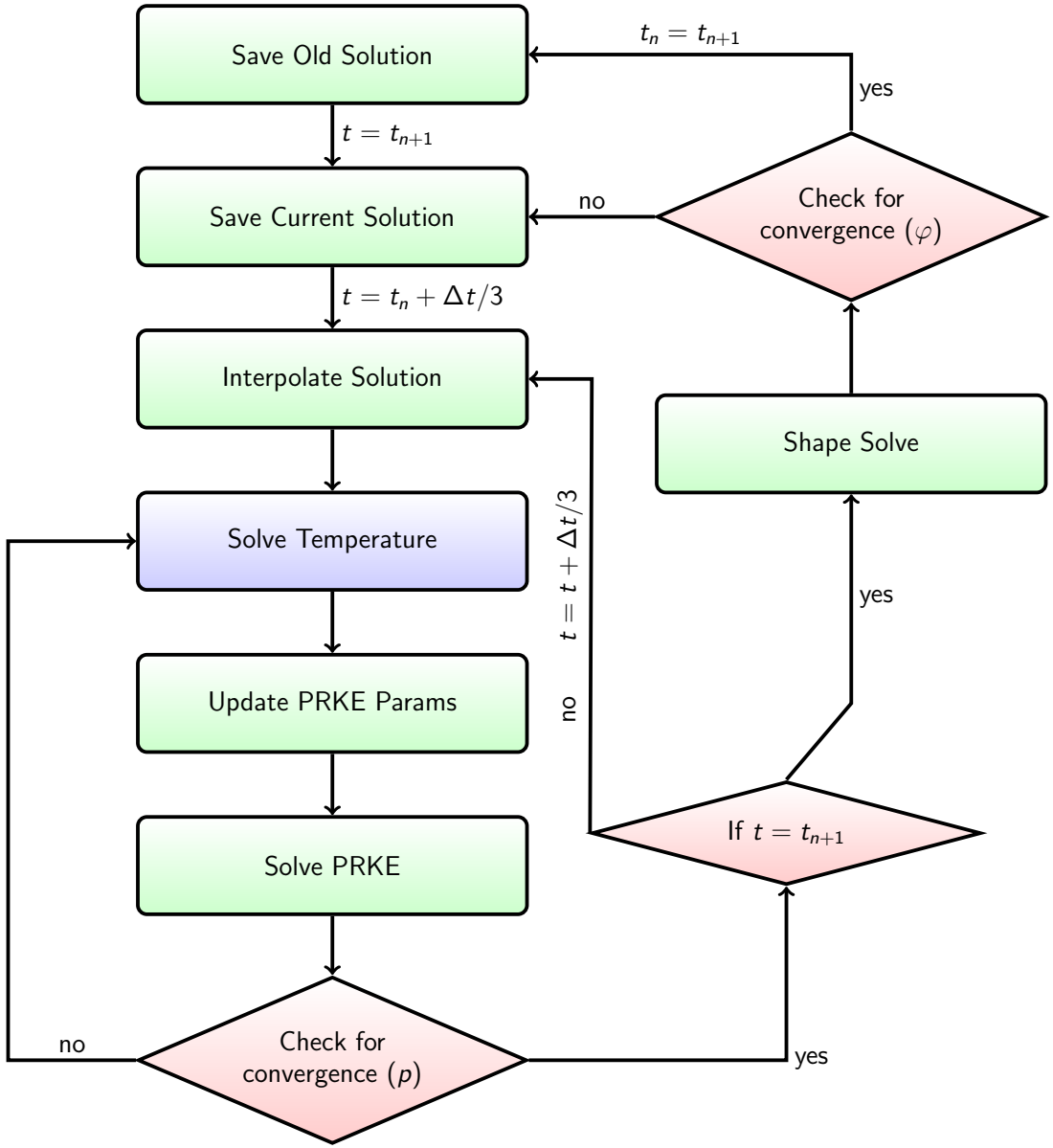
$$a_1 = \int_{t_n}^{t_{n+1}} \left(\frac{t_{n+1}-t'}{\Delta t}\right) p(t')dt'$$

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

Intermediate time scale for temperature



Time scale programming logic



Time Scale Analysis

Dynamical Time Scale

- The time variance of each physics (θ) can be quantified by defining a dynamical time scale (τ):

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

- Finite difference approximation for $d\theta/dt$ and average for $1/\theta$
- Only temporal behavior is of interest, so the L^2 norm will be taken of each quantity, resulting in:

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

- According to the a priori hypothesis, τ is large for shape, somewhat smaller for temperature, and much smaller for amplitude and flux

MATLAB Kinetics Prototype Code

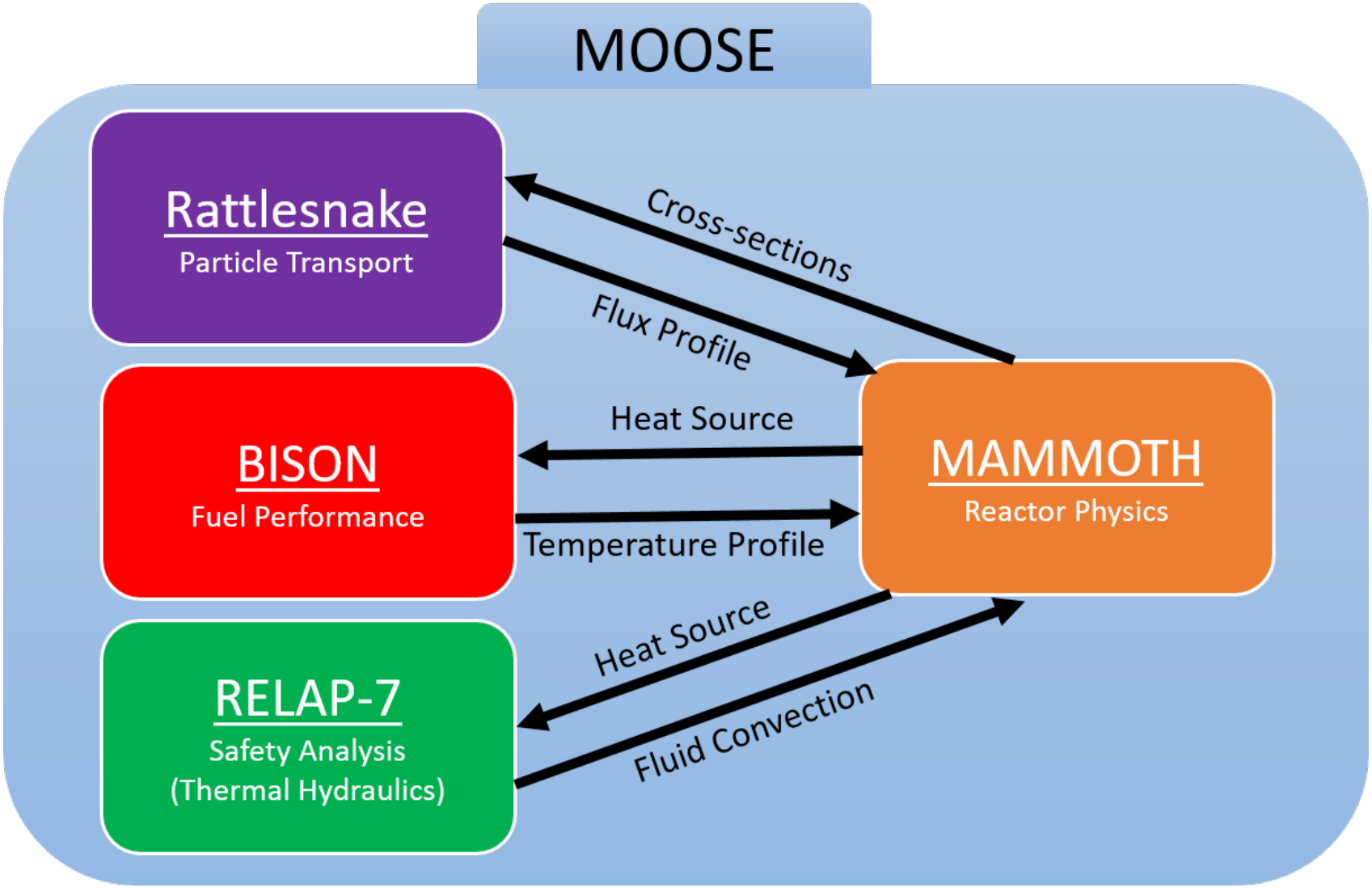
Overview

- One-dimensional, single-group, time-dependent neutron diffusion with precursors
- Heterogenous medium using block-like regions
- Step and ramp perturbations in parameters
- Manufactured solutions using symbolic capability
- FEM meshing with any order shape functions

Solution Methods

1. Implicit Discretization
   - Precursor variable coupling
   - Theta precursor elimination
   - Analytical precursor elimination
2. IQS
   - Precursor variable coupling
   - Theta precursor elimination
   - Analytical precursor elimination
3. IQS P-C
   - Theta precursor elimination
   - Analytical precursor elimination

Reactor Simulation in MOOSE



Results

One Dimensional Example

Geometry

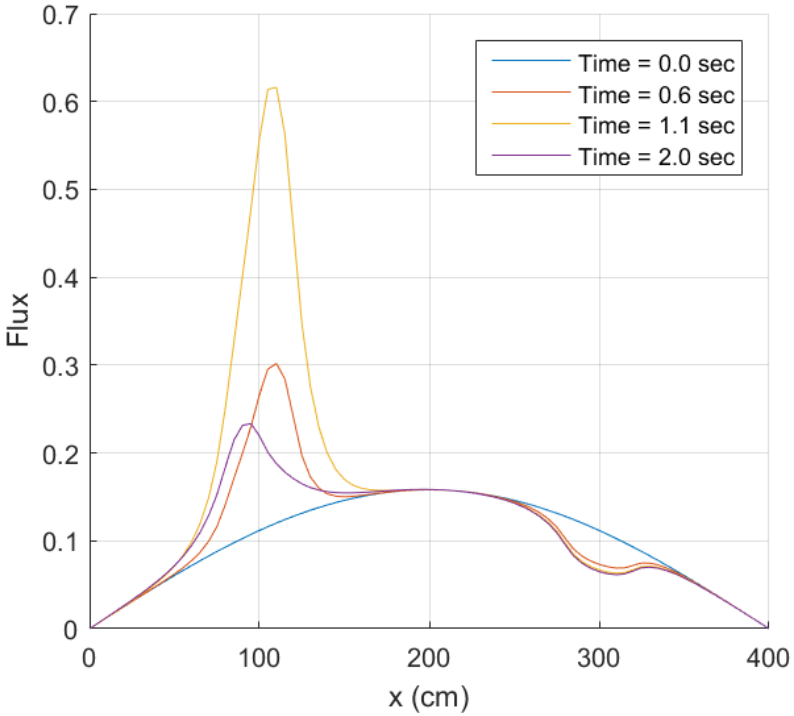
1	1	1	1	2	3	1	1	1	1	1	1	1	1	4	4	1	1	1	1

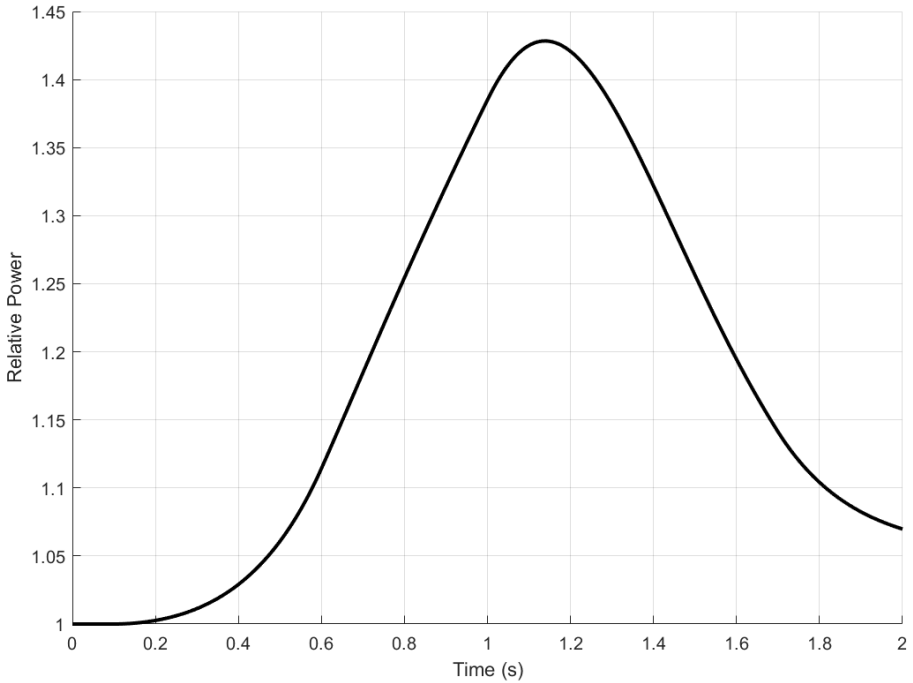
- Region 2: Ramp Σ_a decrease (rod removal)
- Region 3: Ramp Σ_a decrease then increase (rod removal then insertion)
- Region 4: Ramp Σ_a increase (rod insertion)

Region Perturbations

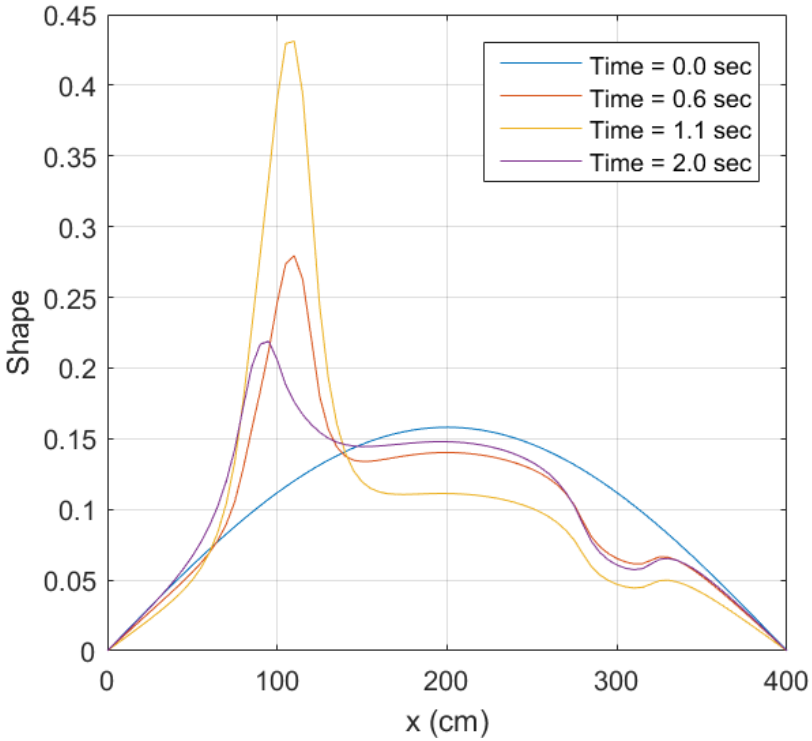
Region	Material Property	0.0 s	0.1 s	0.6 s	1.0 s	1.7 s
2	$\Sigma_a(cm^{-1})$	1.1	1.1	1.095	1.095	1.095
3	$\Sigma_a(cm^{-1})$	1.1	1.1	1.09	1.09	1.1
4	$\Sigma_a(cm^{-1})$	1.1	1.1	1.105	1.105	1.105

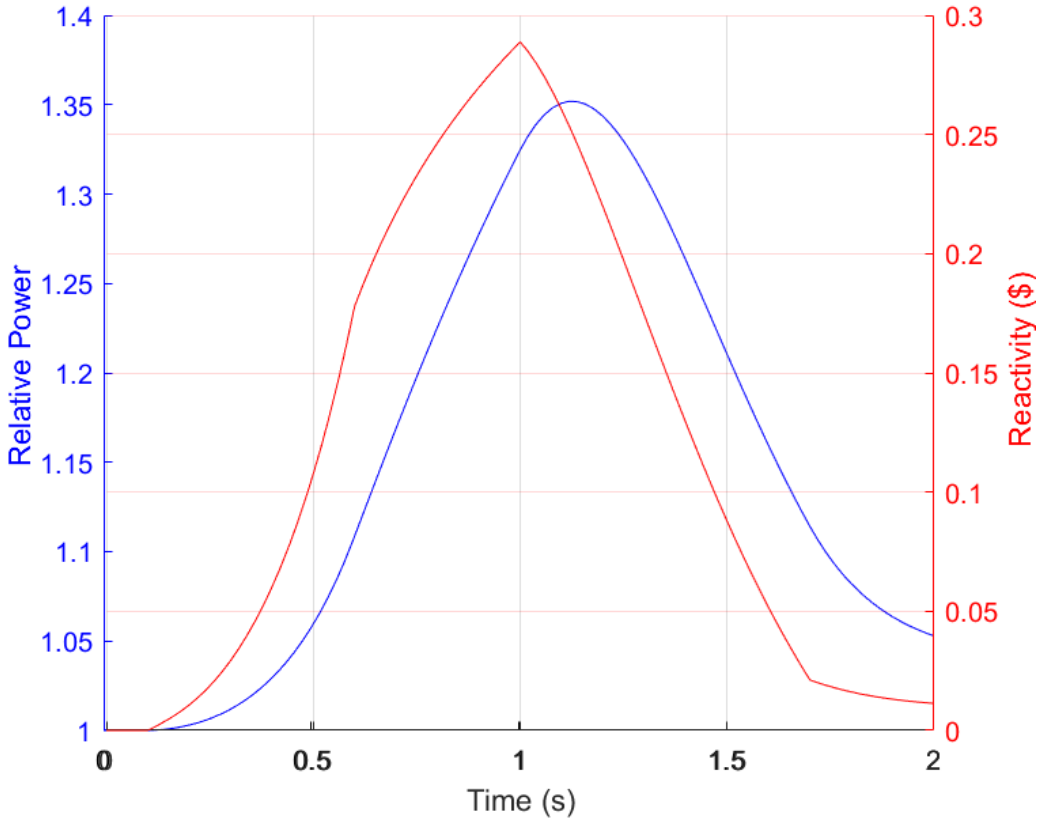
Baseline Results



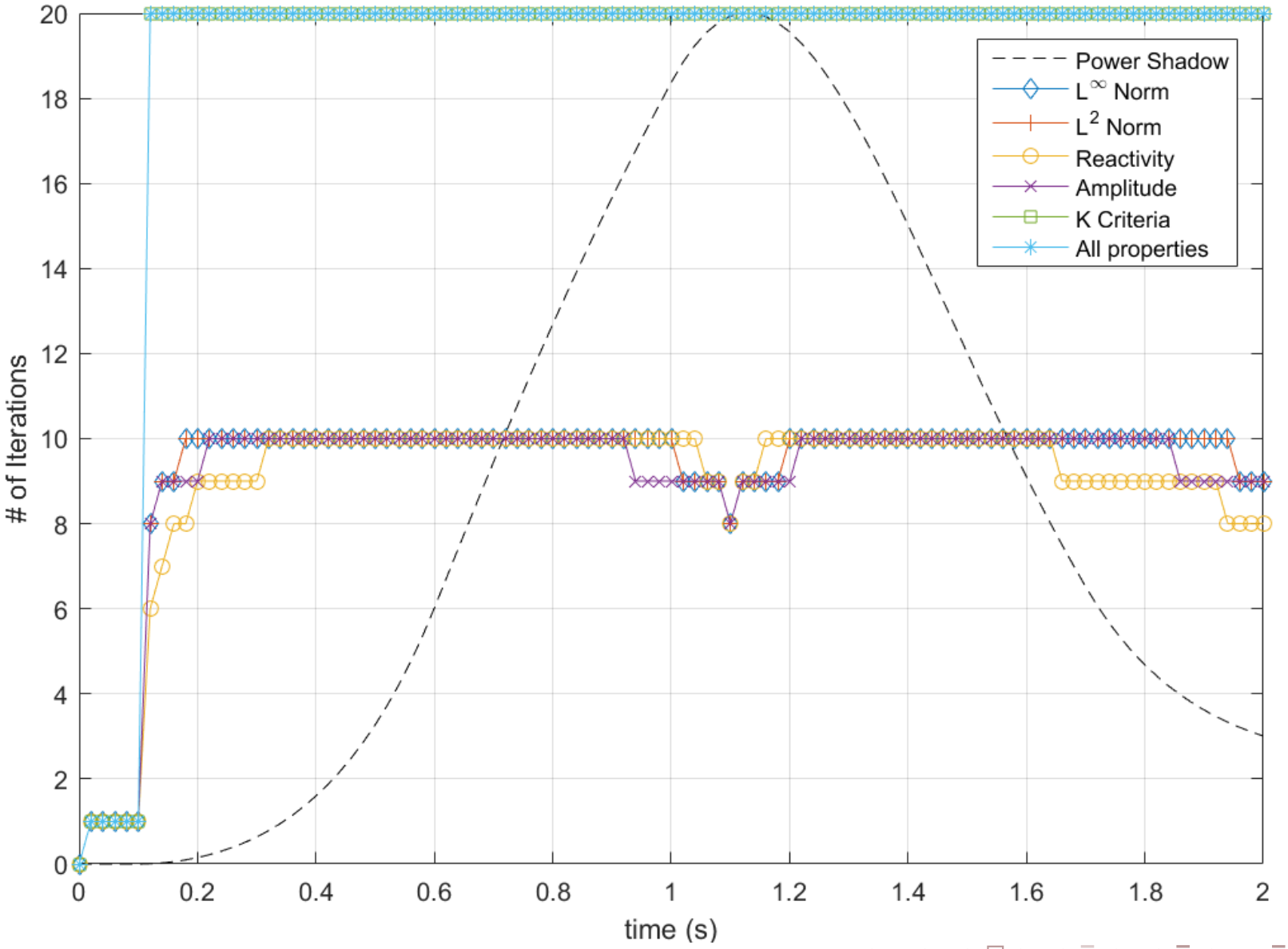


IQS Results

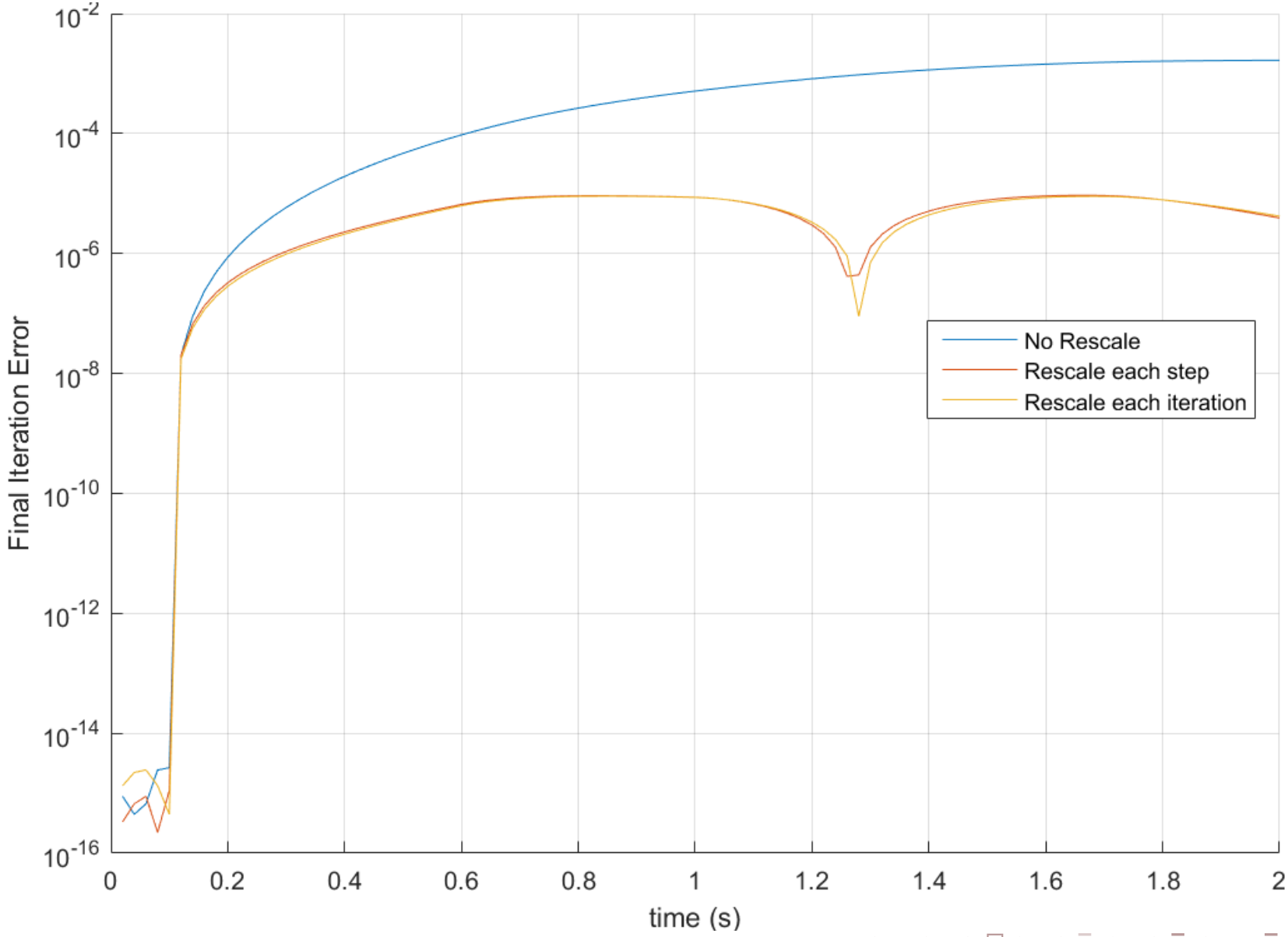

Time = 0.0 sec
Time = 0.6 sec
Time = 1.1 sec
Time = 2.0 sec
Shape
x (cm)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0
100
200
300
400


Relative Power
Reactivity ($)
Time (s)
1
1.05
1.1
1.15
1.2
1.25
1.3
1.35
1.4
0
0.05
0.1
0.15
0.2
0.25
0.3
0
0.5
1
1.5
2

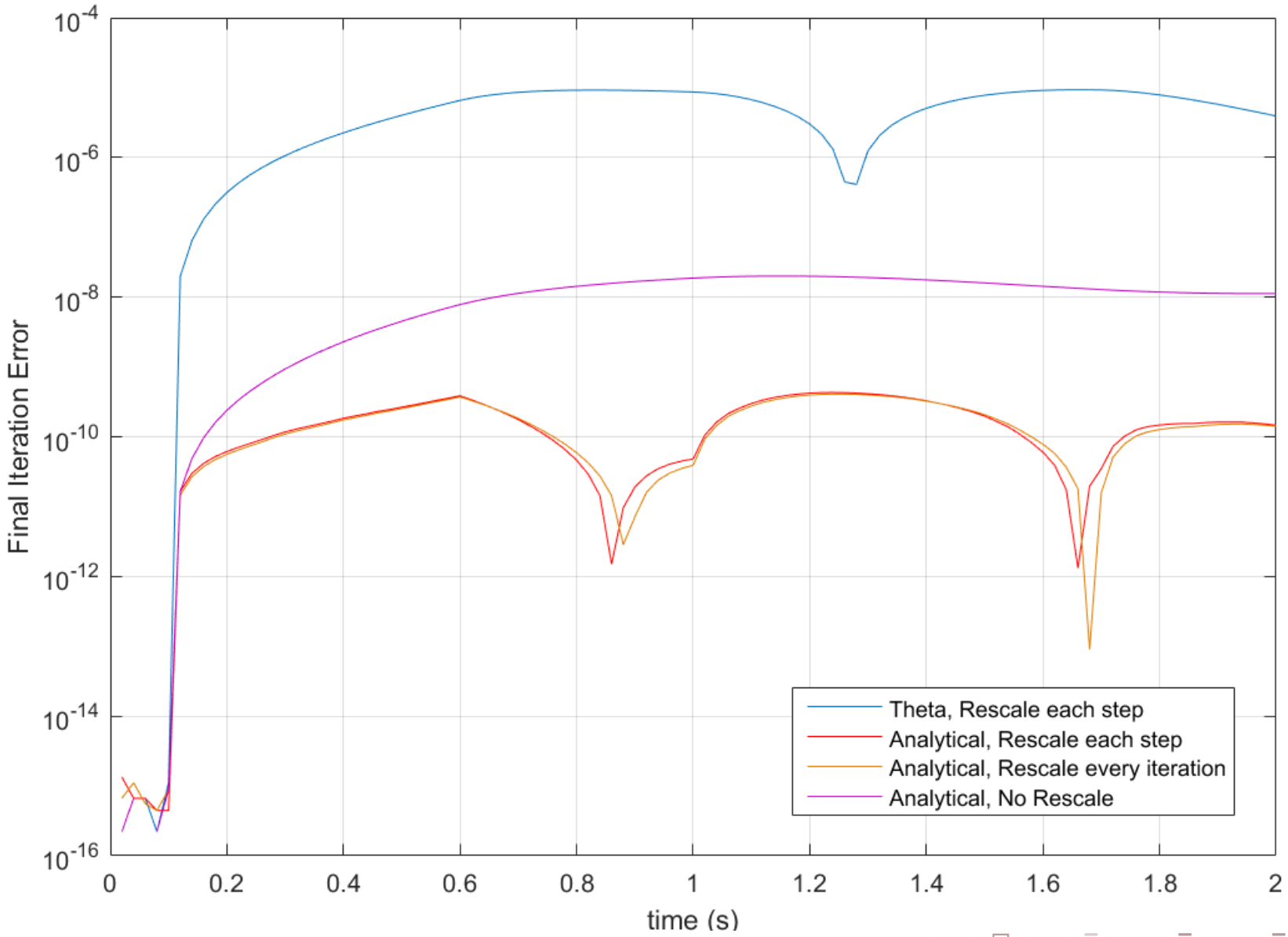
Convergence criteria convergence behavior



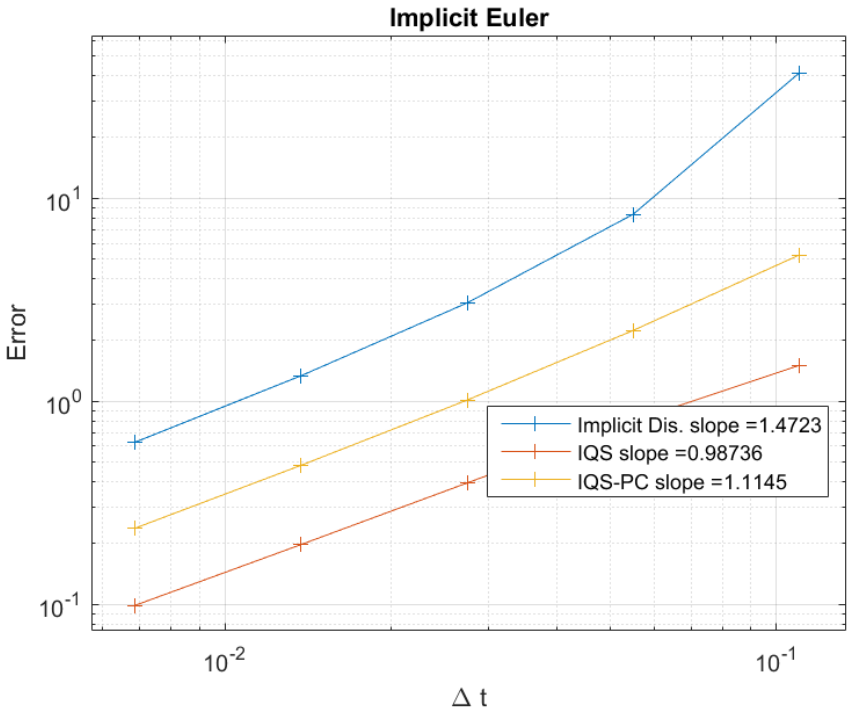
IQS converged error with theta integration of precursors

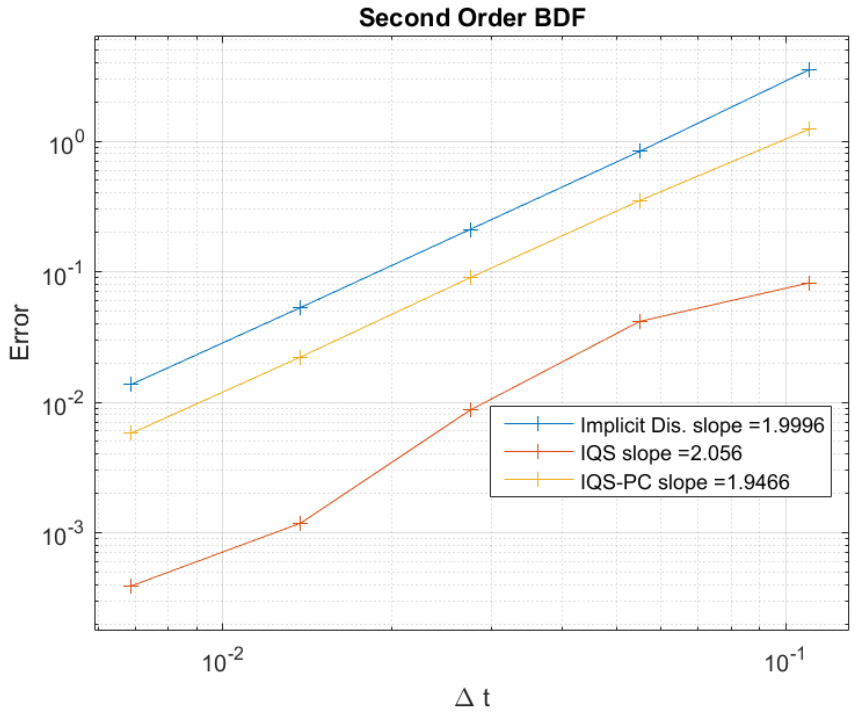


IQS converged error with analytical integration of precursors

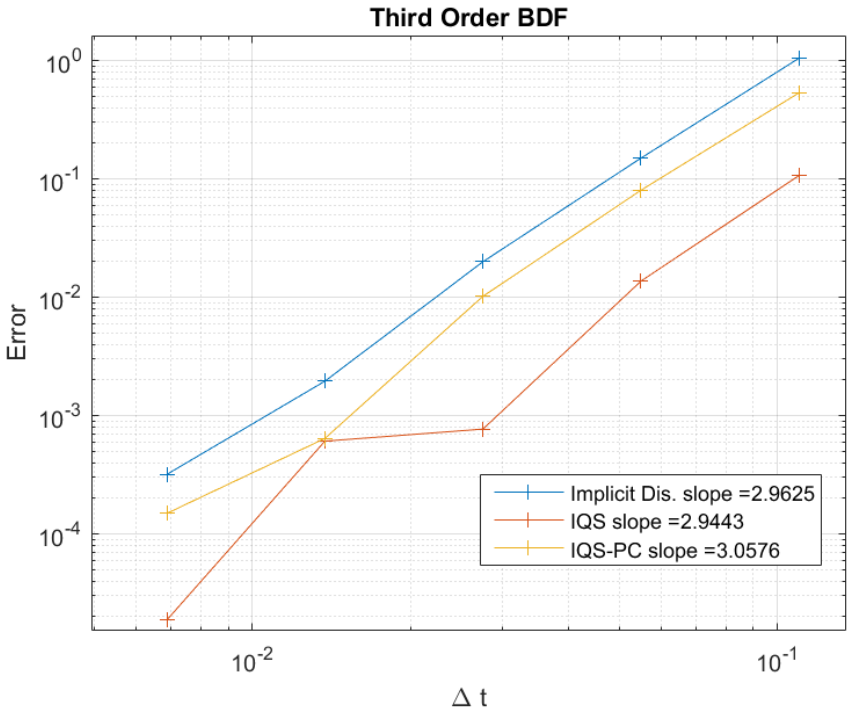


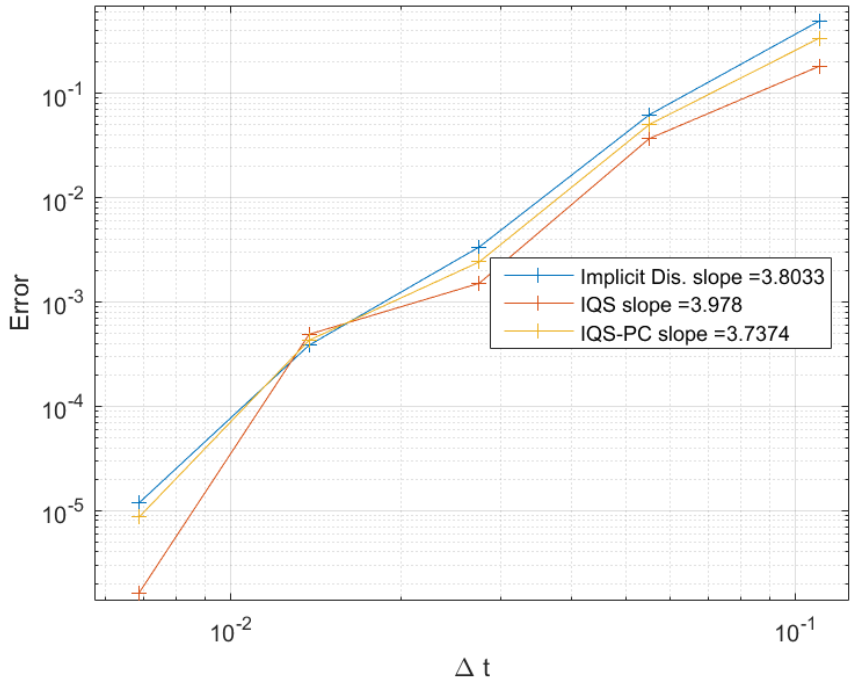
First and second order time step error convergence



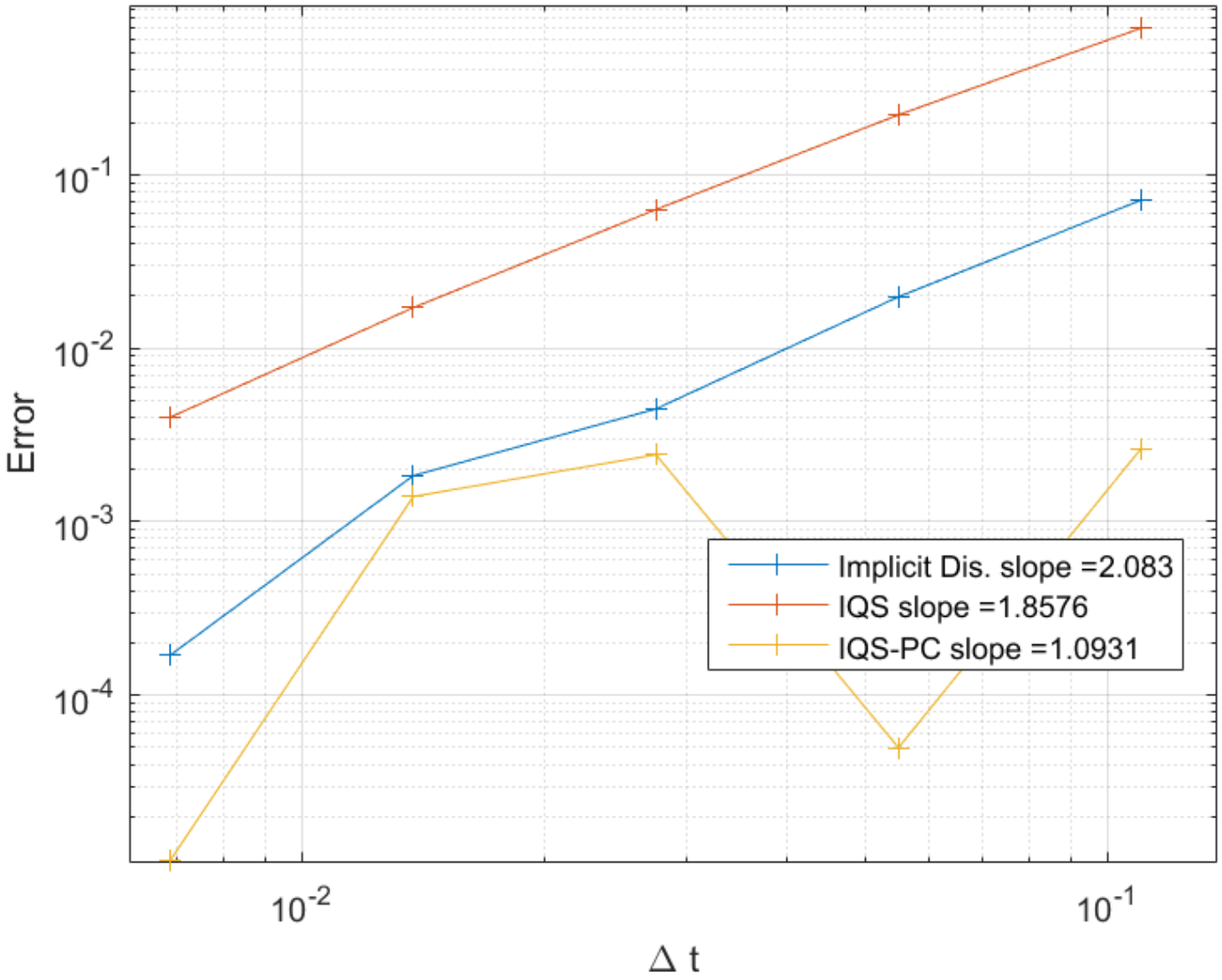


Third and fourth order time step error convergence

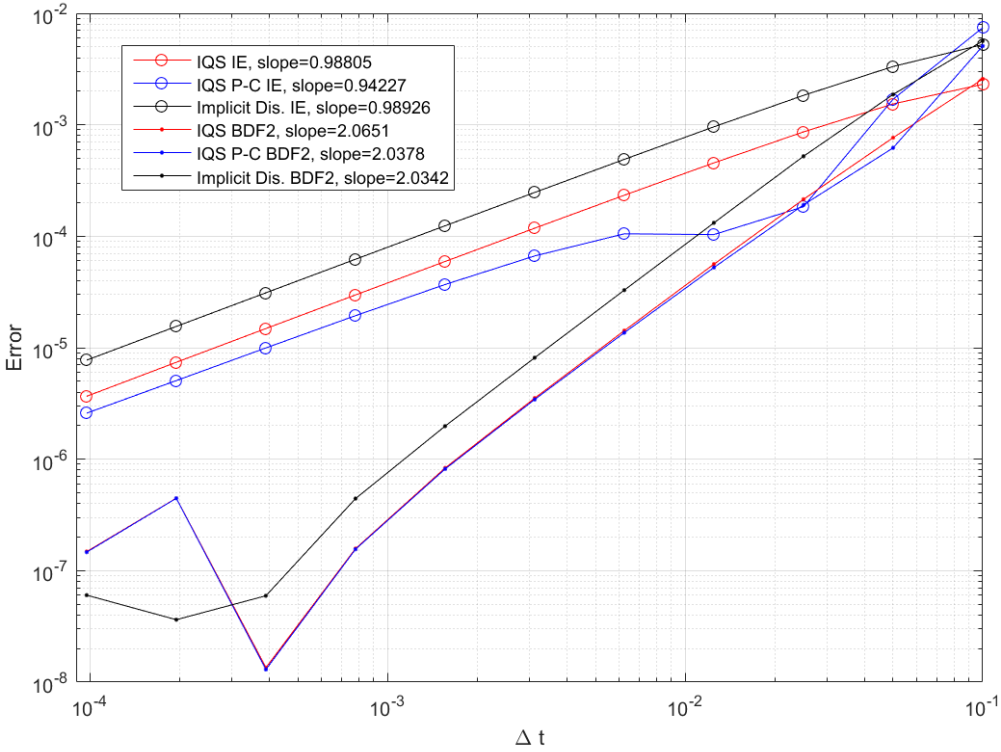


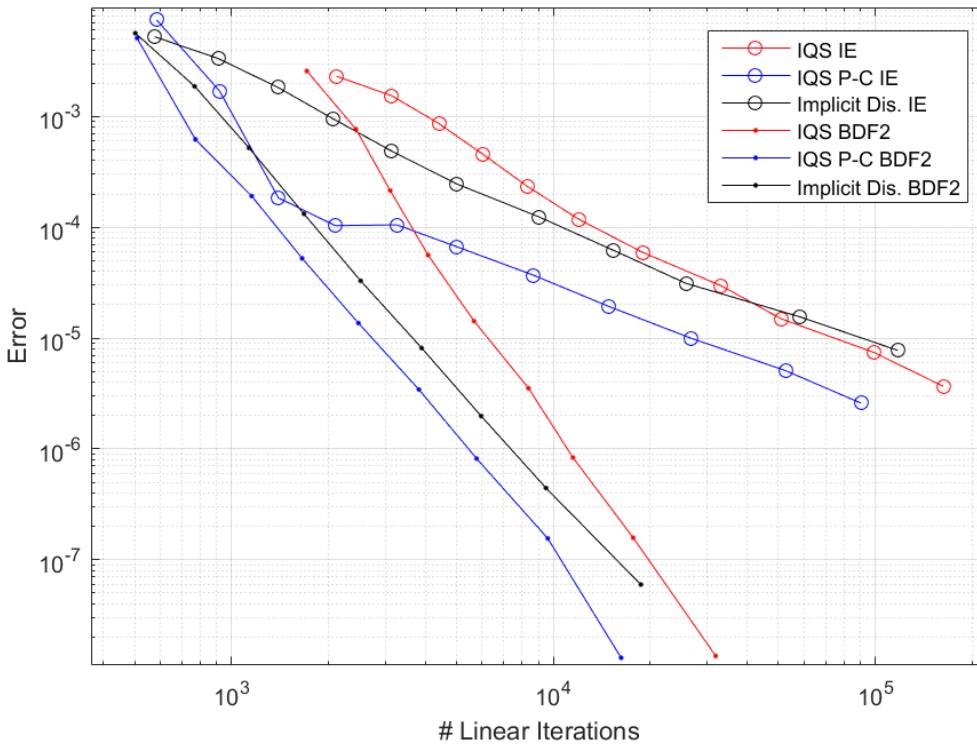


SDIRK33 time step error convergence

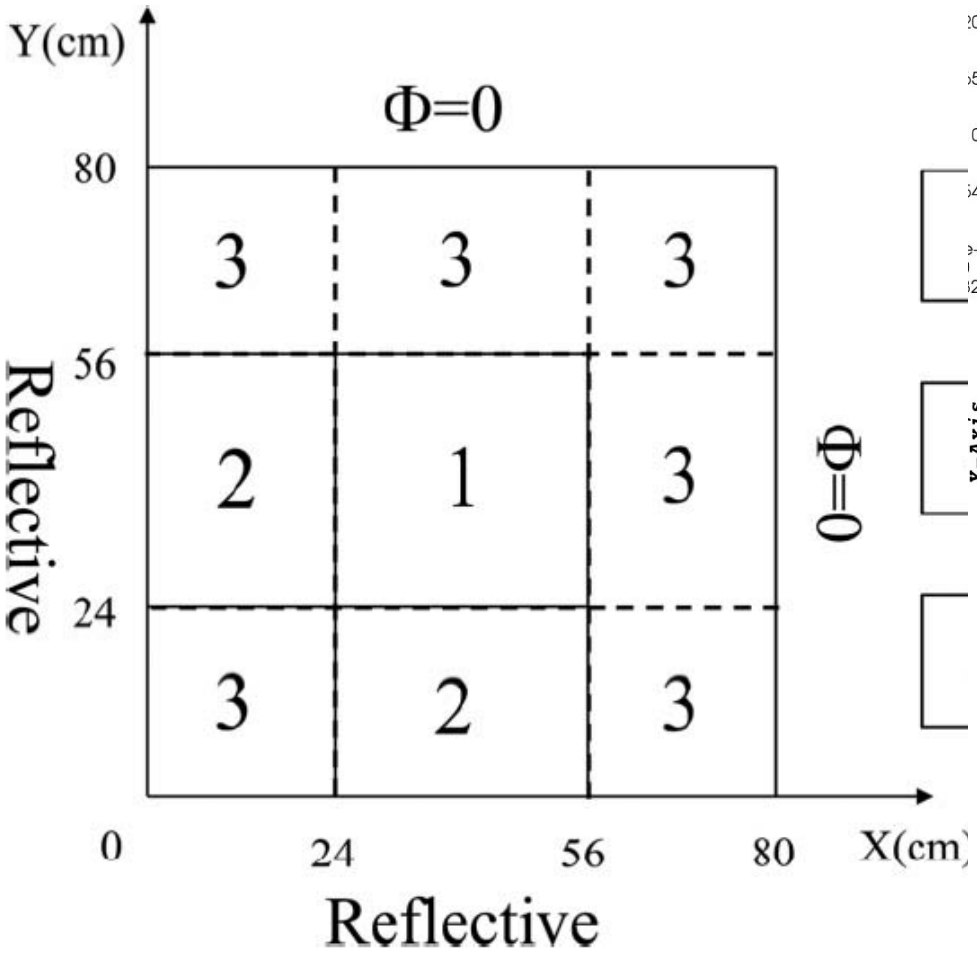


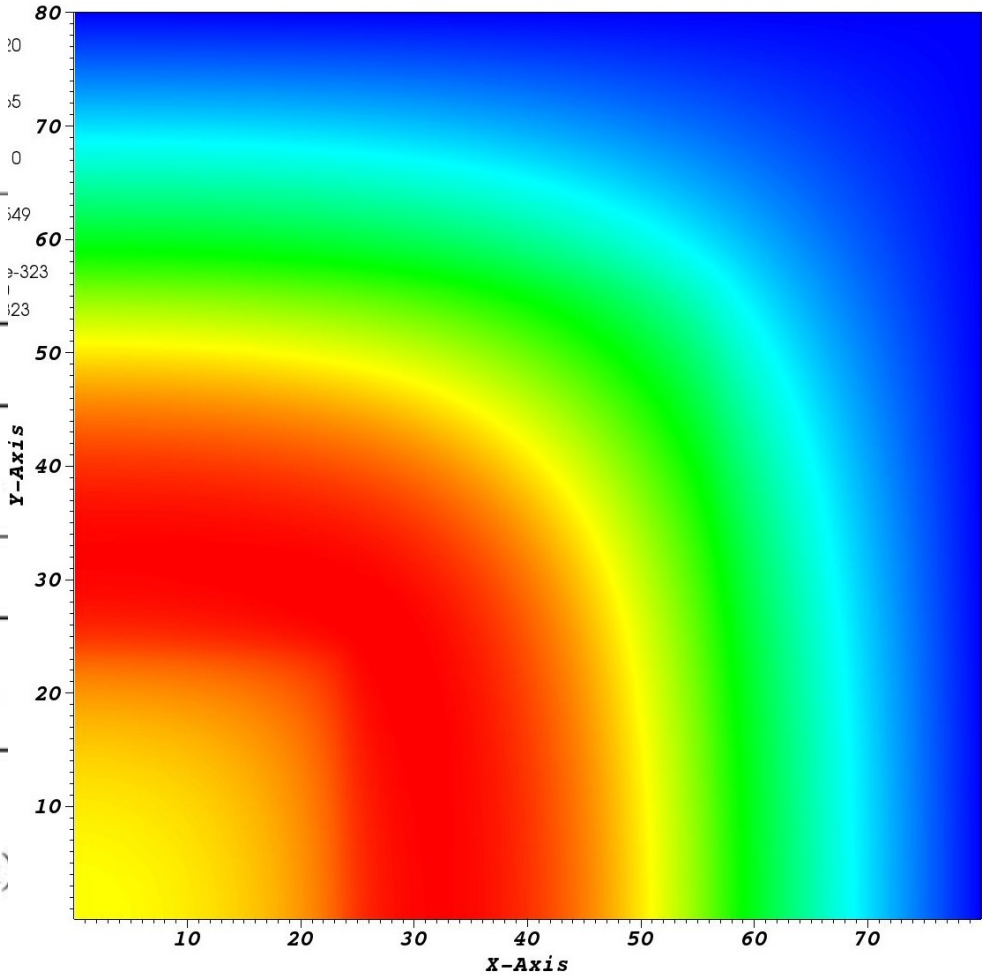
Rattlesnake time step error convergence





TWIGL Benchmark





TWIGL Power Profile

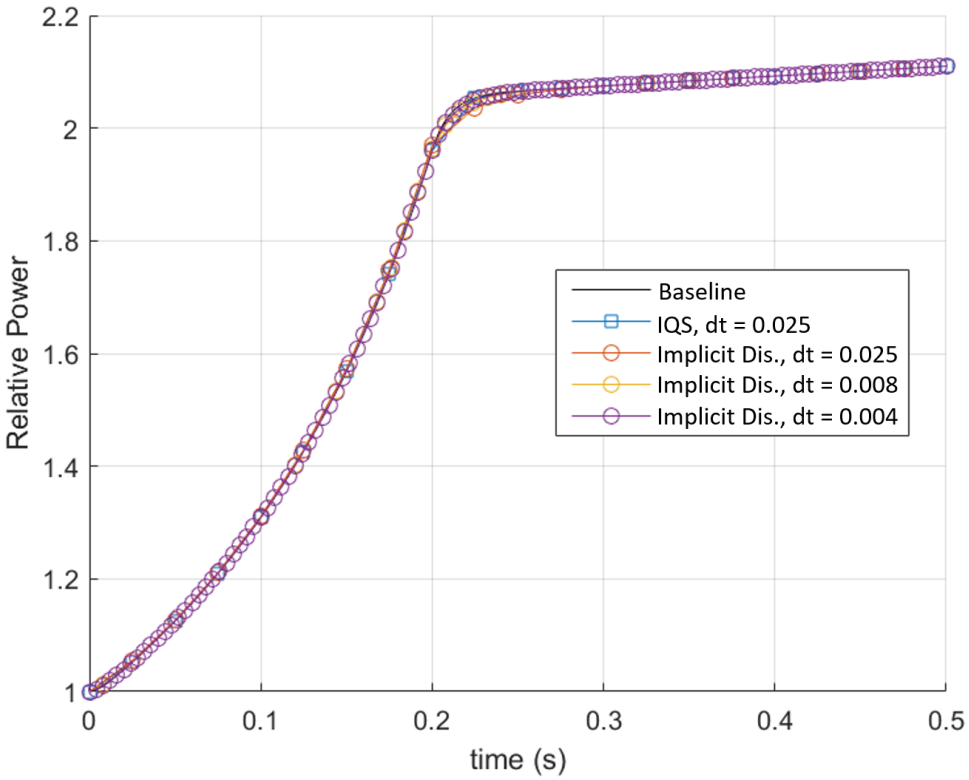


Figure: TWIGL Power Profile

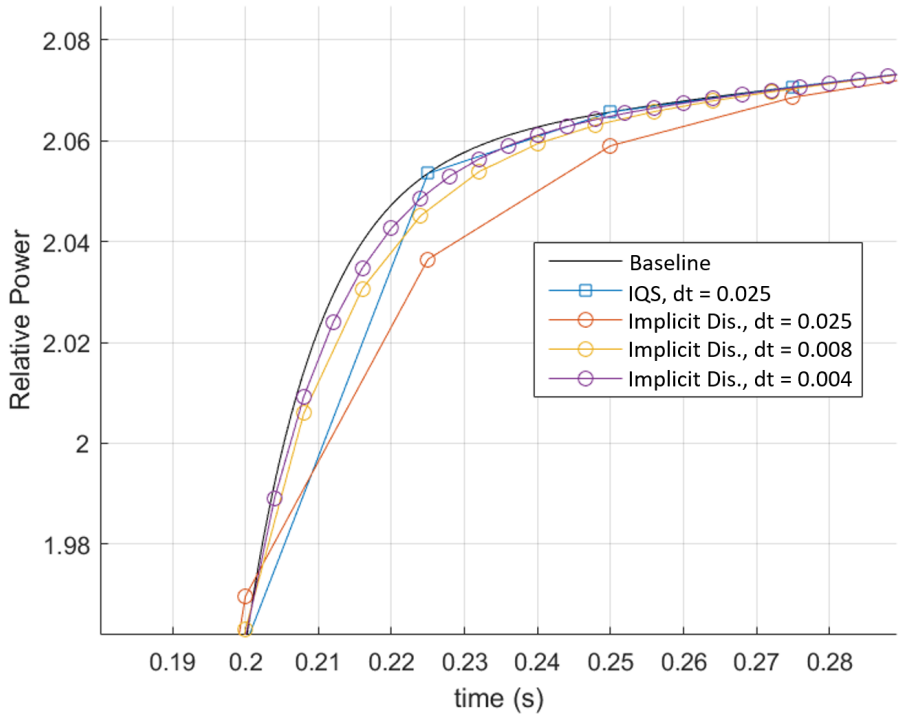
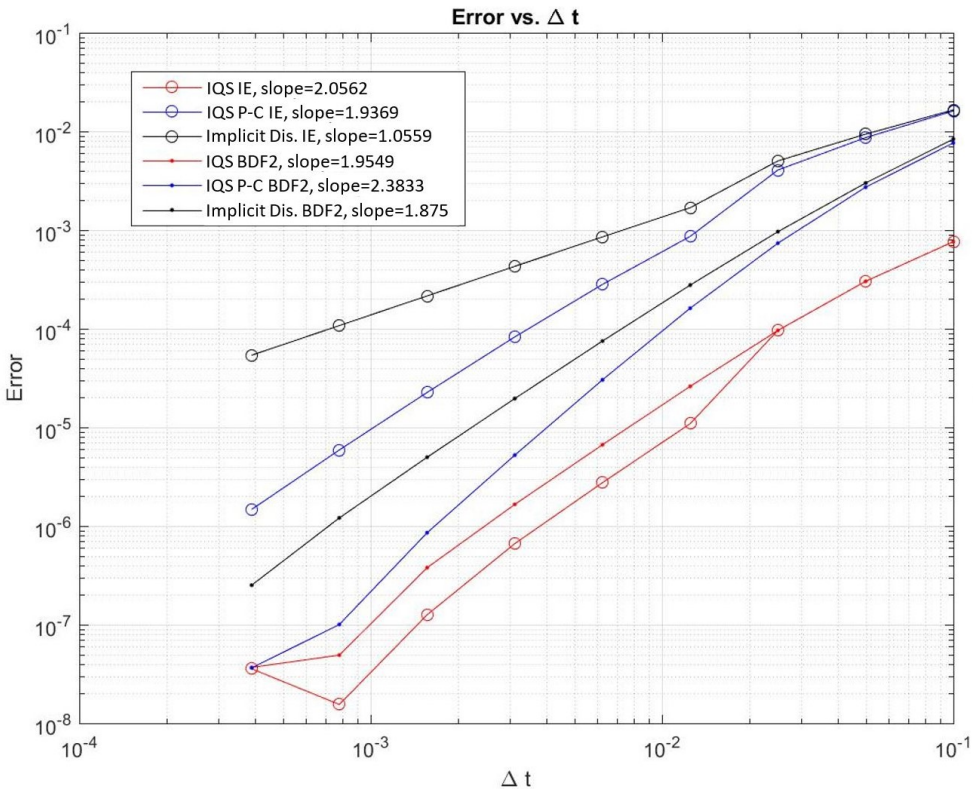
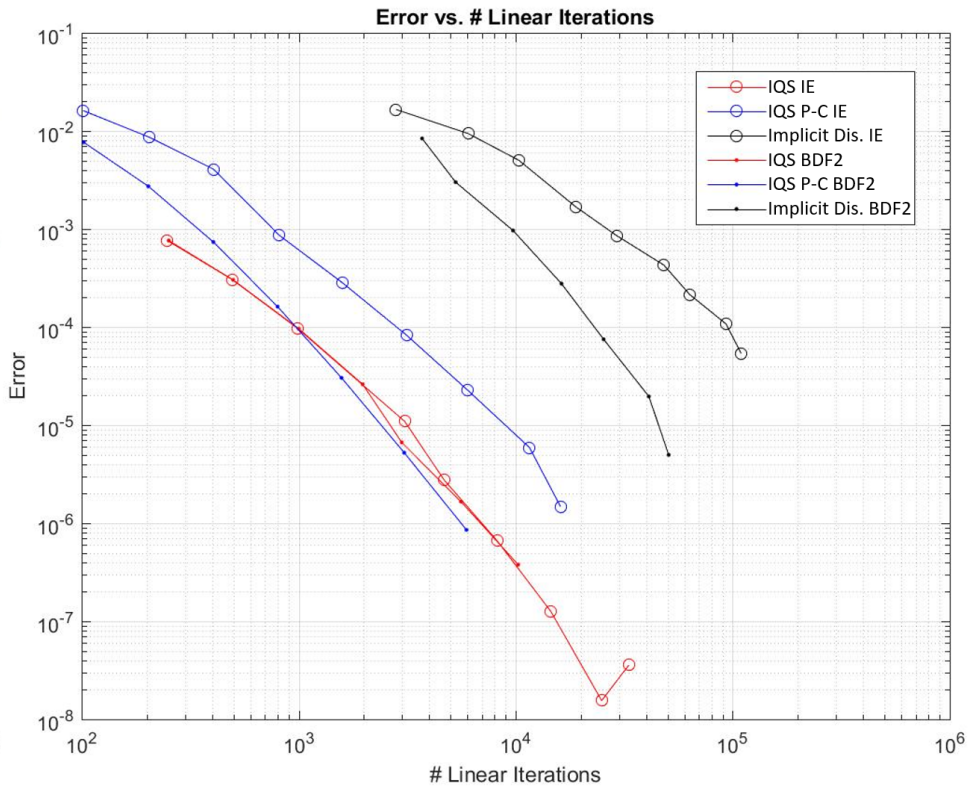


Figure: TWIGL Peak Power Profile

TWIGL time step error convergence





TWIGL with Time Adaptation

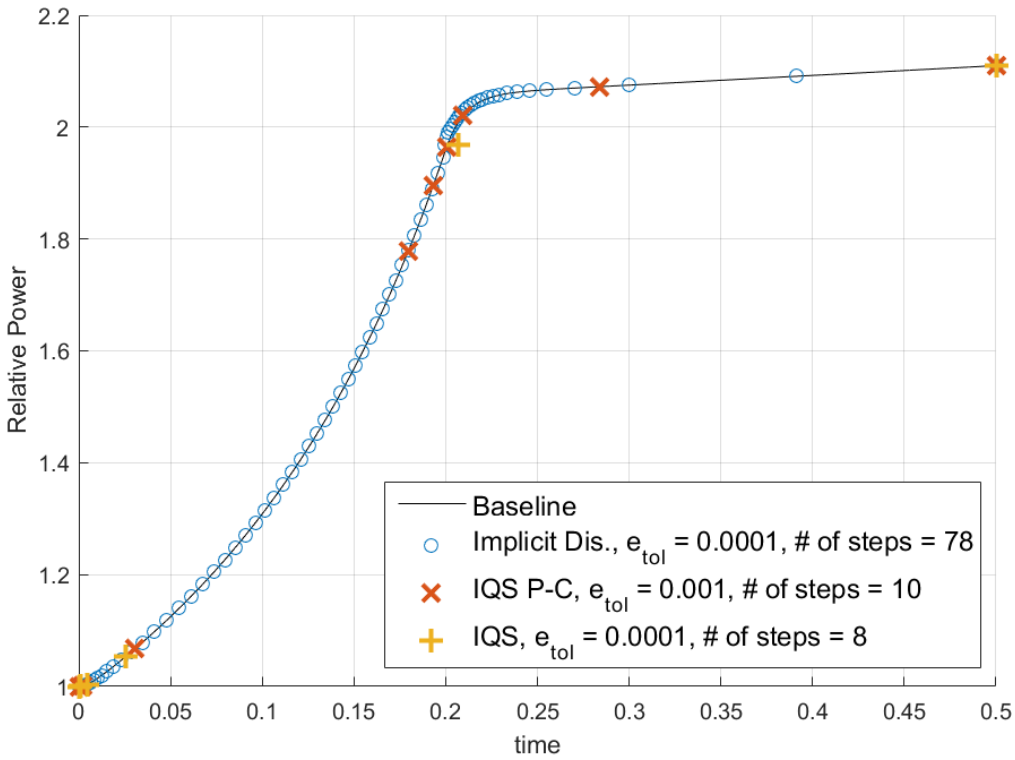
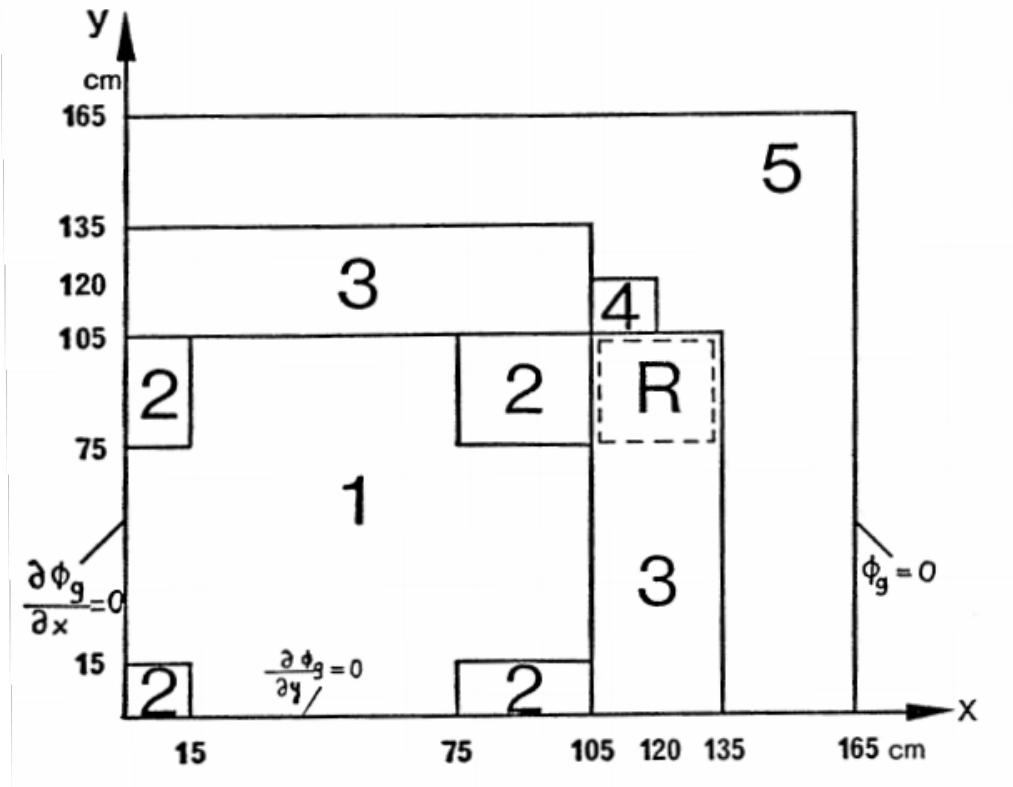
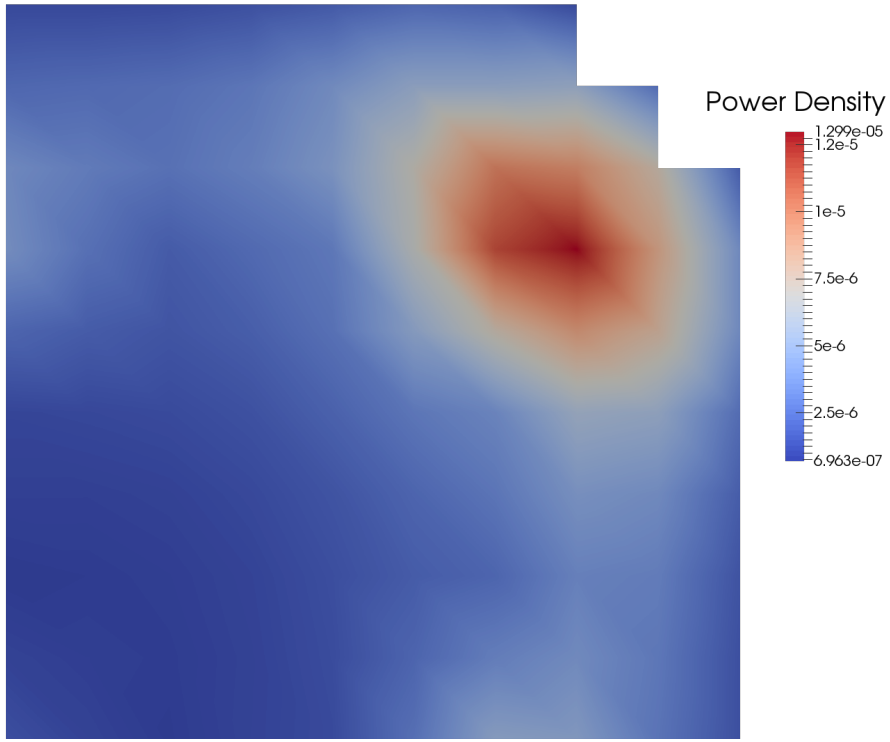


Figure: TWIGL Power Profile

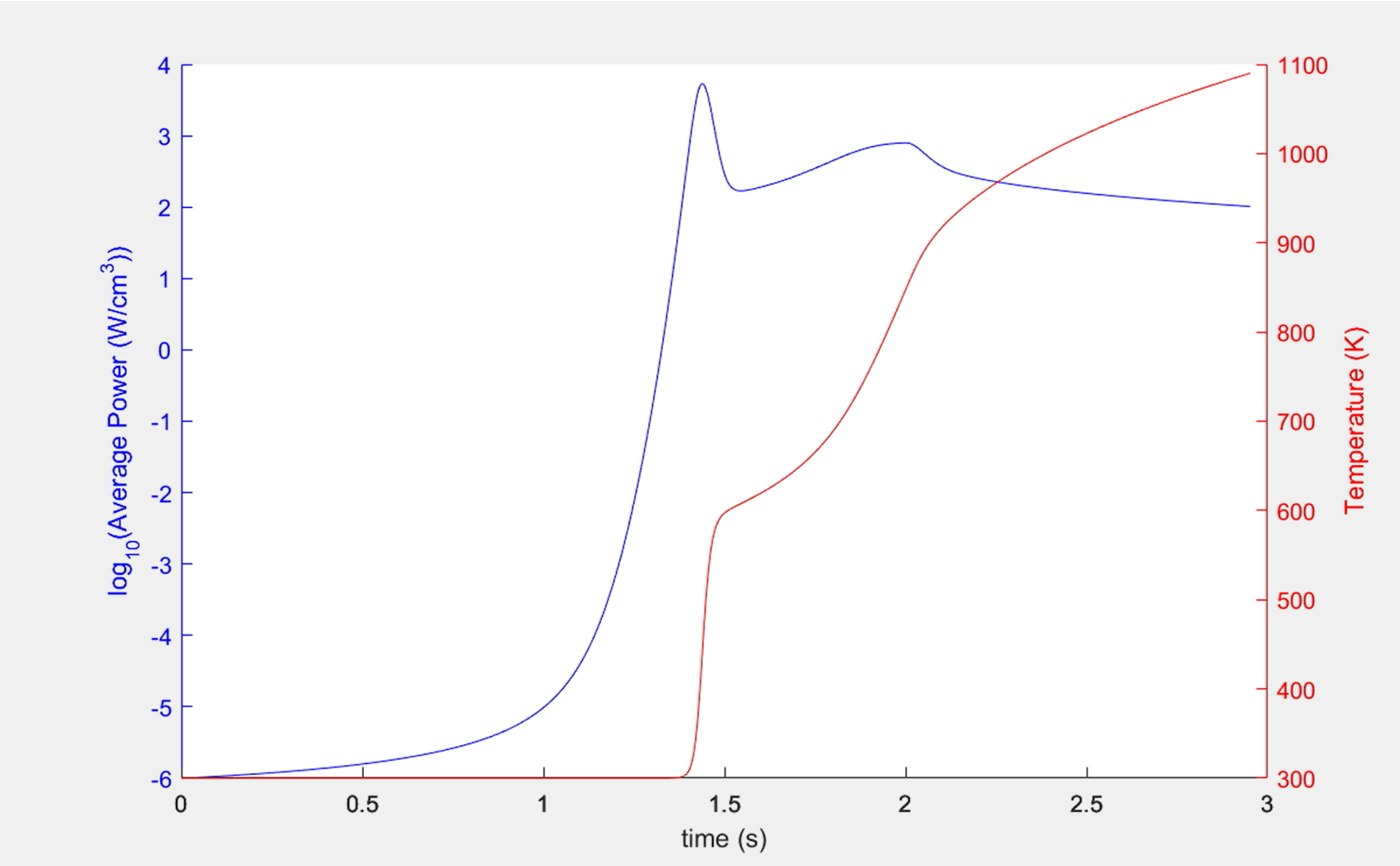
Implicit Discretization				
Test	e_{tol}	Error	Steps	Solves
1	0.05	0.00012677	9	29
2	0.01	3.5555e-05	11	35
3	0.005	4.0364e-05	11	31
4	0.001	0.00294822	33	122
5	0.0005	0.00099778	39	131
6	0.0001	0.00019510	78	236
7	5.0e-05	0.00018372	112	342
8	1.0e-05	8.0564e-05	263	794
IQS				
Test	e_{tol}	Error	Steps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

LRA Benchmark





LRA Power and Temperature Profile



LRA Time Step Error Convergence

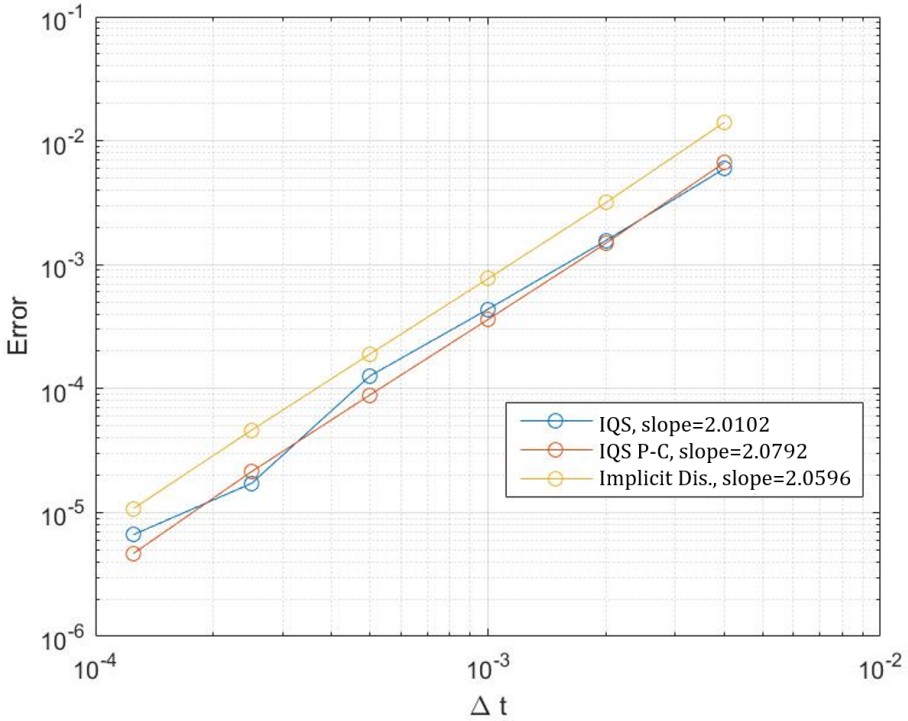


Figure: Only one temperature update per macro step

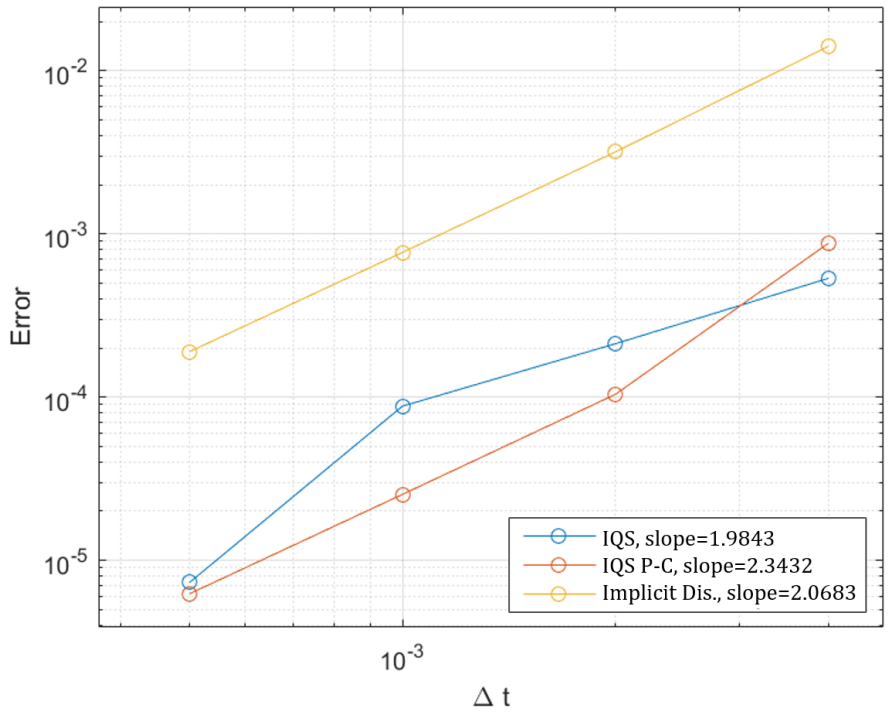
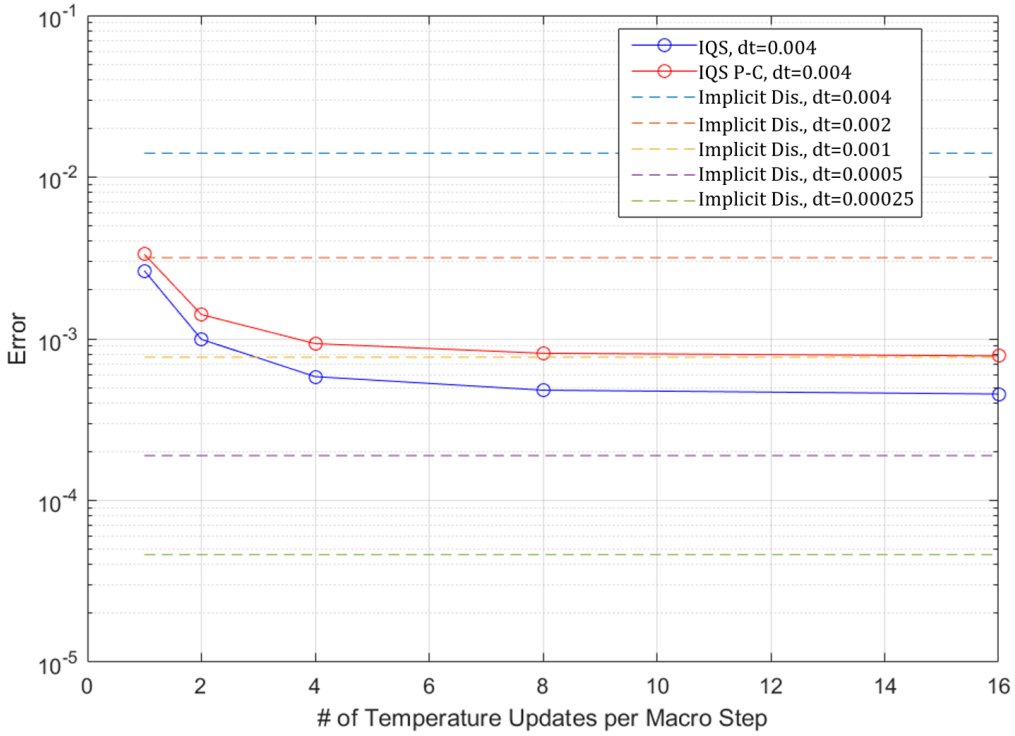


Figure: Five temperature updates per macro step

Analysis on Temperature Updates



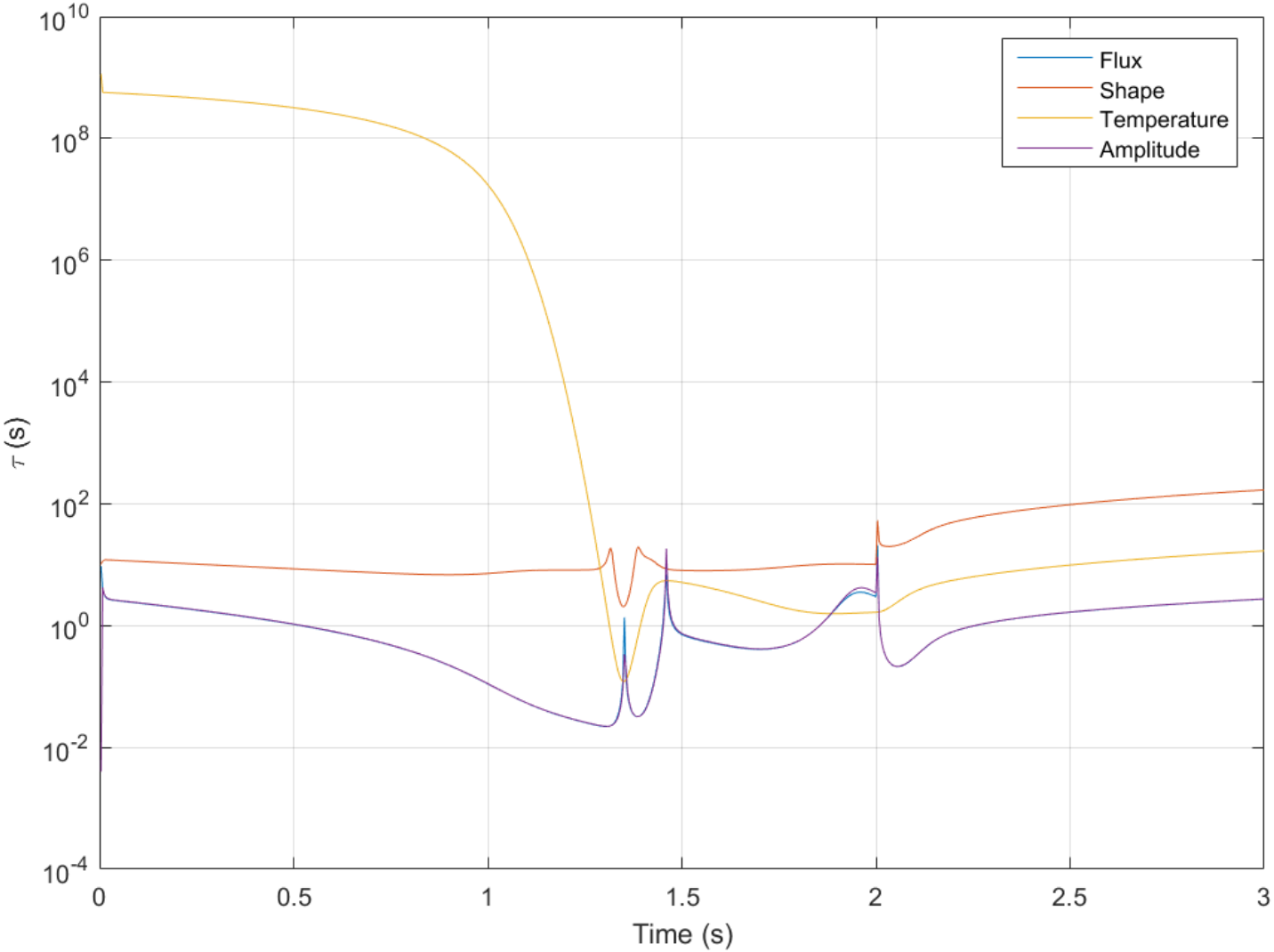
Implicit Discretization				
Run	Δt	Error	Runtime (hr)	Linear Iter.
1	4.0e-3	1.407e-2	4.11	7.13e4
2	2.0e-3	3.174e-3	6.01	9.49e4
3	1.0e-3	7.690e-4	10.38	1.45e5
4	5.0e-4	1.892e-4	21.91	2.08e5
5	2.5e-4	4.590e-5	25.23	3.16e5

IQS				
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%

IQS P-C				
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%

* runtime difference from $\Delta t = 0.004$ implicit dis.

LRA Dynamical Time Scale Analysis



LRA with Time Adaptation

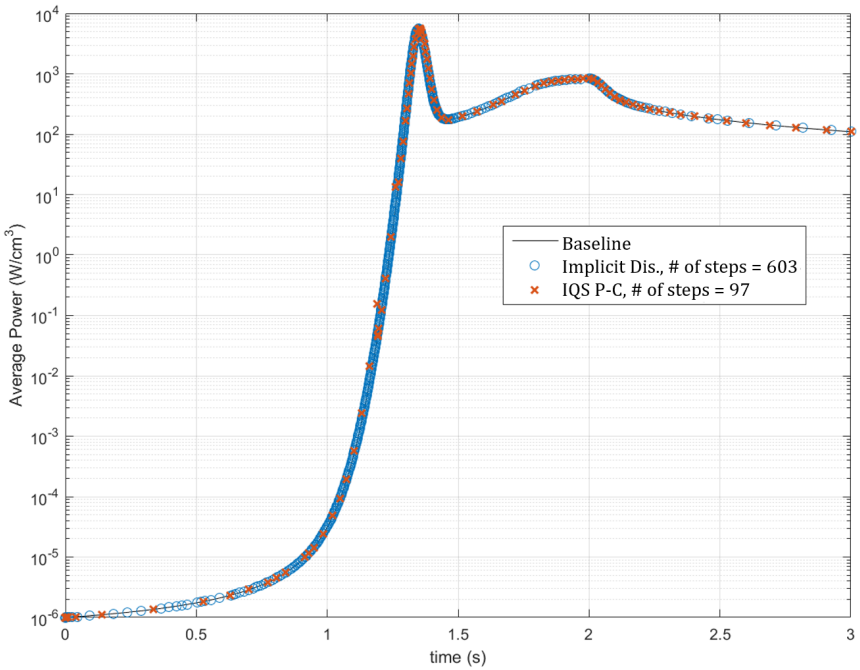


Figure: LRA Power Profile

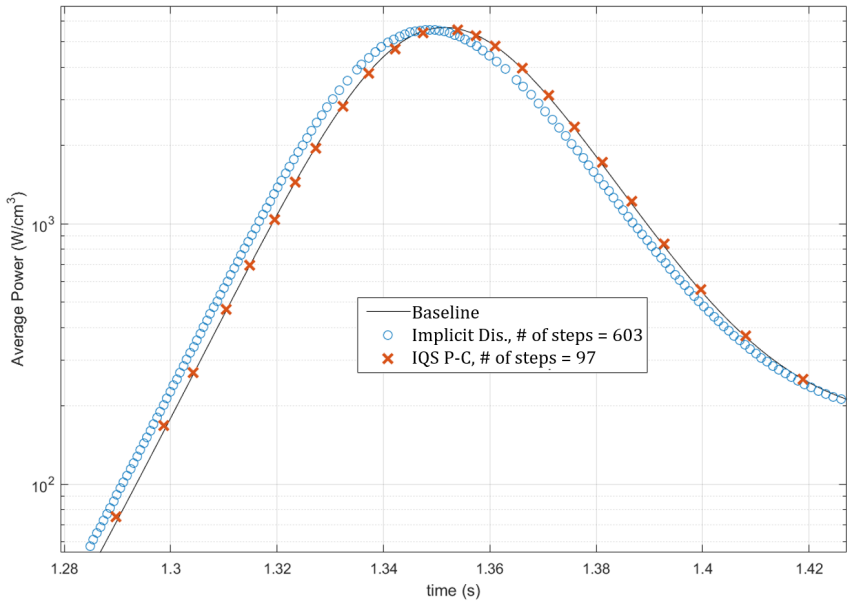


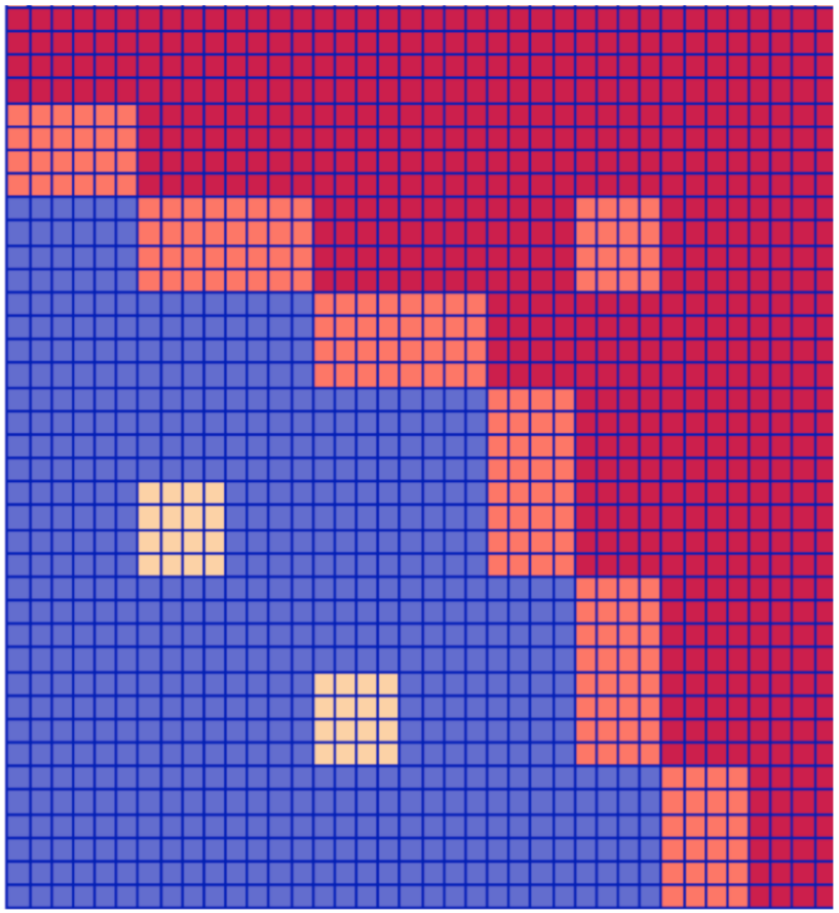
Figure: LRA Peak Power Profile

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

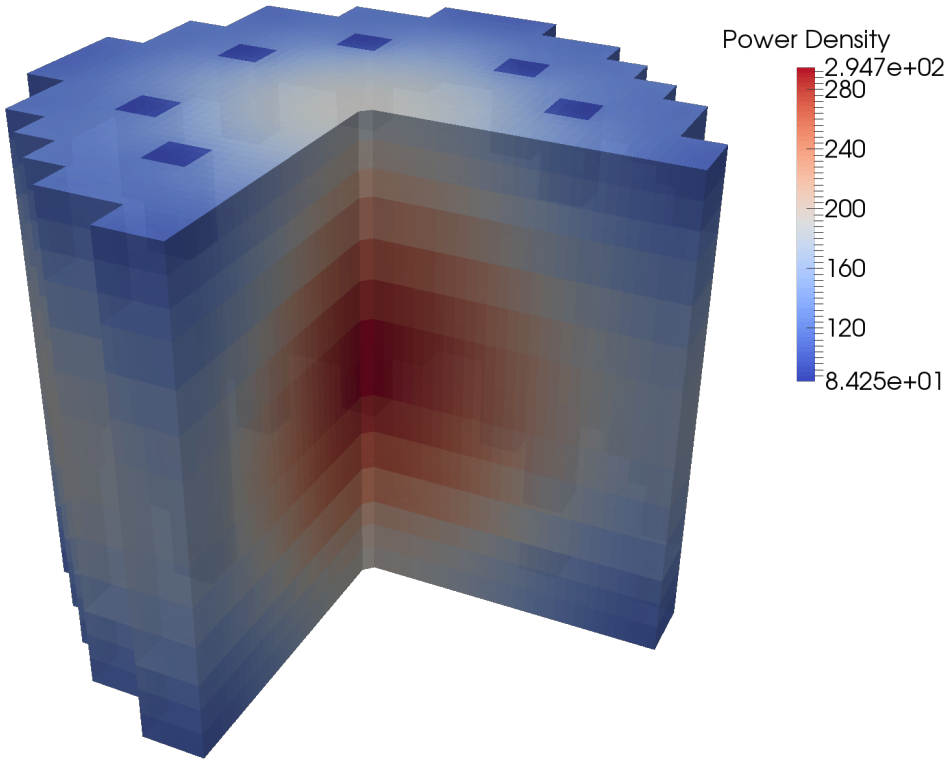
TREAT: Transient-15 Example

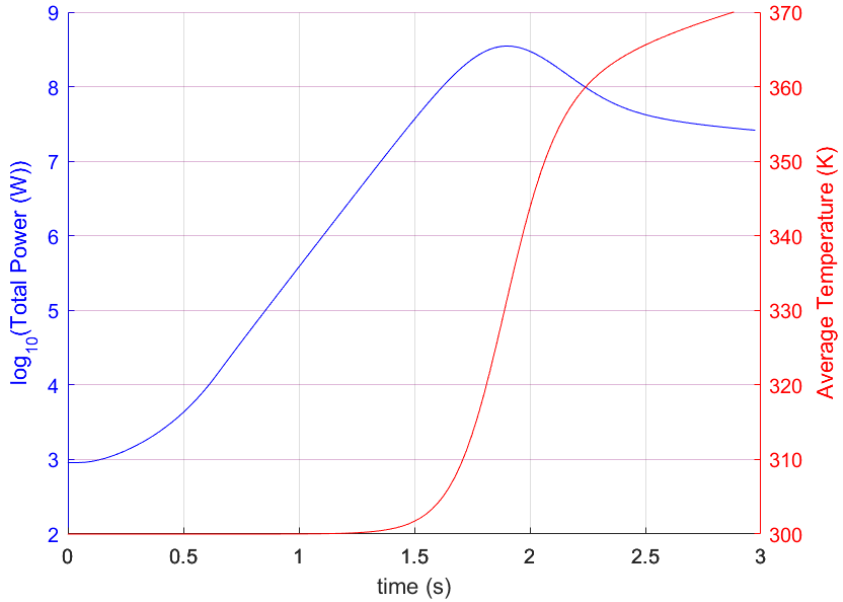
	A	B	C	D	E	F	G	H	J	K	L	M	N	O	P	R	S	T	U
1	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
2	A	A	A	A	A	A	A	A	Z	Z	Z	A	A	A	A	A	A	A	A
3	A	A	A	A	A	A	Z	Z	F	F	F	Z	Z	A	A	A	Z	A	A
4	A	A	A	A	A	Z	F	F	F	F	F	F	F	Z	Z	A	A	A	A
5	A	A	A	A	Z	F	F	F	F	F	F	F	F	F	F	Z	A	A	A
6	A	A	A	Z	F	F	F	C	F	F	F	C	F	F	F	Z	A	A	A
7	A	A	Z	F	F	F	F	F	F	F	F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

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



Transient-15 Power Profile





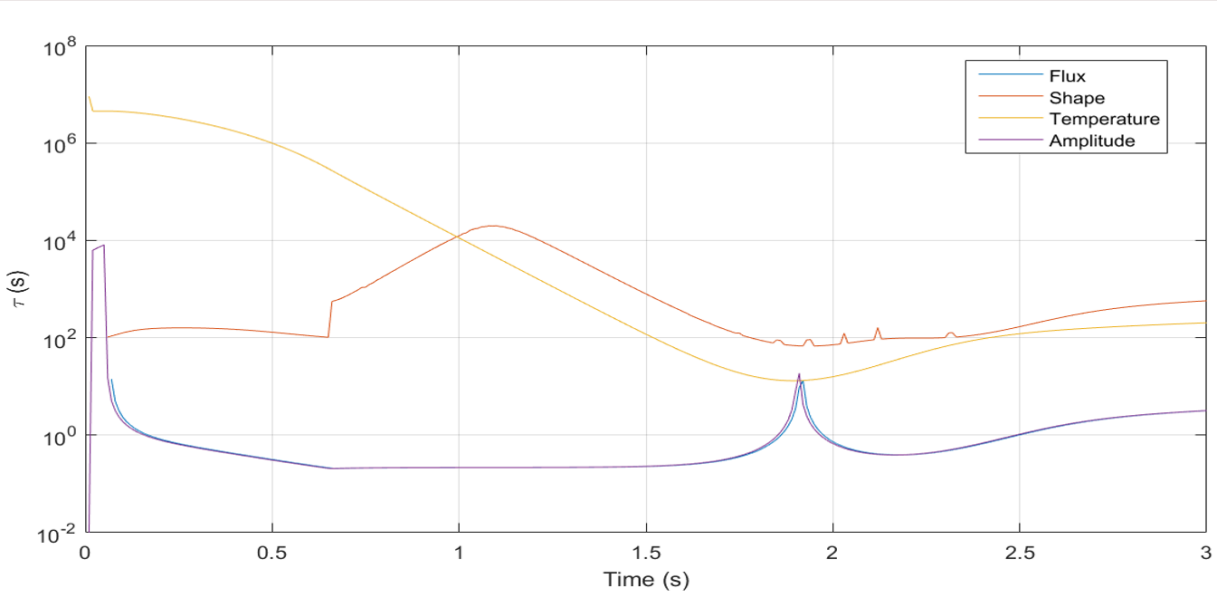
Transient-15 Results

Accuracy and Runtime

Method	No. of Steps	% Increase Runtime*	Max Power Error
Implicit Dis.	300	—	7.875e-4
IQS	300	-11.9%	8.385e-5
IQS (5 updates)	300	49.7%	3.687e-5
IQS P-C	300	-2.1%	7.527e-4
IQS P-C (5 updates)	300	26.5%	1.227e-4

* difference in runtime from implicit discretization

Dynamical Time Scale



Transient-15 with Time Adaptation

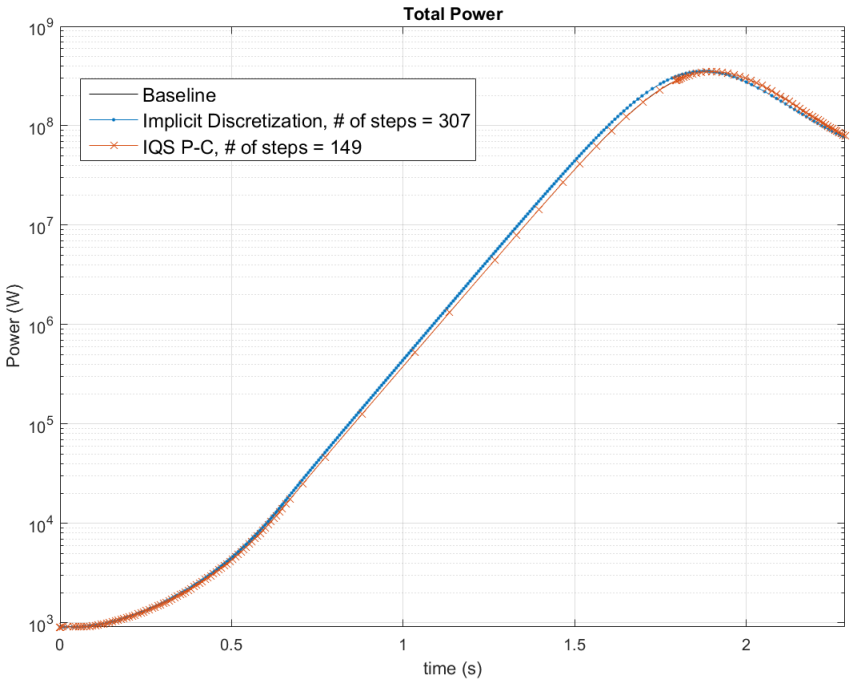


Figure: TREAT Power Profile

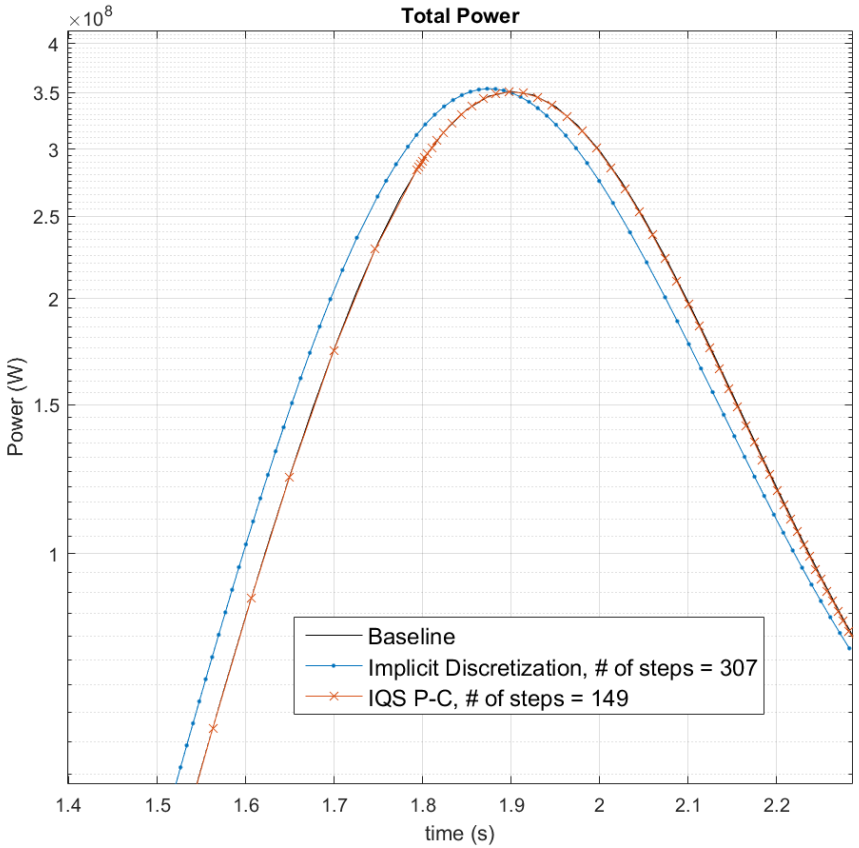


Figure: TREAT Peak Power Profile

Conclusions

IQS Iteration

IQS was testing with fixed point itertion used the following criteria:

1. L^{∞} norm of the change in shape between iterations
2. L^{2} norm of the change in shape between iterations
3. Difference in reactivity between iterations
4. Difference in amplitude between iterations
5. IQS uniqueness consistency criteria

- 1-4 had equivalent behavior
- 5 required analytical integration of precursors for reasonable error convergence

Conclusions (cont.)

IQS Time Step Error Convergence

- Demonstrated proper convergence through order 4
- Required higher order interpolation of PRKE parameters for higher order convergence
- Effectiveness of error convergence demonstrated through time adaptation
- Marginal improvement over implicit discretization for "difficult" 1D problem
- Significant improvement for TWIGL and LRA

IQS with Temperature Feedback

- IQS was successfully implemented into the Rattlesnake/MOOSE framework
- Added intermediate time scale loop for temperature
- Incorporation of temperature into the quasi-static process significantly improved computational efficiency
- Dynamical time scale analysis demonstrated proof-of-concept for variant time scales

Recommendations for Future Research

Multiphysics

Test IQS with other reactor physics, including:

- More complex temperature feedback
- Thermal Hydraulics
- Fuel Mechanics

Transport

Apply IQS to other transport models (other Rattlesnake action systems):

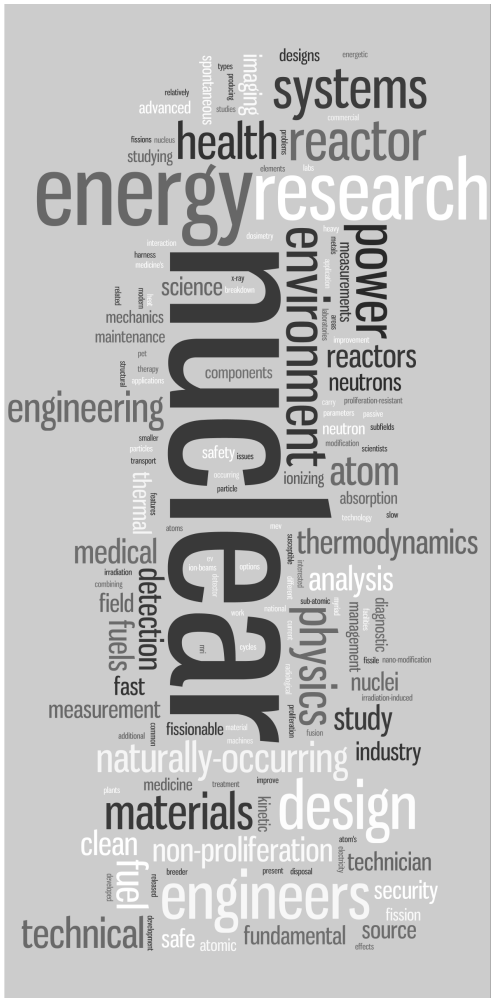
- DFEM-Diffusion
- SAAF-Sn
- SAAF-Pn
- LS-Sn

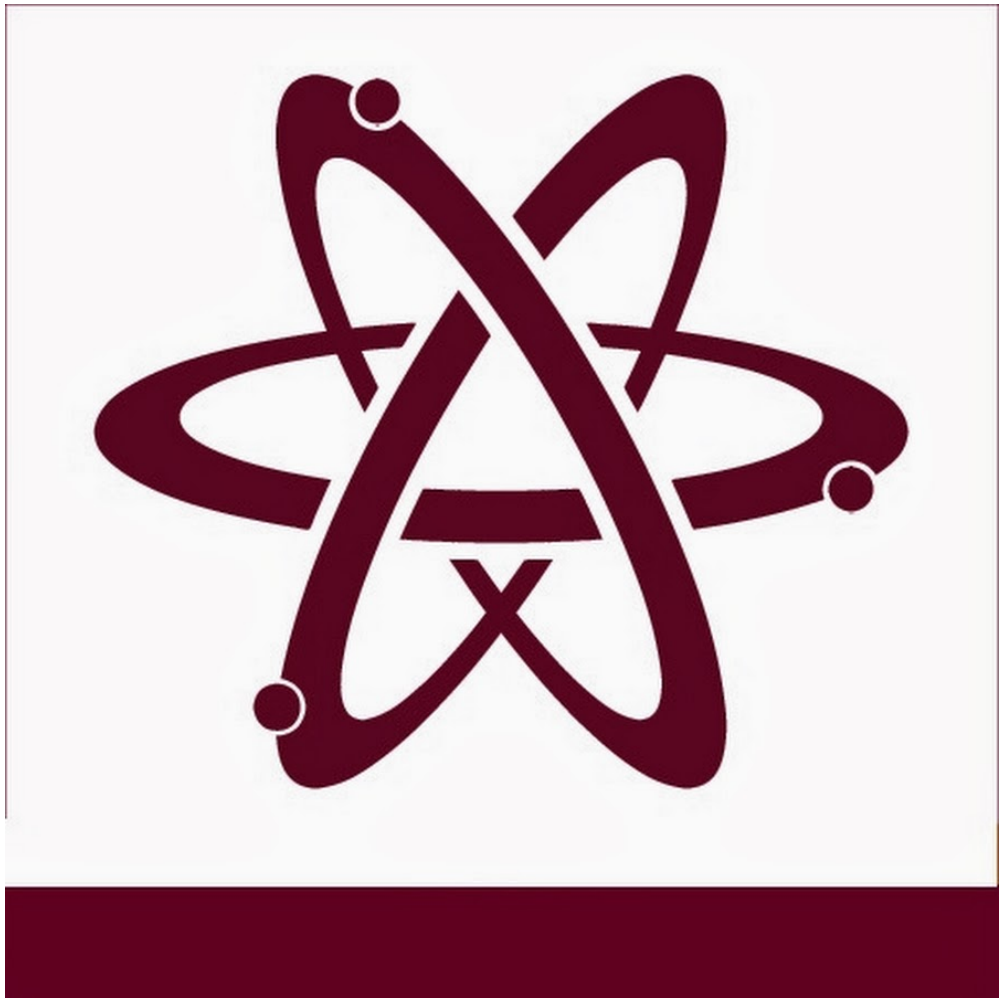
PJFNK

- Investigate PJFNK iteration of IQS
- Optimize IQS preconditioning

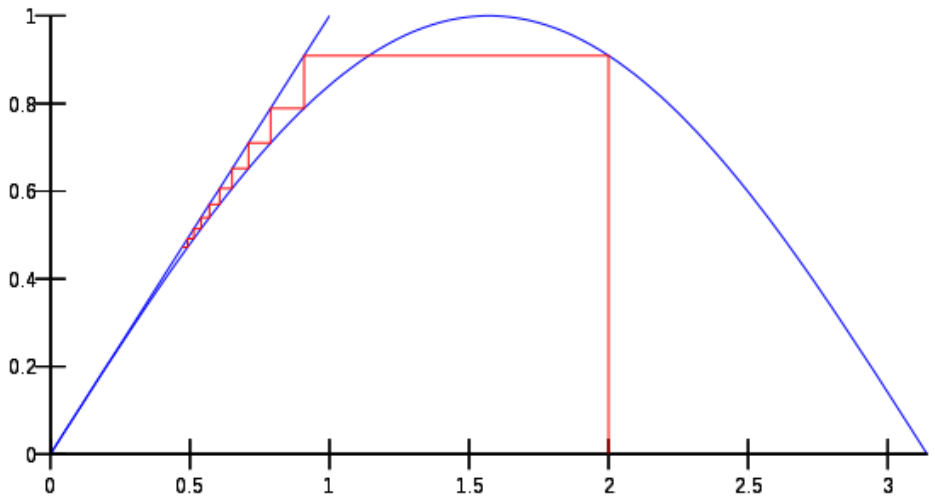
Questions?

Thank You & Gig Em





Fixed-point iteration



IQS iteration approach

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

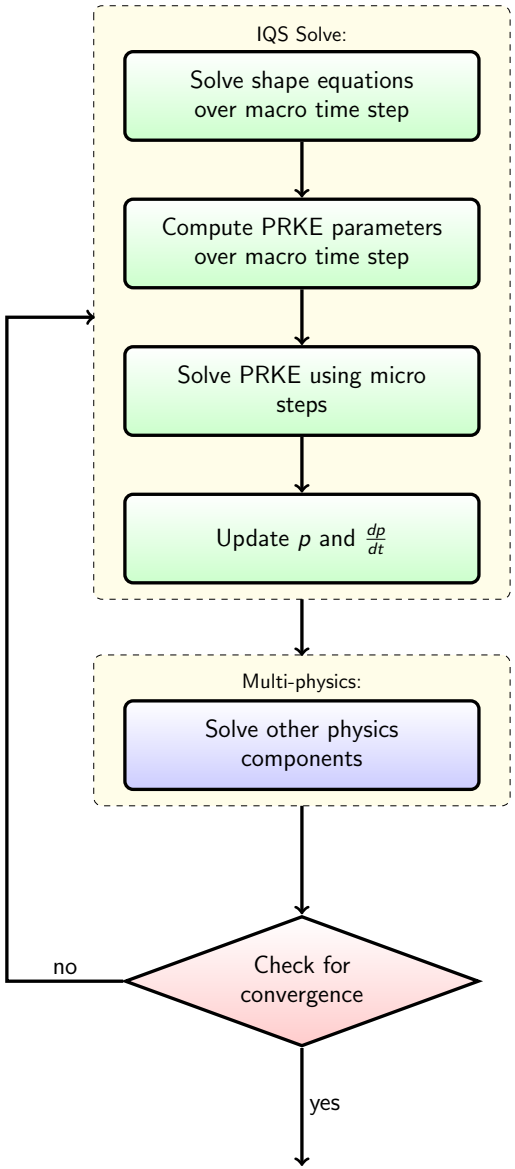
Step 3: Solve the PRKE on micro steps over the entire macro step

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

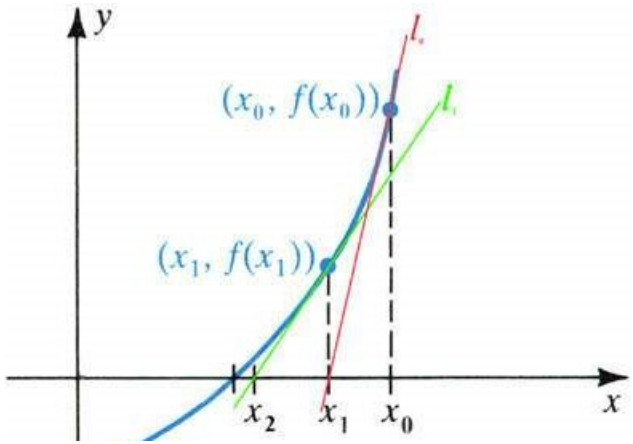
Step 5: Check if the shape solution has converged:

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

Fixed-point iteration programming logic



Newton iteration



Jacobian-Free Newton-Krylov Method

- Newton system:

$$J\delta\varphi = F(\varphi, p, t) - A(\varphi, p)\varphi \equiv -R(\varphi, p)$$

- Where J is the Jacobian and defined as:

$$J_{ij} = \partial R_i / \partial \varphi_j$$

- For IQS, the Jacobian is impossible to define so it must be approximated:

$$J\delta\varphi \approx [R(\varphi + \epsilon\delta\varphi, p') - R(\varphi, p)]/\epsilon$$

- The resulting system is solved using GMRES iteration

Time Discretization Schemes

General Equation Form

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

Theta Method

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

- Implicit Euler: $\theta = 1$ (First Order)
- Explicit Euler: $\theta = 0$ (First Order)
- Crank-Nicolson: $\theta = 1/2$ (Second Order)

Backward Difference Formula (BDF)

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

Order (k)	α_1^k	α_2^k	α_3^k	α_4^k	α_5^k
1	1	1			
2	-1/3	4/3	2/3		
3	2/11	-9/11	18/11	6/11	
4	-3/25	16/25	-36/25	48/25	12/25

Time Discretization Schemes (cont.)

Singly-Diagonally-Implicit Runge-Kutta (SDIRK)

For a general differential equation $dy/dt = f(t, y)$:

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

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

Butcher Tableu:

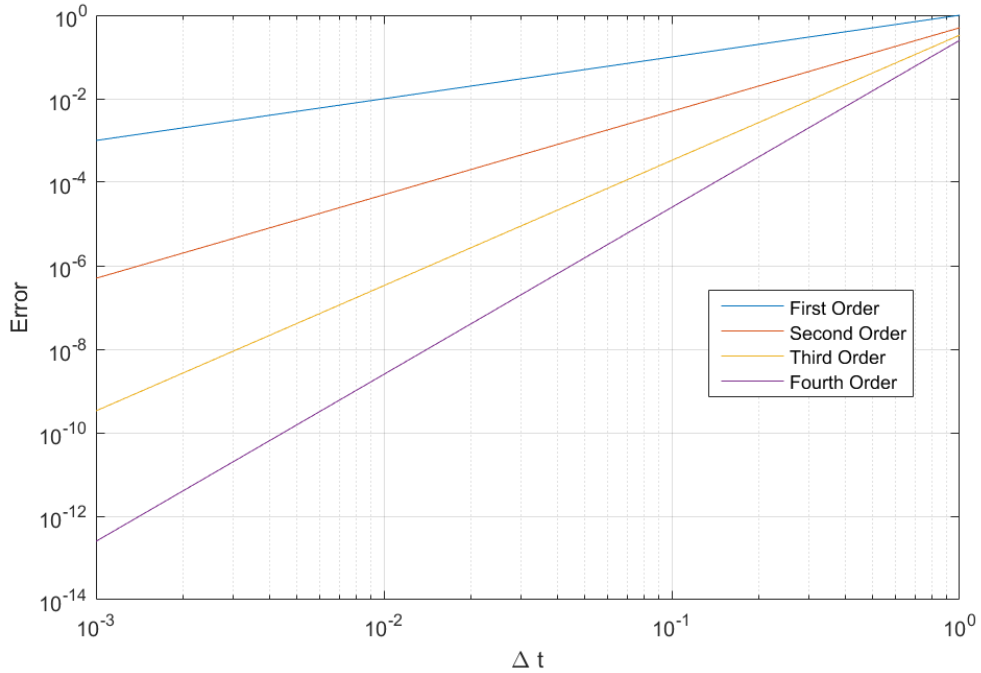
c_1	a_{11}			
c_2	a_{21}	a_{22}		
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s

SDIRK33:

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 $1 - 9\lambda + 18\lambda^2 - 6\lambda^3 = 0$

Time Step Error Convergence



- Validation through error convergence is vital for error quantification and time adaptation
- Analysis involves:
  1. Running simulation at variant time step sizes
  2. Determining the error in power at time of interest based on baseline (very small time step) calculation
  3. Using least-squares fit to determine the slope of error vs. Δt on log scale

Time Adaptation Theory

Motivation

- The concept of time adaptation is to have the behavior of some aspect of the evaluation determine the size of the time step.
- The computational efficiency of IQS is best demonstrated when time adaptation is applied.
- Step doubling adaptation was chosen because it is relatively simple and it utilizes the behavior of the solution to determine step size.

Local Truncation Error

We can estimate the local truncation error of the latest solve with a Taylor series expansion:

$$\|LTE_n\|_{L^2} = \Delta t_n^{p+1} \left\| \frac{\phi_n^{(p+1)}}{(p+1)!} + \Delta t_n \frac{\phi_n^{(p+2)}}{(p+2)!} + ... \right\|_{L^2}$$

Where p is the time discretization method's order and ϕ_n is the solution at time $= t_n$. Δt_n was the latest solves time step and Δt_{n+1} is the next solves time step that has a desired error $\|LTE_{n+1}\|_{L^2}$. It can be

Step Doubling Theory

New Step Size

Using the definitions of the local errors:

$$\Delta t_{n+1}^{p+1} \simeq \Delta t_n^{p+1} \theta \frac{\|LTE_{n+1}\|_{L^2}}{\|LTE_n\|_{L^2}}$$

Where $\theta \equiv 1 + O(\Delta t_n)$. $\|LTE_{n+1}\|_{L^2}$ is some user defined relative error tolerance (e_{tol}) and $e_n \equiv \frac{\theta}{\|LTE_n\|_{L^2}}$ is a method's approximation to the last step's local error. Therefore in practice:

$$\Delta t_{new} = \Delta t_{old} \left[\frac{e_{tol}}{e_n}\right]^{1/(p+1)}$$

Step Doubling

Step doubling approximates the local error (e_n) by taking the difference in the local error of a solution with Δt ($\phi_{\Delta t}$) and $\Delta t/2$ ($\phi_{\Delta t/2}$):

$$e_n = \frac{\left\|\phi_{\Delta t/2} - \phi_{\Delta t}\right\|_{L^2}}{\max\left(\left\|\phi_{\Delta t/2}\right\|_{L^2}, \|\phi_{\Delta t}\|_{L^2}\right)}$$

MATLAB Kinetics Prototype Code (cont.)

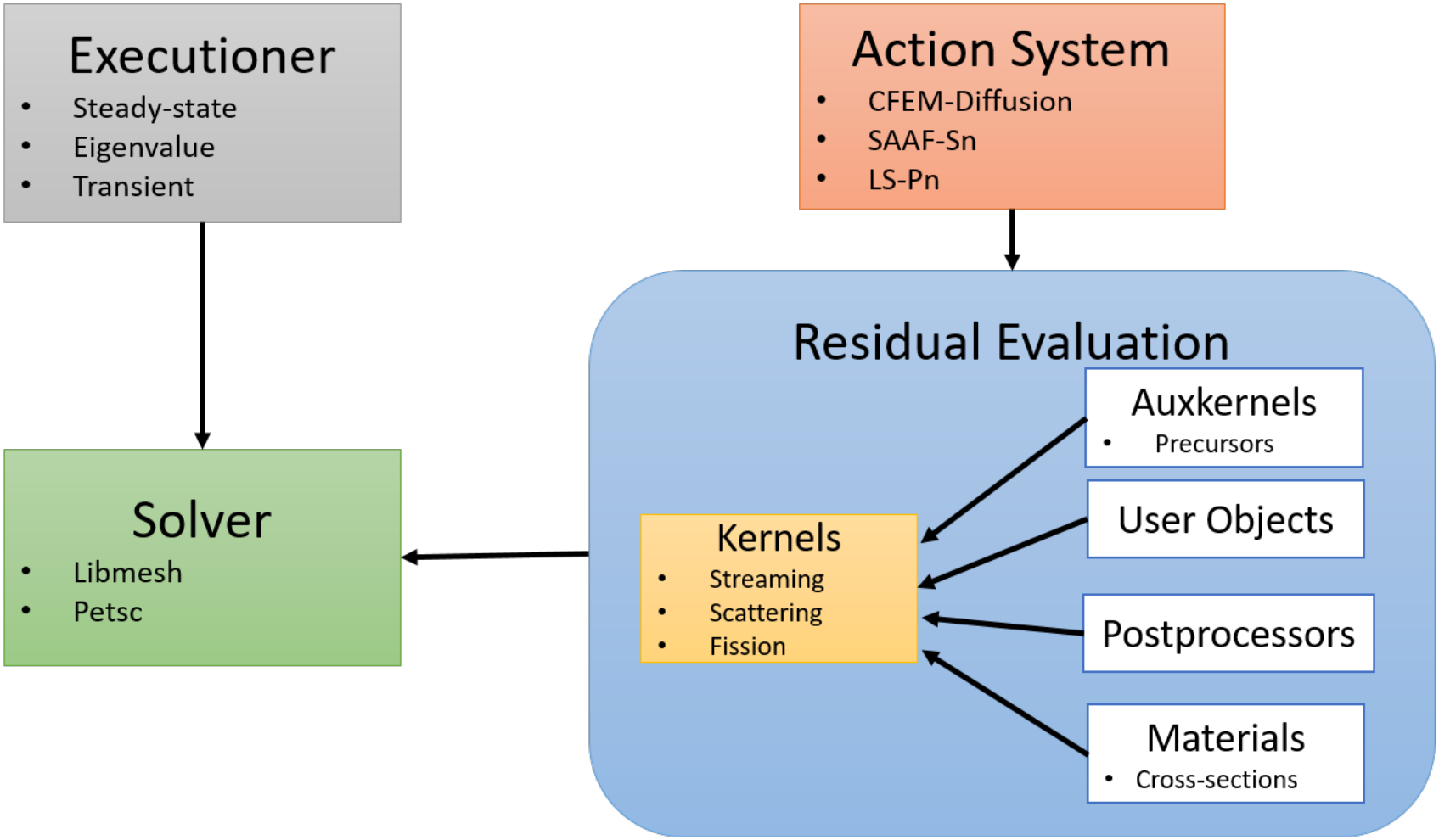
Time Discretization Schemes

- Embedded Rung-Kutta time adaptation with ode15s
- Implicit Euler
- First-fourth order BDF
- Third order SDIRK (SDIRK33)

IQS Process

1. Intial steady-state eigenvalue evaluation
2. ode15s baseline calculation for error comparison
3. PRKE parameter calculation at macro steps
4. ode15s PRKE evaluation using interpolated PRKE parameters for amplitude
5. Build left- and right-hand-side system for shape
6. Evaluate shape using backslash operation
7. Iterate amplitude and shape until convergence
8. Evaluate precursors

Rattlesnake Structure



IQS Implementation in Rattlesnake

IQS Components in Rattlesnake

- IQS Executioner
  - Convergence criteria for Picard iteration:

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

  - Evaluates PRKE using implicit Euler, Crank-Nicolson, or SDIRK33 with step doubling adaptation for $\frac{1}{p}\frac{dp}{dt}$ term
- PRKE Parameter Postprocessors
  - Performs integrations for PRKE parameters
  - Residuals from kernels are saved for $\rho - \bar{\beta}$ integration
- PRKE User Object
  - Gathers postprocessor values
- IQS Removal Kernel
  - Removal kernel for $\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}\varphi^g$ term
- Auxkernels
  - Precursor auxkernel with analytical integration
  - Temperature auxkernel with analytical integration

IQS Implementation in Rattlesnake (cont.)

IQS Kernels

$$\frac{1}{v^g}\frac{\partial \varphi^g}{\partial t} = \underbrace{\frac{\chi_p^g}{k_{eff}}\sum_{g'=1}^{G}(1-\beta)\nu^{g'}\Sigma_f^{g'}\varphi^{g'}}_{FluxKernel} + \underbrace{\sum_{g'\neq g}^{G}\Sigma_s^{g'\to g}\varphi^{g'}}_{FluxKernel} - \underbrace{\left(-\vec{\nabla}\cdot D^g\vec{\nabla}\right)\varphi^g}_{FluxKernel} - \underbrace{\Sigma_r^g\varphi^g}_{FluxKernel}$$

$$\underbrace{-\frac{1}{v^g}\boxed{\overbrace{\frac{1}{p}\frac{dp}{dt}}^{FromExecutioner}}\varphi^g}_{IQSKernel} + \underbrace{\frac{1}{p}\sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i}_{ModifiedFluxKernel}$$