Improved Quasi-Static Method

IQS Method Implemention for CFEM Diffusion in Rattlesnake

Zachary M. Prince[†], **Jean C. Ragusa**[†], **Yaqi Wang**[⋆]

[†]Department of Nuclear Engineering

Texas A&M University, College Station, TX, USA

[⋆]Idaho National Laboratory

zachmprince@tamu.edu, jean.ragusa@tamu.edu, yaqi.wang@inl.gov

June 7, 2016

1 Overview

The improved quasi-static (IQS) method is a transient spatial kinetics method that involves factorizing flux into space- and time-dependent components. These components include the flux's power and shape. Power is 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 shape may not require computation at every time step, invoking the quasi-static nature.

In this Section, we recall the equations for the IQS method, starting from the standard multigroup diffusion equations written below:

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

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

with

$$\beta = \sum_{i=1}^{I}\beta_i \tag{3}$$

Factorization is the most important step in the derivation of the IQS method. The factorization approach leads to a decomposition of the multigroup flux into the product of a time-dependent amplitude (p) and a space-/time-dependent multigroup shape (φ^g):

$$\phi^g(\vec{r},t) = p(t)\varphi^g(\vec{r},t) \tag{4}$$

Then the flux and precursor equations become:

$$\begin{aligned}\frac{1}{v^g}\left(\frac{dp}{dt}\varphi^g + p\frac{\partial\varphi^g}{\partial t}\right) =&\frac{\chi_p^g}{k_{eff}}(1-\beta)\sum_{g'=1}^{G}\nu^{g'}\Sigma_f^{g'}p\varphi^{g'} - \left(-\nabla\cdot D^g\nabla + \Sigma_r^g\right)p\varphi^g \\ &+\sum_{g'\neq g}^{G}\Sigma_s^{g'\rightarrow g}p\varphi^{g'} + \sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i \ , \quad 1\leq g\leq G \end{aligned} \tag{5}$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{k_{eff}} \sum_{g=1}^{G} \nu^g \Sigma_f^g p \varphi^g - \lambda_i C_i \ , \quad 1 \leq i \leq I \tag{6}$$

PRKE formulation (amplitude equations):

$$\begin{aligned} \sum_{g=1}^{G} \int_D \left[\frac{1}{v^g} \left(\frac{dp}{dt} \varphi^g + p \frac{\partial \varphi^g}{\partial t} \right) \right] w^g d^3 r = \\ \sum_{g=1}^{G} \int_D \left[\frac{\chi_p^g}{k_{eff}} (1-\beta) \sum_{g'=1}^{G} \nu^{g'} \Sigma_f^{g'} p \varphi^{g'} - \left(-\nabla \cdot D^g \nabla + \Sigma_r^g \right) p \varphi^g \right] w^g d^3 r \\ + \sum_{g=1}^{G} \int_D \left[\sum_{g' \neq g}^{G} \Sigma_s^{g' \to g} p \varphi^{g'} + \sum_{i=1}^{I} \chi_{d,i}^g \lambda_i C_i \right] w^g d^3 r \ , \quad 1 \leq g \leq G \end{aligned} \tag{7}$$

$$\int_D \frac{dC_i}{dt} \sum_{g=1}^{G} \chi_{d,i}^g w^g d^3 r = \int_D \frac{\beta_i}{k_{eff}} \left[\sum_{g=1}^{G} \left(\sum_{g'=1}^{G} \nu^{g'} \Sigma_f^{g'} \varphi^{g'} - \lambda_i C_i \right) \chi_{d,i}^g w^g \right] d^3 r \ , \quad 1 \leq i \leq I \tag{8}$$

It can be shown that the most appropriate weighting function (w^g) is the initial adjoint flux (ϕ^{*g}). For brevity, the following definition will be applied: $\int_D \phi^{*g}(\vec{r}) f(\vec{r}) d^3 r = (\phi^{*g}, f)$

$$\begin{aligned} \frac{dp}{dt} \sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right) + p \frac{d}{dt} \sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right) = \\ p \sum_{g=1}^{G} \left(\phi^{*g}, \frac{\chi_p^g}{k_{eff}} (1-\beta) \sum_{g'=1}^{G} \nu^{g'} \Sigma_f^{g'} \varphi^{g'} + \sum_{g' \neq g}^{G} \Sigma_s^{g' \to g} \varphi^{g'} - \left(-\nabla \cdot D^g \nabla + \Sigma_r^g \right) \varphi^g \right) \\ + \sum_{i=1}^{I} \sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g \lambda_i C_i) \end{aligned} \tag{9}$$

$$\frac{d}{dt} \sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g C_i) = \frac{1}{k_{eff}} \sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g \beta_i \sum_{g'=1}^{G} \nu^{g'} \Sigma_f^{g'} \varphi^{g'}) p - \sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g \lambda_i C_i) \quad 1 \leq i \leq I \tag{10}$$

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

$$\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right) = constant \tag{11}$$

Therefore the PRKE formulation reduces to:

$$\begin{aligned} \frac{dp}{dt} = & \frac{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{\chi_p^g}{k_{eff}} (1-\beta) \sum_{g'=1}^{G} \nu^{g'} \Sigma_f^{g'} \varphi^{g'} + \sum_{g' \neq g}^{G} \Sigma_s^{g' \to g} \varphi^{g'} - \left(-\nabla \cdot D^g \nabla + \Sigma_r^g \right) \varphi^g \right)}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right)} p \\ & + \sum_{i=1}^{I} \frac{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g \lambda_i C_i)}{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g C_i)} \xi_i \end{aligned} \tag{12}$$

$$\frac{d\xi_i}{dt} = \frac{1}{k_{eff}} \frac{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g \beta_i \sum_{g'=1}^{G} \nu^{g'} \Sigma_f^{g'} \varphi^{g'})}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right)} p - \frac{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g \lambda_i C_i)}{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g C_i)} \xi_i \quad 1 \leq i \leq I \tag{13}$$

Where:

$$\xi_i = \frac{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^g C_i)}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g \right)} \tag{14}$$

It is convenient to define the effective reactivity, delay-neutron fraction, and delayed-neutron precursor decay constant:

$$\frac{\rho-\bar{\beta}}{\Lambda}=\frac{\sum_{g=1}^{G}\left(\phi^{*g},\frac{\chi_p^g}{k_{eff}}(1-\beta)\sum_{g'=1}^{G}\nu^{g'}\Sigma_f^{g'}\varphi^{g'}+\sum_{g'\neq g}^{G}\Sigma_s^{g'\rightarrow g}\varphi^{g'}-\left(-\nabla\cdot D^g\nabla+\Sigma_r^g\right)\varphi^g\right)}{\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi^g\right)} \tag{15}$$

$$\frac{\bar{\beta}}{\Lambda}=\sum_{i=1}^{I}\frac{\bar{\beta}_i}{\Lambda}=\sum_{i=1}^{I}\frac{1}{k_{eff}}\frac{\sum_{g=1}^{G}(\phi^{*g},\chi_{d,i}^g\beta_i\sum_{g'=1}^{G}\nu^{g'}\Sigma_f^{g'}\varphi^{g'})}{\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi^g\right)} \tag{16}$$

$$\bar{\lambda}_i=\frac{\sum_{g=1}^{G}(\phi^{*g},\chi_{d,i}^g\lambda_i C_i)}{\sum_{g=1}^{G}(\phi^{*g},\chi_{d,i}^g C_i)} \tag{17}$$

So:

$$\frac{dp}{dt}=\left[\frac{\rho-\bar{\beta}}{\Lambda}\right]p+\sum_{i=1}^{I}\bar{\lambda}_i\xi_i \tag{18}$$

$$\frac{d\xi_i}{dt}=\frac{\bar{\beta}_i}{\Lambda}-\bar{\lambda}_i\xi_i\quad 1\leq i\leq I \tag{19}$$

Equations (18) and (19) are a formulation of the point reactor kinetics equation (PRKE), but the parameters (Equations (15)-(17)) are dependent on the shape. If the assumption is made that the shape is time-independent, the shape is computed once at the first time step and used for the PRKE parameter evaluation at all other steps. However, if the shape is dependent on time, the shape needs to be computed in transient using equation (5) and (6) in order retain accuracy. Equations (20) and (21) shows the usual form of the shape and precursor equations with amplitude put on the right hand side. Equation (20) is very similar to the multigroup flux equation (1), except the removal cross-section term is augmented by a $\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}$ term and the precursor contribution has a $\frac{1}{p}$ mulitiplier. Equation (21) is very similar to the normal precursor equation (2), except the fission source term is multiplied by p. These differences are crucial for IQS implementation in Rattlesnake.

$$\begin{aligned}\frac{1}{v^g}\frac{\partial\varphi^g}{\partial t}=&\frac{\chi_p^g}{k_{eff}}(1-\beta)\sum_{g'=1}^{G}\nu^{g'}\Sigma_f^{g'}\varphi^{g'}+\sum_{g'\neq g}^{G}\Sigma_s^{g'\rightarrow g}\varphi^{g'}\\&-\left(-\nabla\cdot D^g\nabla+\Sigma_r^g+\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}\right)\varphi^g+\frac{1}{p}\sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i,\quad 1\leq g\leq G\end{aligned} \tag{20}$$

$$\frac{dC_i}{dt}=\frac{\beta_i}{k_{eff}}p\sum_{g=1}^{G}\nu^g\Sigma_f^g\varphi^g-\lambda_i C_i\ ,\quad 1\leq i\leq I \tag{21}$$

Computing this shape can become expensive, especially in two or three dimensions. Subsequently, 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 which is the root of IQS. To visualize:

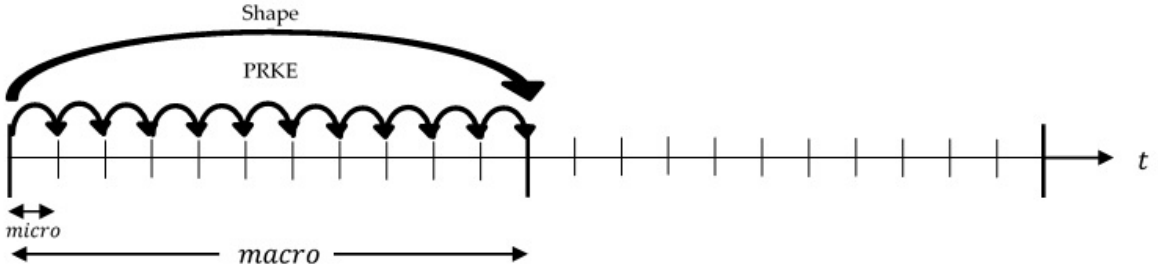


Figure 1: IQS method visualization

Additionally, to improve consistency and accuracy, each macro time step can be iterated so the best shape is used to compute power at the micro time steps. This iteration process must converge the shape such that the uniqueness condition ($\frac{d}{dt}\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi^g\right)=0$) is preserved.

1.1 IQS Predictor-Corrector (IQS P-C)

The Predictor-Corrector version of IQS factorizes the flux and derives the PRKE the same way as the standard version, but the evaluation of the coupled system is different. This version first solves the flux diffusion (represented by Equations 1 and 2) to get a predicted flux. The predicted flux at this step is then converted to shape by rescaling as follows:

$$\varphi_{n+1}^g = \underbrace{\phi_{n+1}^g}_{\text{predicted}} \frac{K_0}{K_{n+1}} \tag{22}$$

Where:

$$K_{n+1} = \sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\phi_{n+1}^g\right) \tag{23}$$

$$K_0 = \sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi_{n+1}^g\right) = \sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\phi_0^g\right) \tag{24}$$

The PRKE parameters are then computed with this shape using Equations 15 - 17 and interpolated over the macro step, then the PRKE is evaluated. With the newly computed amplitude, the shape is rescaled and the corrected flux is evaluated:

$$\underbrace{\phi_{n+1}^g}_{\text{corrected}} = p_{n+1}\times\varphi_{n+1}^g \tag{25}$$

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. The disadvantage is that the method assumes $\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi_{n+1}^g\right)$ is inherently constant.

2 Rattlesnake Implementation

This section will explain how, thus far, IQS has been implemented in Rattlesnake.

2.1 Executioner

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

$$Error_{IQS} = \left|\frac{\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi^{g,n}\right)}{\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi^{g,0}\right)}-1\right| \tag{26}$$

2.2 Action System

IQS defines its uniqueness from its executioner type; however, many changes needed to be made in the Rattlesnake action system in order to support 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, which Rattlesnake is already set up to solve:

$$\frac{1}{v^g}\frac{\partial \varphi^g}{\partial t} = \underbrace{\frac{\chi_p^g}{k_{eff}}\sum_{g'=1}^{G}(1-\beta)\nu^{g'}\Sigma_f^{g'}\varphi^{g'}}_{FluxKernel} + \underbrace{\sum_{g'\neq g}^{G}\Sigma_s^{g'\to g}\varphi^{g'}}_{FluxKernel} - \underbrace{\left(-\nabla\cdot D^g\nabla\right)\varphi^g}_{FluxKernel} - \underbrace{\Sigma_r^g\varphi^g}_{FluxKernel}$$
$$\underbrace{-\frac{1}{v^g}\overbrace{\frac{1}{p}\frac{dp}{dt}}^{FromExecutioner}\varphi^g}_{IQSKernel} + \underbrace{\frac{1}{p}\sum_{i=1}^{I}\chi_{d,i}^g\lambda_i C_i}_{ModifiedFluxKernel} \tag{27}$$

To enable Rattlesnake to solve this equation, another kernel was created that evaluated $\sum_{g=1}^{G}\frac{1}{v^g p}\frac{dp}{dt}\varphi^g$ and added when the IQS executioner is called. Second, four postprocessors were created in order to calculate the PRKE parameters. The parameter calculations were separated by $\frac{\bar{\beta}_i}{\Lambda}$ numerator, $\bar{\lambda}_i$ numerator/denominator, $\frac{\rho}{\Lambda}/\frac{\bar{\beta}}{\Lambda}$ denominator, and $\frac{\rho-\bar{\beta}}{\Lambda}$ numerator. The first three are relatively simple, only relying on material properties and solution quantities. The $\frac{\rho-\bar{\beta}}{\Lambda}$ numerator requires the use of the MOOSE save in feature, which saves the residual from a calculated kernel or boundary contribution in the shape evaluation to an auxiliary variable. Finally, a user object was created to pull together all the postprocessor values and carryout the numerator/denominator divisions that were then passed to the executioner.

2.3 Precursor Integration

This section presents two different time-integration methods to solve coupled IQS shape + precursor equations, recalled below using, for simplicity, a single neutron group and a single precursor group.

$$\frac{1}{v}\frac{\partial\varphi}{\partial t} = \nu\Sigma_f(1-\beta)\varphi - \left(-\nabla\cdot D\nabla + \Sigma_a + \frac{1}{v}\frac{1}{p}\frac{dp}{dt}\right)\varphi + \frac{1}{p}\lambda C \tag{28}$$

$$\frac{dC}{dt} = \beta\nu\Sigma_f\varphi p - \lambda C \tag{29}$$

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 anlaytical, of the precursors ODE. This document will discuss two techniques for solving the precursor equation. First is a time discretization method that is currently being implemented in Rattlesnake. The second is a analytical integration of the precursors, the latter method has proven to be more beneficial for IQS convergence.

2.3.1 Time Discretization using the Theta Method

A fairly simple way to evaluate the precursor equation is to employ the θ-scheme ($0 \leq \theta \leq 1$), explicit when $\theta = 0$, implicit when $\theta = 1$, and Crank-Nicholson when $\theta = 1/2$). Generally, if there is a function u whose governing equation is $\frac{du}{dt} = f(u,t)$, then the θ-discretization is

$$\frac{u^{n+1}-u^n}{\Delta t} = (1-\theta)f(u^n,t) + \theta f(u^{n+1},t)\,. \tag{30}$$

Applying this to the precursor equation:

$$\frac{C^{n+1}-C^n}{\Delta t} = (1-\theta)\beta S_f^n p^n - (1-\theta)\lambda C^n + \theta\beta S_f^{n+1}p^{n+1} - \theta\lambda C^{n+1} \tag{31}$$

Where S_f is the fission source equivalent for shape:

$$S_f^n = (\nu\Sigma_f)^n\varphi^n \tag{32}$$

Rearranging to solve for the precursor at the end of the time step yields

$$C^{n+1} = \frac{1-(1-\theta)\Delta t\lambda}{1+\theta\Delta t\lambda}C^n + \frac{(1-\theta)\Delta t\beta}{1+\theta\Delta t\lambda}S_f^n p^n + \frac{\theta\Delta t\beta}{1+\theta\Delta t\lambda}S_f^{n+1}p^{n+1} \tag{33}$$

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. Rattlesnake currently implements both implicit and Crank-Nicholson as options for precursor evaluation.

2.3.2 Analytical Integration

Through prototyping, it has been found that neither implicit nor Crank-Nicholson time discretization of precursors are preferable methods for solving the shape equation in IQS. It has been found that these discretizations result in a lack of convergence of the shape over the IQS iteration process. In order to remedy the error, a analytical representation of the precursors was implemented in the prototype and the shape solution was able to converge (the normalization constant of the IQS method can be preserved to 10^{-10} while the theta-scheme only allowed convergence in the normalization factor to about 10^{-3}). The following section shows how this method was implemented in the prototype and the desired implementation for Rattlesnake.
Using an exponential operator, the precursor equation can be analytically solved for:

$$\int_{t_n}^{t_{n+1}} C(t')e^{\lambda t'}dt' = \int_{t_n}^{t_{n+1}} \beta(t')S_f(t')p(t')e^{\lambda t'}dt' \tag{34}$$

yielding

$$C^{n+1} = C^n e^{-\lambda(t_{n+1}-t_n)} + \int_{t_n}^{t_{n+1}} \beta(t')S_f(t')p(t')e^{-\lambda(t_{n+1}-t')}dt' \tag{35}$$

Because β and S_f being integrated are not known continuously over the time step, they can be interpolated linearly over the macro step. Such that:

$$h(t) = \frac{t_{n+1}-t}{t_{n+1}-t_n}h^n + \frac{t-t_n}{t_{n+1}-t_n}h^{n+1} \quad t_n \le t \le t_{n+1} \tag{36}$$

However, for the PRKE solve, we do have a very accurate representation of $p(t')$ over the time interval $[t_n, t_{n+1}]$.
Finally, we have the final expression for the analytical value for C^{n+1}:

$$C^{n+1} = C^n e^{-\lambda\Delta t} + \left(a_3\beta^{n+1} + a_2\beta^n\right)S_f^{n+1} + \left(a_2\beta^{n+1} + a_1\beta^n\right)S_f^n \tag{37}$$

Where the integration coefficients are defined as:

$$a_1 = \int_{t_n}^{t_{n+1}} \left(\frac{t_{n+1}-t'}{\Delta t}\right)^2 p(t')e^{-\lambda(t_{n+1}-t')}dt' \tag{38}$$

$$a_2 = \int_{t_n}^{t_{n+1}} \frac{(t'-t_n)(t_{n+1}-t')}{(\Delta t)^2}p(t')e^{-\lambda(t_{n+1}-t')}dt' \tag{39}$$

$$a_3 = \int_{t_n}^{t_{n+1}} \left(\frac{t'-t_n}{\Delta t}\right)^2 p(t')e^{-\lambda(t_{n+1}-t')}dt' \tag{40}$$

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 between those points can be done to maximize accuracy. The prototype code uses Matlab software to interpolate the amplitude between micro steps and a quadrature integration for the coefficients. So the challenge for Rattlesnake is to replicate this procedure: passing the amplitude vector to the DNP auxkernel, interpolating it, and integrating the coefficients.

2.4 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 specied, the IQS removal kernel ($\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}$) and the modified precursor kernel are bypassed, while all the postprocessors are still executed. However, it is difficult to rescale the flux to shape before the PRKE parameter postprocessor are executed. So the parameters are computed using the full flux, but amplitude is space independent and comes out of the integrals. As seen in parameter definitions, when shape is replaced with flux, the amplitude comes out of the integral and cancels out. So the conversion of the predicted flux in Equation 22 to shape is unnecessary if the corrected flux is solved with Equation 41. After obtaining the corrected flux, the precursors are re-evaluated using a `EXEC_LINEAR`.

$$\underbrace{\phi^g_{n+1}}_{\text{corrected}} = \underbrace{\phi^g_{n+1}}_{\text{predicted}} \frac{K_0}{K_{n+1}} p_{n+1} \tag{41}$$

2.5 Input

The input deck 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 number of micro time steps, IQS error tolerance, and 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 was made for the adjoint calculation, as well as including another multi-app and transfer in the IQS input deck.

2.6 Unintended Contributions

The implementation of IQS in Rattlesnake put pressure on many features of Rattlesnake and MOOSE that reveled bugs and possible improvement. Two significant issues in Rattlesnake that were found involved the adjoint solve and the diffusion fission kernel. When testing IQS, it was found that the adjoint flux solution was not the same as the forward flux solution in a single group test, which is obviously invalid. Also, when investigating which action to include the IQS kernel in, it turned out that the fission diffusion kernel was placed in the neutron transport action, so this kernel was demoted to the neutron diffusion actions for clarity. Additionally, the pressure on the save in feature in MOOSE propagated its application to boundary conditions and initial solves. MOOSE also updated its ability to restart dense vector data and to set MooseApp executioner right after executioner is created. Merge requests: #5474, #5489, #5495, and #5497

2.7 IQS in Other Action Systems

For simplicity and intuitiveness the previous section only described IQS implementation in CFEM diffusion. 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 Equation 1 and 20. However, in the derivation for IQS is 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.

The derivation of IQS was done using multi-group diffusion, but the derivation with transport is very similar. For simplicity, the transport equation is written in operator form in Equation 42. Factorization is defined as $\Psi^g(\vec{r},\vec{\Omega},t) = p(t)\psi^g(\vec{r},\vec{\Omega},t)$ and the resulting shape equation is defined by Equation 43. Finally, the PRKE parameters are defined by Equations 44 and 45, where $(\Psi^{*g}, f^g) = \int_{4\pi}\int_D \Psi^{*g}(\vec{r},\vec{\Omega}) f^g(\vec{r},\vec{\Omega}) d^3r d\Omega$.

$$\frac{1}{v^g}\frac{\partial \Psi^g}{\partial d} = \left(H^g + P^g_p - L^g\right)\Psi^g + S^g_d \tag{42}$$

$$\frac{1}{v^g}\frac{\partial \psi^g}{\partial d} = \left(H^g + P^g_p - L^g - \frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}\right)\psi^g + \frac{1}{p}S^g_d \tag{43}$$

$$\frac{\rho - \bar{\beta}}{\Lambda} = \frac{\sum_{g=1}^{G} \left(\Psi^{*g}, (H^g + P_p^g - L^g)\psi^g\right)}{\sum_{g=1}^{G} \left(\Psi^{*g}, \frac{1}{v^g}\psi^g\right)} \tag{44}$$

$$\frac{\bar{\beta}}{\Lambda} = \sum_{i=1}^{I} \frac{\bar{\beta}_i}{\Lambda} = \sum_{i=1}^{I} \frac{\sum_{g=1}^{G} (\Psi^{*g}, P_{d,i}^g \psi^g)}{\sum_{g=1}^{G} \left(\Psi^{*g}, \frac{1}{v^g}\psi^g\right)} \tag{45}$$

The derivation of IQS in transport is simple and straight-forward, and its implementation into Rattlesnake is even more so. The main differences between diffusion implementation in transport is outlined below:

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

3 Time Adaptation

A very important aspect of IQS is how it performs with time adaptation because this will most likely be used with IQS. The time adaptation used for quantifying IQS's ability is step doubling. 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 and a solution with two half steps. If the step is small enough, the error will be smaller than a user driven tolerance and the magnitude of the next step will be calculated based on the error. If the step is too large, the step will be repeated with a smaller step calculated with the resulting error. The error of the time step is approximated by Equation 46, where φ_1^g and φ_2^g are the solutions with the full step and half step, respectively. φ is changed to ϕ for regular flux evaluation and IQS P-C. If $e_n > e_{max}$ the time step is repeated; if $e_n < e_{max}$ the system moves to the next time step. The next Δt is calculated using Equation 47, where p is the convergence order of the time integration scheme being used. e_{max} and e_{tol} are user defined parameters; e_{max} is usually less than e_{tol} to better guarantee that the calculated Δt_{new} will pass the error criteria so time steps won't be repeated.

$$e_n = \frac{\left\| \sum_{g=1}^{G} \varphi_2^g - \sum_{g=1}^{G} \varphi_1^g \right\|_{L^2}}{\max\left(\left\| \sum_{g=1}^{G} \varphi_2^g \right\|_{L^2}, \left\| \sum_{g=1}^{G} \varphi_1^g \right\|_{L^2} \right)} \tag{46}$$

$$\Delta t_{new} = \Delta t_{old} \left(\frac{e_{tol}}{e_n} \right) \tag{47}$$

To be clear, each solution undergoes Picard iterations until $Error_{IQS}$ converges before the error is calculated; for IQS P-C, each solution is re-scaled by p. Additionally, if the PJFNK iteration does not converge, the entire step doubling process is repeated with half the time original time step.

4 RESULTS

This section describes results of an examples that tests the IQS implementation and shows its effectiveness on computation speed and accuracy. Three examples were selected for this purpose. The first is a homogeneous one-group problem, subjected to a heterogeneous material change (absorption cross-section change as a ramp in time for a subset of the geometry). The second is the two-dimensional TWIGL ramp transient benchmark, described further. The final is the LRA benchmark, which a two-dimensional, temperature feedback example. Each example was solved with regular diffusion (brute force), IQS, and IQS P-C. Each method was tested with backward-Euler and BDF2 for the first two examples; just Crank-Nicholson was used for third example. The performance for each method are represented by convergence plots with constant Δt and by the number of time steps with step doubling time adaptation.

4.1 One-Dimensional Custom Example

The example is very simple and computes quickly, it entails a one dimensional, heterogeneous 400 cm slab with a varying absorption cross section. Figure 2 how the regions of the slab are divided and Table 1 shows the initial material properties. Table 2 shows the ramp of the absorption cross-section of each region.

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

Figure 2: 1-D heterogeneous slab region identification

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

Region	$D(cm)$	$\Sigma_a(cm^{-1})$	$\nu\Sigma_f(cm^{-1})$	$v(cm/s)$	β	$\lambda(s^{-1})$
1	1.0	1.1	1.1	1,000	0.006	0.1
2	1.0	1.1	1.1	1,000	0.006	0.1
3	1.0	1.1	1.1	1,000	0.006	0.1
4	1.0	1.1	1.1	1,000	0.006	0.1

Table 2: 1-D heterogeneous slab absorption cross-section slope perturbation

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

Figure 3 shows the power at each macro time step as compared to the traditional brute force (full flux time discretization) method. The strong correlation between the two curves shows that IQS is consistent with a proven method for a highly transient example. Figure 7 shows that IQS is not only consistent for this example, but also has a better error constant in the convergence study. Figures 4 - 6 plots shape changes in the IQS method, showing where the shape solution is necessary and a simple PRKE evaluation is inadequate. Figure 7 shows the error convergence for brute force and IQS methods. This plot shows that IQS has consistent convergence slope for first and second order time schemes and that IQS has a better error constant for the convergence. Table 3 shows the results for time adaptation. These results show that traditional IQS does not perform well with step doubling time adaptation and IQS P-C performs moderately better than brute force.

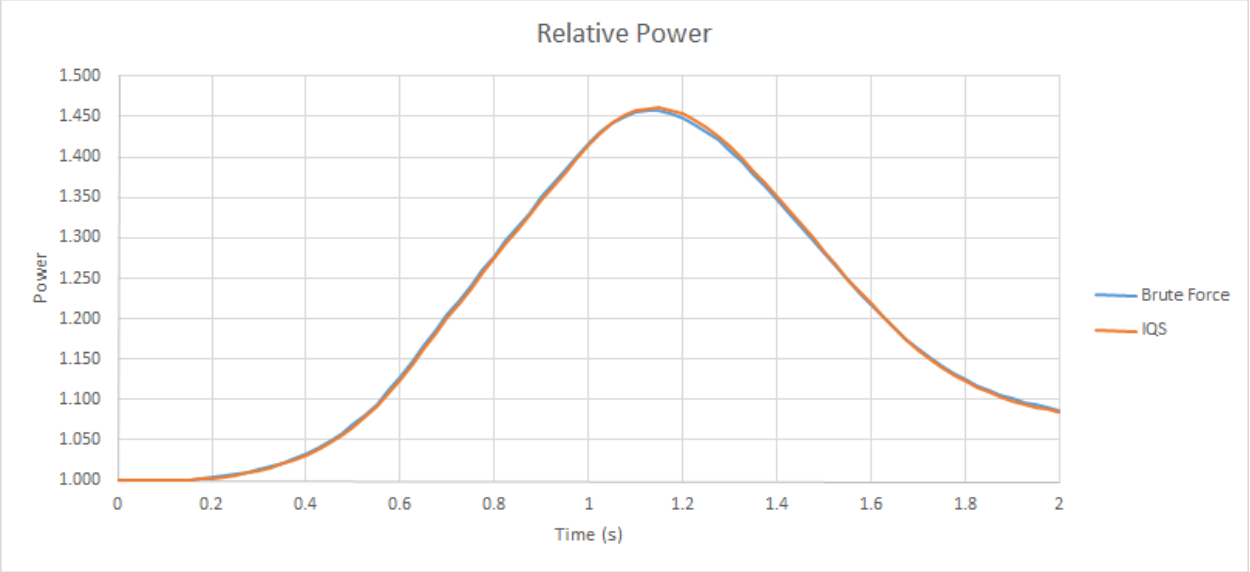


Figure 3: Power level of 1D heterogeneous example using IQS and Brute Force with $\Delta t = 0.025$

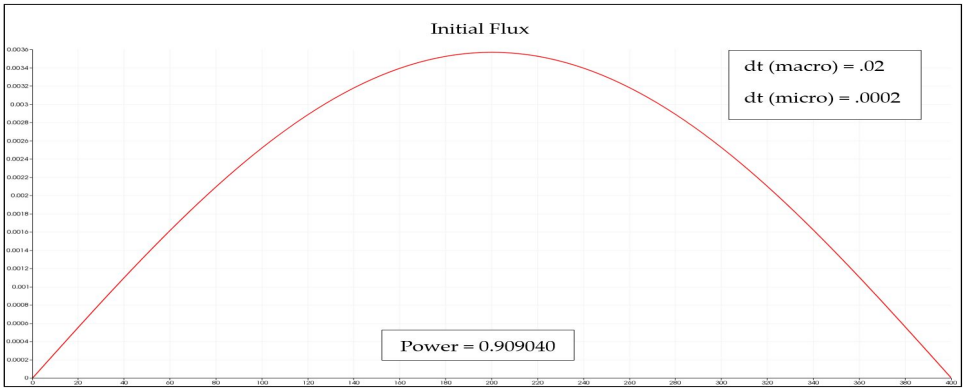


Figure 4: Initial Flux Plot

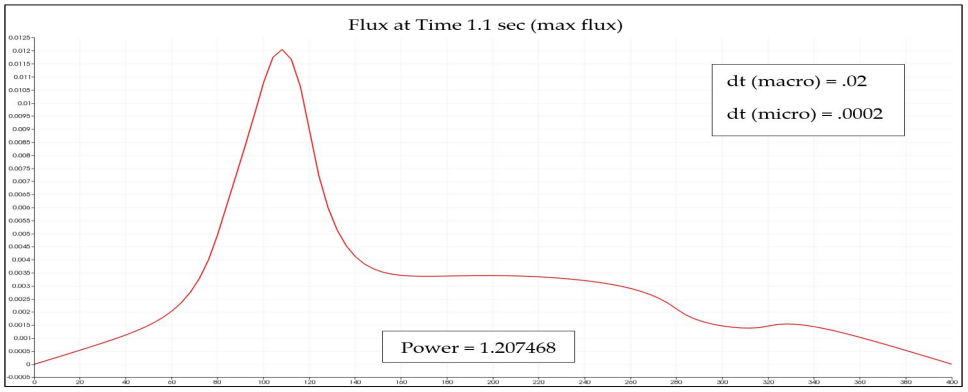


Figure 5: Flux Plot when Absorption Cross Section is at Minimum

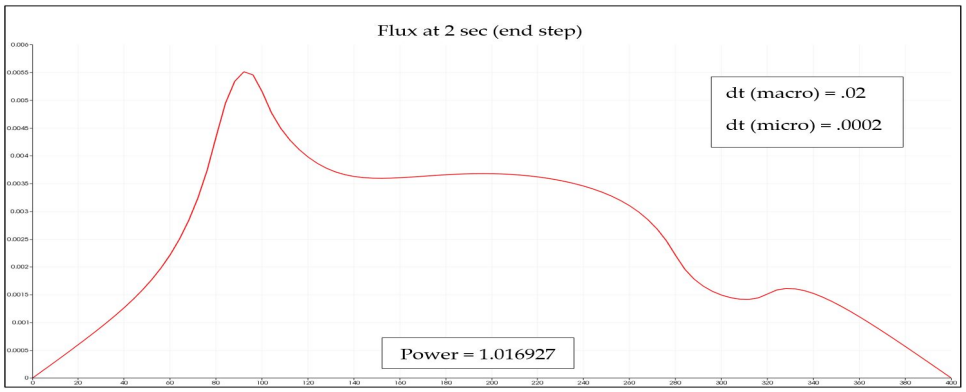


Figure 6: Final Flux Computation (not steady-state)

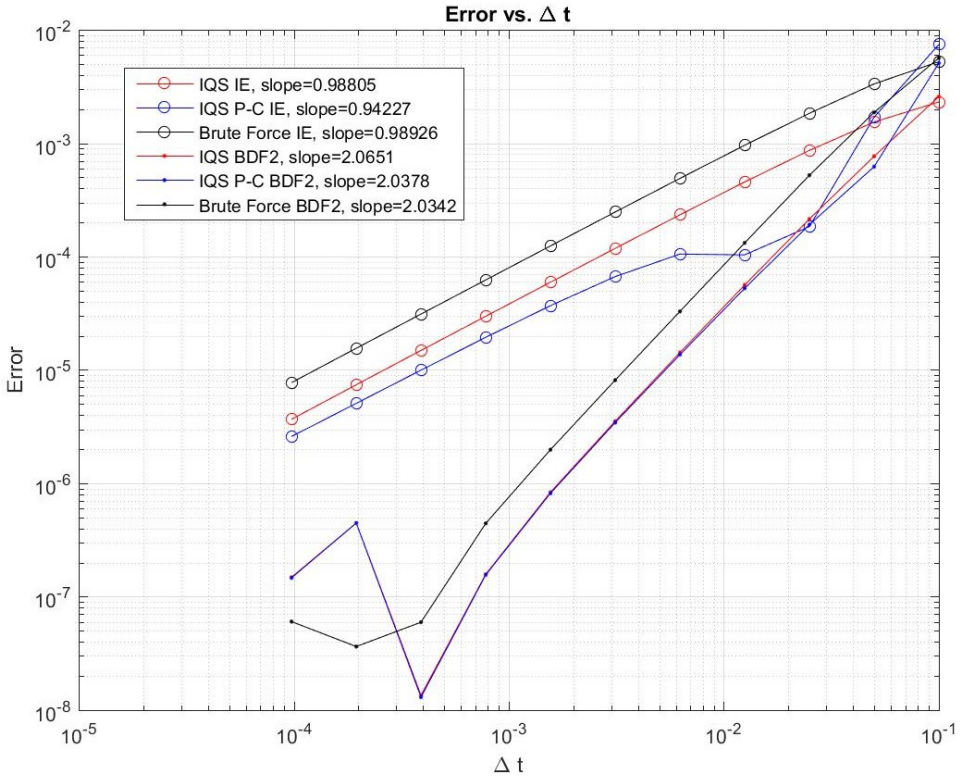


Figure 7: Error convergence comparison of 1D hetergenous example

Table 3: 1-D heterogeneous slab step doubling results

		Brute Force			IQS			IQS P-C		
Test	e_{max}	Error	Steps	Solves	Error	Steps	Solves	Error	Steps	Solves
1	0.05	0.021596	10	32	0.10084	12	175	0.028019	10	33
2	0.01	0.032864	21	80	0.08030	41	850	0.044721	20	74
3	0.005	0.053159	27	96	0.01215	115	2220	0.052095	25	96
4	0.001	0.056546	56	188	0.03242	848	12511	0.061274	50	162
5	0.0005	0.062882	77	245	0.03491	1718	25841	0.061240	69	225
6	0.0001	0.060089	177	537	0.03554	8702	129985	0.060824	159	480
7	5.0e-05	0.059513	252	767	0.03622	17282	256463	0.061078	224	680
8	1.0e-05	0.061063	561	1691	0.03142	79988	1104227	0.061901	501	1509

4.2 TWIGL Benchmark

This benchmark problem originates from the Argonne National Lab Benchmark Problem Book. It is a 2D, 2-group reactor core model with no reflector region shown in Figure 8. Table 4 shows the material properties of each fuel region and the ramp perturbation of Material 1.

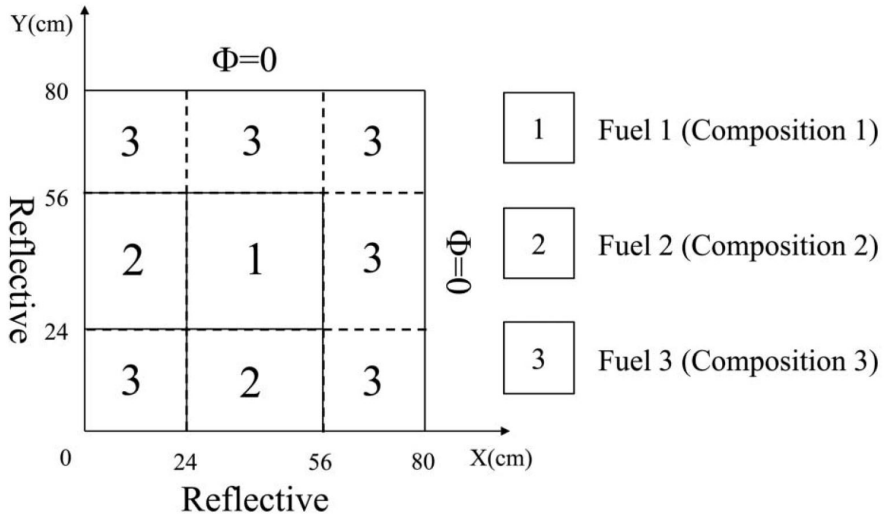


Figure 8: TWIGL benchmark problem description

Table 4: 1-D heterogeneous slab absorption cross-section slope perturbation

Material	Group	$D(cm)$	$\Sigma_a(cm^{-1})$	$\nu\Sigma_f(cm^{-1})$	χ	$\Sigma_s(cm^{-1})$ $g \to 1$	$\Sigma_s(cm^{-1})$ $g \to 2$
1	1	1.4	0.010	0.007	1.0	0.0	0.01
	2	0.4	0.150	0.200	0.0	0.0	0.00
2	1	1.4	0.010	0.007	1.0	0.0	0.01
	2	0.4	0.150	0.200	0.0	0.0	0.00
3	1	1.3	0.008	0.003	1.0	0.0	0.01
	2	0.5	0.050	0.060	0.0	0.0	0.00
	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:

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

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

Figures 9 and 10 show the IQS solution as compared with the Brute Force solution. It is important to note the IQS shape plot is scaled differently than the Brute Force flux plot because the amplitude term is not included, but the gradients of colors is comparable. These plots show that IQS is consistent in more complex, higher dimensional problems in RATTLESNAKE. Finally, Figure 11 plots the error convergence of IQS and the Brute Force methods. The curves show the impressive convergence of IQS for the highly transient TWIGL example. Table 5 shows the results for time adaptation. The results show that both IQS methods perform exceptionally well compared to brute force. It also shows that traditional IQS performed better with large e_{max}, while IQS P-C was better with smaller e_{max}.

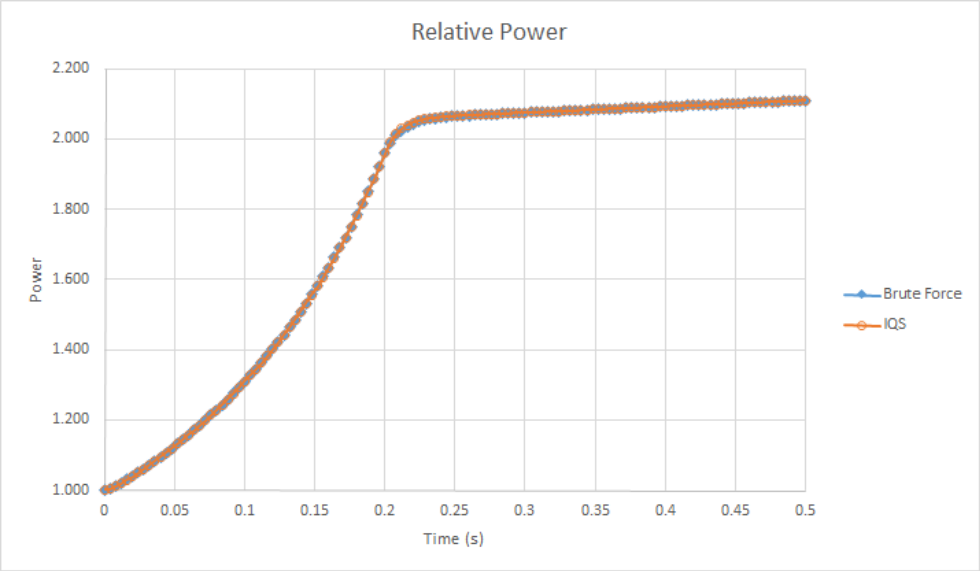
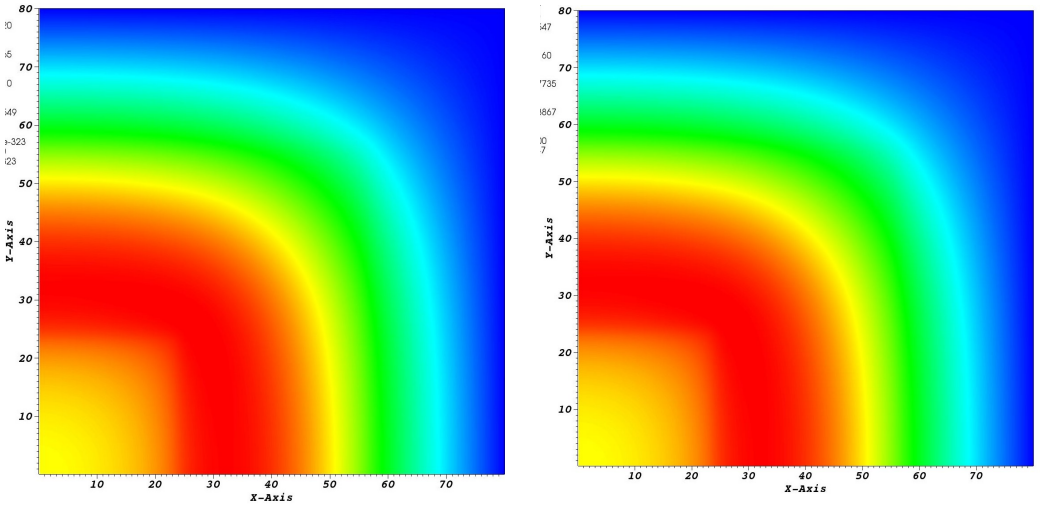


Figure 9: Power level comarison of 1D heterogeneous example between IQS and Brute Force using $\Delta t = 0.004$



(a) Brute force flux

(b) IQS Shape

Figure 10: TWIGL Benchmark flux/shape comparison at $t = 0.2$

Table 5: TWIGL step doubling results

		Brute Force			IQS			IQS P-C		
Test	e_{max}	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	7	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

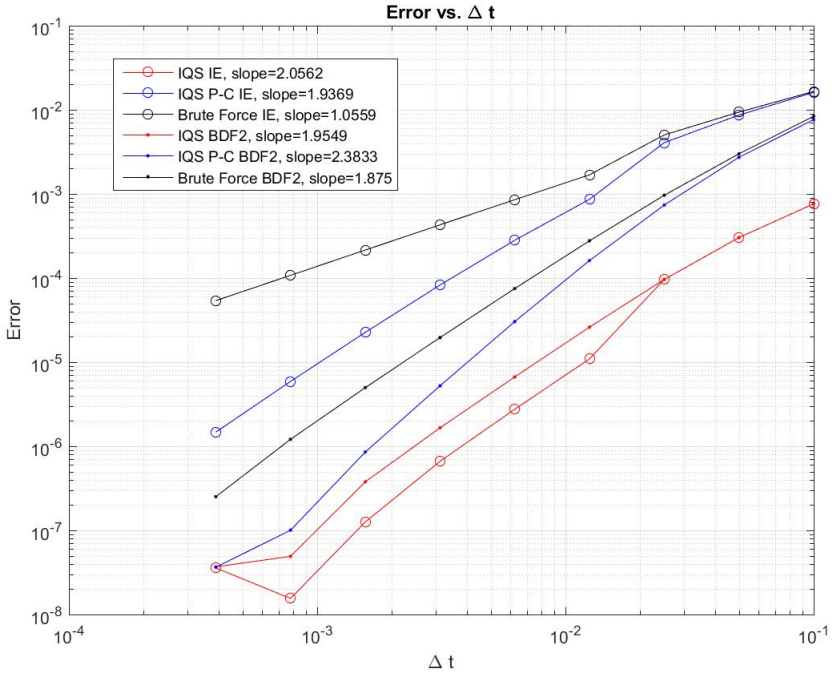


Figure 11: Error convergence comparison of TWIGL Benchmark

4.3 LRA Benchmark

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

$$-\frac{1}{v_1}\frac{\partial \phi_1}{\partial t} = -\nabla D_1 \nabla \phi_1 + (\Sigma_{a,1} + \Sigma_{s,1\to 2})\phi_1 - \nu(1-\beta)S_f - \sum_{i=1}^{2}\lambda_i C_i, \tag{48}$$

$$-\frac{1}{v_2}\frac{\partial \phi_2}{\partial t} = -\nabla D_2 \nabla \phi_1 + \Sigma_{a,2}\phi_2 - \Sigma_{s,1\to 2}\phi_1, \tag{49}$$

$$S_f = \sum_{g=1}^{2}\Sigma_{f,g}\phi_g, \tag{50}$$

$$\frac{\partial C_i}{\partial t} = \nu\beta_i f - \lambda_i C_i, \quad i = 1,2, \tag{51}$$

$$\frac{\partial T}{\partial t} = \alpha f, \tag{52}$$

$$\Sigma_{a,1} = \Sigma_{a,1}(\vec{r}, t=0)\left[1+\gamma\left(\sqrt{T}-\sqrt{T_0}\right)\right], \tag{53}$$

$$P = \kappa S_f, \tag{54}$$

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; $\Sigma_{s,1\to 2}$ 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; $S_6 f$ 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

$$-\nabla D_1 \nabla \phi_1 + (\Sigma_{a,1} + \Sigma_{s,1\to 2})\phi_1 = \frac{1}{k}\sum_{g=1}^{2} \nu\Sigma_{f,g}\phi_g, \tag{55}$$

$$-\nabla D_2 \nabla \phi_1 + \Sigma_{a,2}\phi_2 = \Sigma_{s,1\to 2}\phi_1. \tag{56}$$

$$\tag{57}$$

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

$$\bar{P} \equiv \frac{\int_{V_{core}} P(\vec{r}, t = 0) d\vec{r}}{\int_{V_{core}} d\vec{r}}, \tag{58}$$

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 12.

Initial two-group constants are presented in Table 6. 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 7. All fuel materials have the same delayed neutron data. Some scalar data are listed in Table 8.

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

$$\Sigma_{a,2}(t) = \Sigma_{a,2}(t = 0) \begin{cases} 1 - 0.0606184t, & t \leq 2 \\ 0.8787631, & t > 2 \end{cases} \tag{59}$$

where t is time in seconds.

Figure 13 show the IQS solution as compared with the Brute Force solution. These plots show that IQS is consistent with Doppler feedback problems in RATTLESNAKE. Finally, Figure 14 plots the error convergence of IQS and the Brute Force methods.

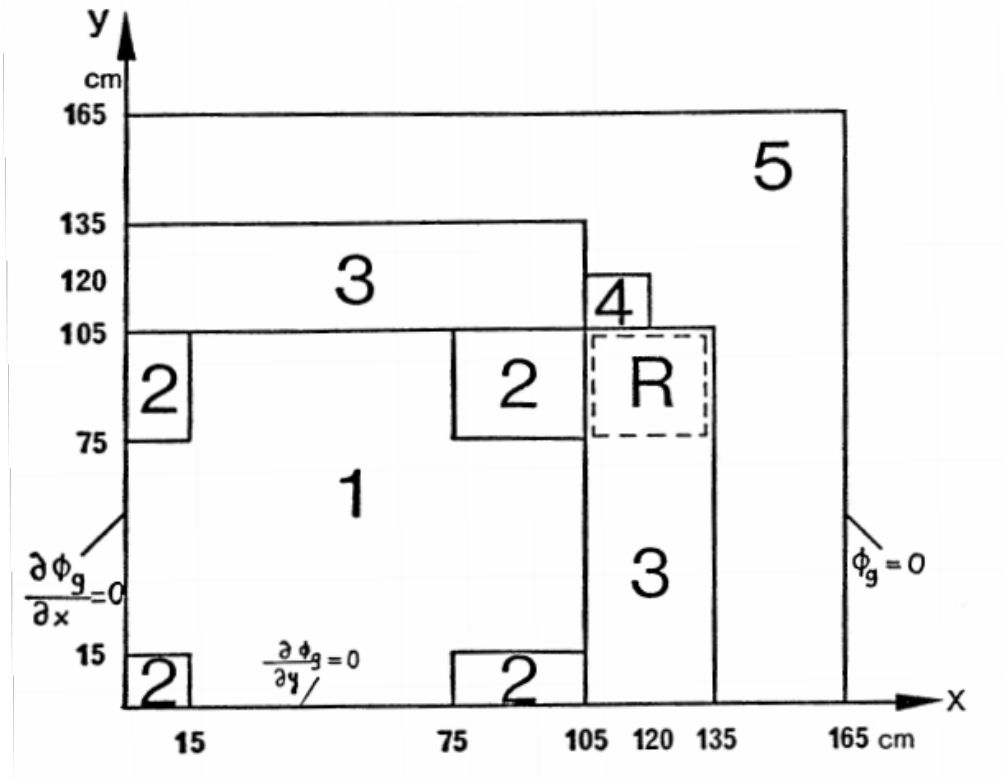


Figure 12: LRA benchmark geometry with region assignment.

Table 6: LRA benchmark initial two-group constants.

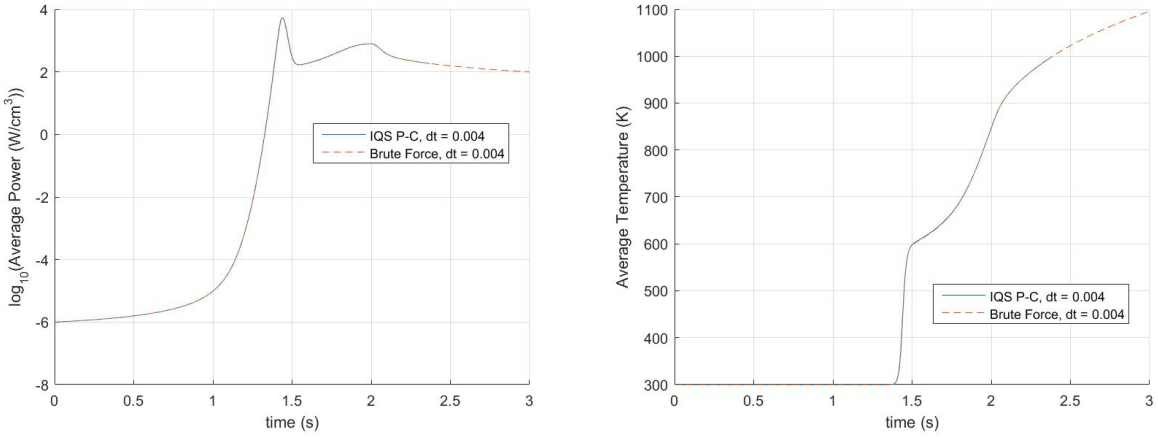
Region	Material	Group g	D_g (cm)	$\Sigma_{a,g}$ (cm^{-1})	$\nu\Sigma_{f,g}$ (cm^{-1})	$\Sigma_{s,1\to2}$ (cm^{-1})	χ_g	v_g $(cm \cdot s^{-1})$
1	Fuel 1 with rod	1	1.255	0.008252	0.004602		1	3.0×10^7
		2	0.211	0.1003	0.1091	0.02533	0	3.0×10^5
2	Fuel 1 without rod	1	1.268	0.007181	0.004609		1	3.0×10^7
		2	0.1902	0.07047	0.08675	0.02767	0	3.0×10^5
3	Fuel 2 with rod	1	1.259	0.008002	0.004663		1	3.0×10^7
		2	0.2091	0.08344	0.1021	0.02617	0	3.0×10^5
4	Fuel 2 without rod	1	1.259	0.008002	0.004663		1	3.0×10^7
		2	0.2091	0.073324	0.1021	0.02617	0	3.0×10^5
5	Reflector	1	1.257	0.0006034	-		-	3.0×10^7
		2	0.1592	0.01911	-	0.04754	-	3.0×10^5

Table 7: 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 8: LRA benchmark scalar values.

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



(a) Power Profile

(b) Temperature Profile

Figure 13: LRA Benchmark power and temperature comparison

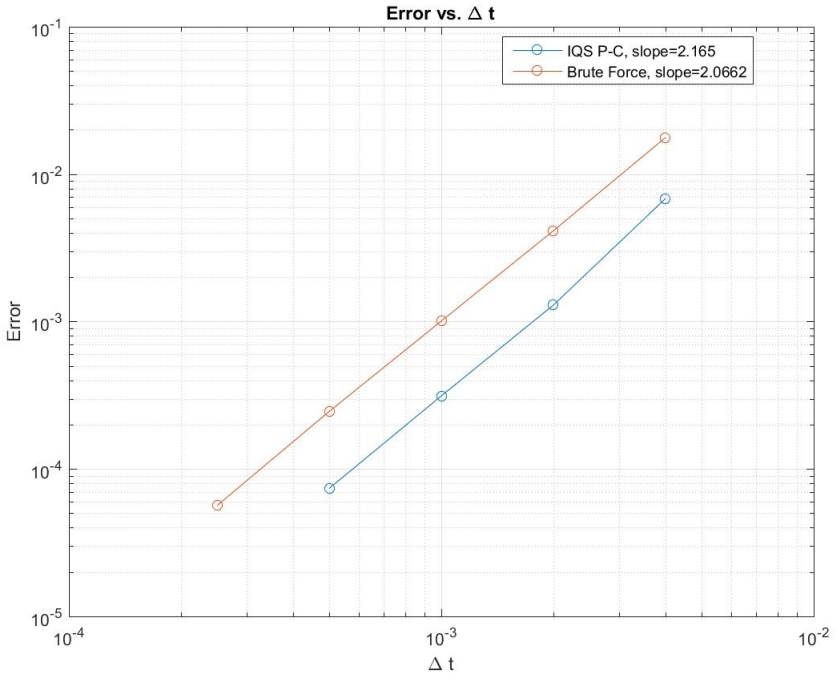


Figure 14: Error convergence comparison of LRA Benchmark