INL/EXT-16-38059

Improved Quasi-Static Method

IQS Method Implementation for CFEM Diffusion in Rattlesnake

Zachary M. Prince, Jean C. Ragusa, Yaqi Wang

February 2016

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/EXT-16-38059

Improved Quasi-Static Method
IQS Method Implementation for CFEM Diffusion in Rattlesnake

Zachary M. Prince[1]
Jean C. Ragusa[1]
Yaqi Wang[1]

[1]**Department of Nuclear Engineering**
Texas A&M University, College Station, TX, USA
[2]**Idaho National Laboratory**

February 2016

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

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

February 29, 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:

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

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

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:

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

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

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:

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.

2 Rattlesnake Implementation

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

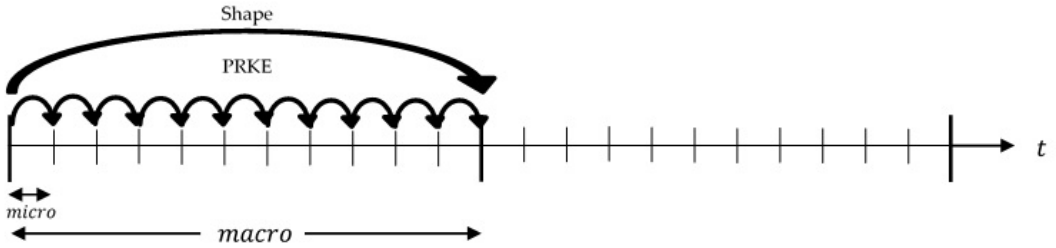


Figure 1: IQS method visualization

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 backward Euler to retain simplicity and insure convergence, but higher order methods are an obvious next step for this computation. The IQS executioner also supplements Transients Picard iteration process by adding its own error criteria:

$$Error_{IQS} = \left| \frac{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^{g,n} \right)}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^{g,0} \right)} - 1 \right| \tag{22}$$

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:

$$\begin{aligned} \frac{1}{v^g} \frac{\partial \varphi^g}{\partial t} = & \underbrace{\frac{\chi_p^g}{k_{eff}} \sum_{g'=1}^{G} (1-\beta) \nu^{g'} \Sigma_f^{g'} \varphi^{g'}}_{FluxKernel} + \underbrace{\sum_{g' \neq g}^{G} \Sigma_s^{g' \to g} \varphi^{g'}}_{FluxKernel} - \underbrace{\left(-\nabla \cdot D^g \nabla\right) \varphi^g}_{FluxKernel} - \underbrace{\Sigma_r^g \varphi^g}_{FluxKernel} \\ & \underbrace{- \frac{1}{v^g} \overbrace{\frac{1}{p} \frac{dp}{dt}}^{FromExecutioner} \varphi^g}_{IQSKernel} + \underbrace{\frac{1}{p} \sum_{i=1}^{I} \chi_{d,i}^g \lambda_i C_i}_{ModifiedFluxKernel} \end{aligned} \tag{23}$$

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.

$$\frac{1}{v}\frac{\partial\varphi}{\partial t} = \nu\Sigma_f(1-\beta)\varphi - \left(-\nabla\cdot D\nabla + \Sigma_a + \frac{1}{v}\frac{1}{p}\frac{dp}{dt}\right)\varphi + \frac{1}{p}\lambda C \tag{24}$$

$$\frac{dC}{dt} = \beta\nu\Sigma_f\varphi p - \lambda C \tag{25}$$

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

$$\frac{u^{n+1}-u^n}{\Delta t} = (1-\theta)f(u^n,t) + \theta f(u^{n+1},t)\,. \tag{26}$$

Applying this to the precursor equation:

$$\frac{C^{n+1}-C^n}{\Delta t} = (1-\theta)\beta S_f^n p^n - (1-\theta)\lambda C^n + \theta\beta S_f^{n+1}p^{n+1} - \theta\lambda C^{n+1} \tag{27}$$

Where S_f is the fission source equivalent for shape:

$$S_f^n = (\nu\Sigma_f)^n\varphi^n \tag{28}$$

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

$$C^{n+1} = \frac{1-(1-\theta)\Delta t\lambda}{1+\theta\Delta t\lambda}C^n + \frac{(1-\theta)\Delta t\beta}{1+\theta\Delta t\lambda}S_f^n p^n + \frac{\theta\Delta t\beta}{1+\theta\Delta t\lambda}S_f^{n+1}p^{n+1} \tag{29}$$

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:

$$\int_{t_n}^{t_{n+1}} C(t')e^{\lambda t'}dt' = \int_{t_n}^{t_{n+1}} \beta(t')S_f(t')p(t')e^{\lambda t'}dt' \tag{30}$$

yielding

$$C^{n+1} = C^n e^{-\lambda(t_{n+1}-t_n)} + \int_{t_n}^{t_{n+1}} \beta(t')S_f(t')p(t')e^{-\lambda(t_{n+1}-t')}dt' \tag{31}$$

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:

$$h(t) = \frac{t_{n+1}-t}{t_{n+1}-t_n}h^n + \frac{t-t_n}{t_{n+1}-t_n}h^{n+1} \quad t_n \leq t \leq t_{n+1} \tag{32}$$

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}:

$$C^{n+1} = C^n e^{-\lambda\Delta t} + \left(a_3\beta^{n+1} + a_2\beta^n\right)S_f^{n+1} + \left(a_2\beta^{n+1} + a_1\beta^n\right)S_f^n \tag{33}$$

Where the integration coefficients are defined as:

$$a_1 = \int_{t_n}^{t_{n+1}} \left(\frac{t_{n+1}-t'}{\Delta t}\right)^2 p(t')e^{-\lambda(t_{n+1}-t')}dt' \tag{34}$$

$$a_2 = \int_{t_n}^{t_{n+1}} \frac{(t'-t_n)(t_{n+1}-t')}{(\Delta t)^2}p(t')e^{-\lambda(t_{n+1}-t')}dt' \tag{35}$$

$$a_3 = \int_{t_n}^{t_{n+1}} \left(\frac{t'-t_n}{\Delta t}\right)^2 p(t')e^{-\lambda(t_{n+1}-t')}dt' \tag{36}$$

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 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.

2.5 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

3 Current Status

IQS has almost completely been implemented to CFEM Diffusion in Rattlesnake. The method currently passes multi-dimensional and multi-group null-transient tests. However, there are currently three prevalent issues that are restricting full implementation of IQS in Rattlesnake.

1. The Transient executioner is currently being worked on to improve its flexibility. Until this improvement is complete, the IQS error contribution is unable to be supplemented to Transient. For testing purposes, Transient has been modified locally to support the error contribution.

2. IQS is currently being tested with higher order schemes for diffusion evaluation in the matlab prototype. This is to make sure that IQS will perform with similarly when subjected to higher accuracy.

After CFEM implementation is completed, the next step is to apply IQS to DFEM and ultimately neutron transport. These applications should be much simpler because the base of IQS has already been implemented and verified from this CFEM work. Another method of IQS, called the predictor-corrector method, could also be implemented in the future. This method is very simple to implement and actually faster than standard IQS because there is no iteration process. However, the performance of the method has yet to be evaluated to prove its worth in Rattlesnake.

4 RESULTS

This section describes results of an examples that tests the IQS implementation and shows its effectiveness on computation speed and accuracy. Two examples were selected for this purpose. The first is a homogeneous one-group problem, subjected to a heterogenous 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.

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

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

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

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

Figure 7 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 **??** 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.

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 3 shows the material properties of each fuel region and the ramp perturbation of Material 1.

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

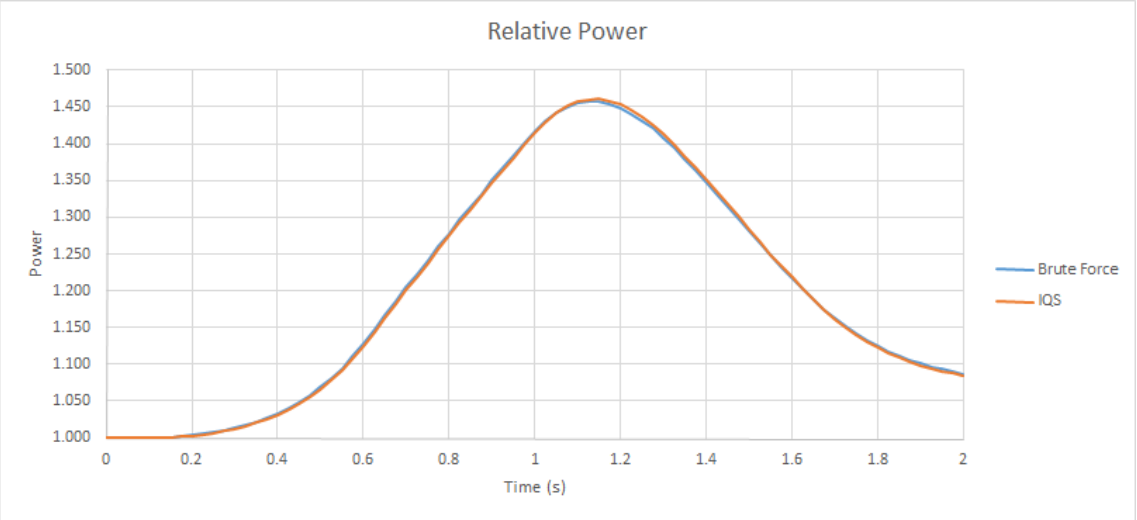


Figure 3: Power level comparison of 1D heterogeneous example between IQS and Brute Force using $\Delta t = 0.025$

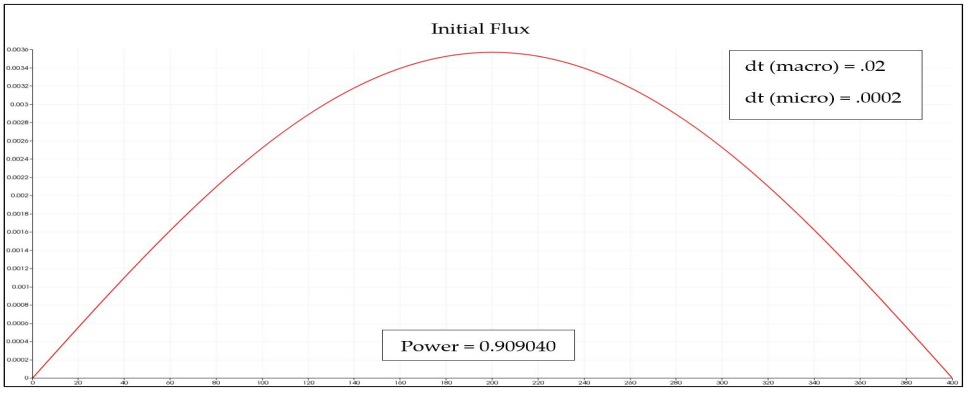


Figure 4: Initial Flux Plot

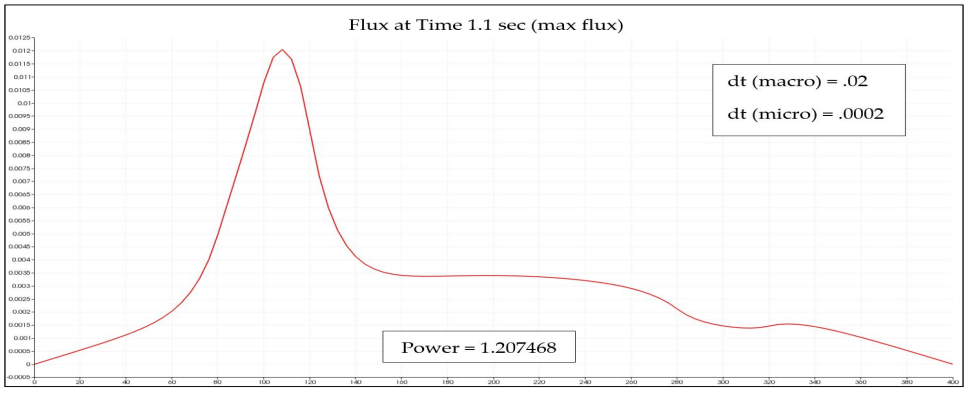


Figure 5: Flux Plot when Absorption Cross Section is at Minimum

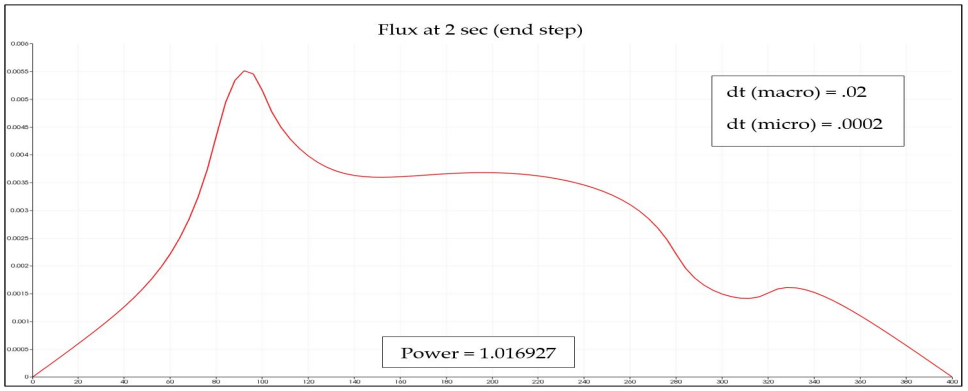


Figure 6: Final Flux Computation (not steady-state)

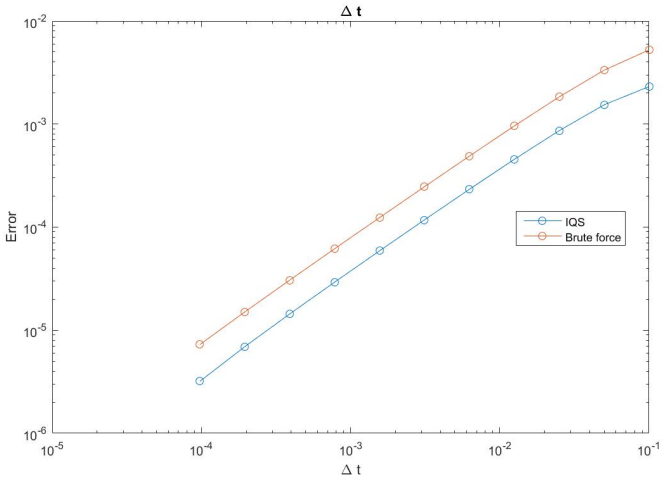


Figure 7: Error convergence comparison of 1D hetergenous example

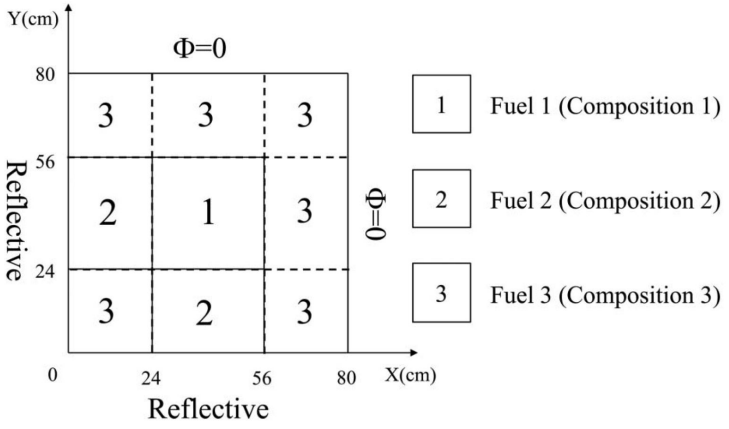


Figure 8: TWIGL benchmark problem description

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:

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

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

Table 3: 1-D heterogeneous slab absorption cross-section slope perturbation

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 transience TWIGL example.

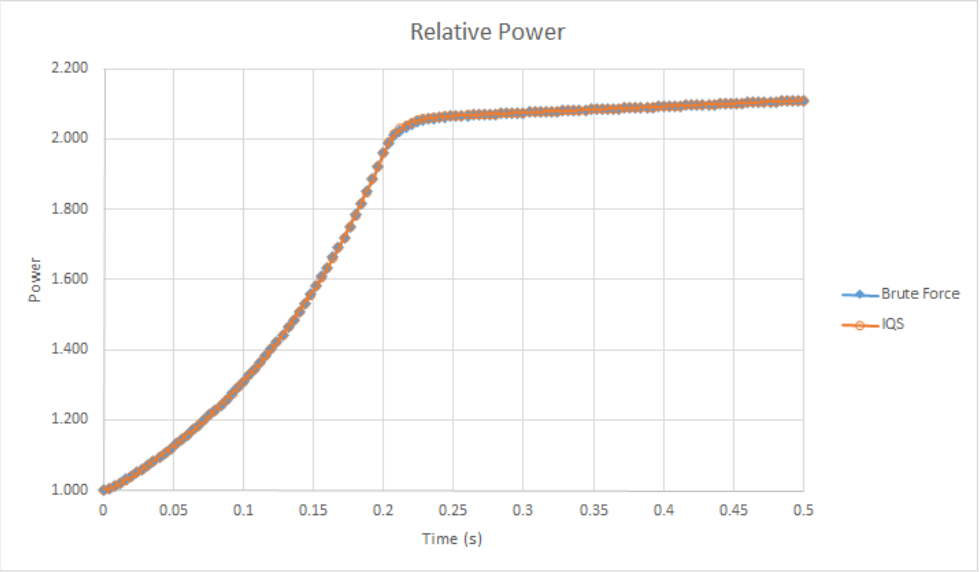
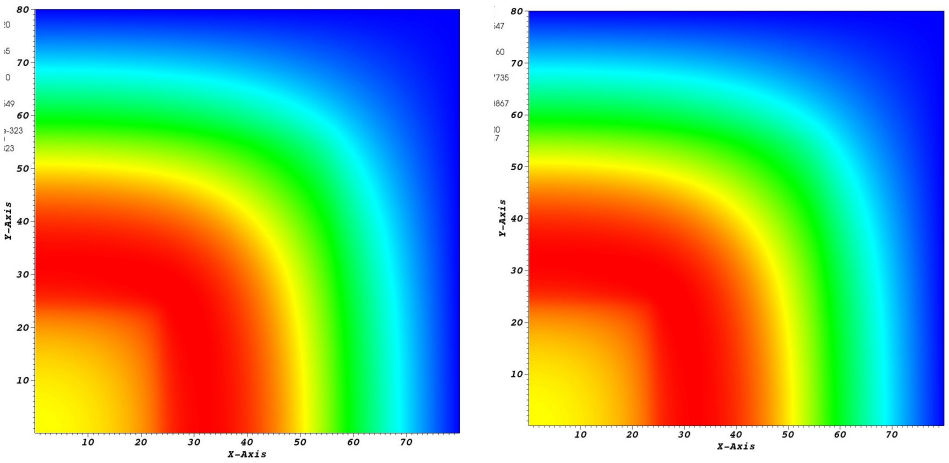


Figure 9: Power level comarison of 1D heterogeneous example between IQS and Brute Force using $\Delta t = 0.004$



(a) Brute force flux

(b) IQS Shape

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

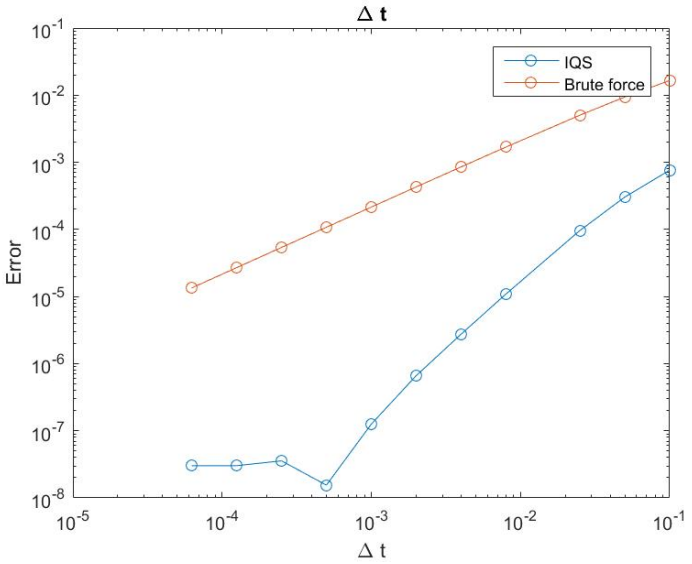


Figure 11: Error convergence comparison of TWIGL Benchmark