Improved Quasi-Static Methods in Rattlesnake

Development and Implementation of Improved Quasi-Static (IQS) methods for time-dependent neutron diffusion and neutron transport solvers in Rattlesnake

Zachary M. Prince[†], Jean C. Ragusa[†], Yaqi Wang[⋆], Mark D. DeHart[⋆]

[†]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, mark.dehart@inl.gov

June 15, 2016

Abstract This report presents the implementation of the Improved Quasi-Static method (IQS) in the Rattlesnake application of the MOOSE framework. In the IQS method, the neutron flux is factored into a time-dependent amplitude and a spatial- and weakly time-dependent shape. The shape evaluation is very similar to a standard flux solve and is computed at large (macro) time steps. The amplitude is obtained by solving the PRKEs on a micro time scale. The PRKE parameters are regularly updated based on the shape changes. Two versions of the IQS methodology have been implemented: (i) the standard approach, which leads to a system of nonlinear equations between the shape and the amplitude, and (ii) a predictor corrector approach that linearizes the system of equations. A time-adaptation technique has been developed to automatically select an appropriate macro time step size based on user-specified tolerances. The implementation has been verified and tested using manufactured solutions, a custom one-dimensional example, the TWIGL ramp benchmark, and the LRA benchmark. The numerical results include error convergence plots and performance with time adaptation. These examples prove it to be a viable and effective method for transient neutronic simulations. More complex simulations (e.g., TREAT transients) are planned to perform coupled neutronics/heat conduction transient simulations within the MAMMOTH application.

1 Theory

The improved quasi-static (IQS) method is a transient spatial kinetics method that involves factorizing the neutron flux into a space- and time-dependent component (the shape) and a time-dependent component (the amplitude) [3, 6]. The technique relies on the shape being less rapidly varying in time compared to the flux, hence requiring fewer shape computations or updates. The IQS method has primarily been developed in the context of the neutron diffusion approximation [2, 4], but reference [5] provides an extension of the technique to transport, implemented in TDTORT. The method has mostly being applied to neutron kinetics, without thermal-hydraulic feedback. In this report, we extend the IQS method in the context of transient multiphysics simulations.

1.1 IQS for Multigroup Diffusion Equations

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

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

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

Extension to the transport equations will be provided subsequently, once a concise operator notation has been introduced. In the above equations,

ϕ^g	=	Scalar flux in energy group g
C_i	=	Concentration of delayed neutron precursor i
Σ_f^g	=	Fission cross section in energy group g
Σ_r^g	=	Removal cross section in energy group g
$\Sigma_s^{g'\to g}$	=	Scattering cross section from energy group g' to g
v^g	=	Neutron velocity in energy group g
χ_p^g	=	Fission spectrum of prompt neutrons
$\chi_{d,i}^g$	=	Fission spectrum of delayed neutrons from precursor i
ν^g	=	Total number of neutrons per fission
D^g	=	Diffusion coefficient in energy group g
λ_i	=	Decay constant of precursor i
β_i	=	Delayed neutron fraction from precursor i
β	=	Total delayed neutron fraction ($\beta = \sum_{i=1}^{I}\beta_i$)

Note that the prompt fission terms have been normalized by k_{eff}, the eigenvalue of the initial state, as transient simulations are often carried out for initially critical reactors. The expressions for subcritical and source-driven problems are straightforward to derive but omitted in

this report for brevity.

The factorization approach in IQS 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):

$$\phi^g(\vec{r},t) = p(t)\varphi^g(\vec{r},t) \tag{2}$$

We stress that the factorization is not an approximation. Reporting it in the flux and precursor equations yields, after some simple algebra, the following equations for the shape and the precursors:

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

$$\frac{dC_i}{dt} = \frac{\beta_i}{k_{eff}}\boxed{p}\sum_{g=1}^{G}\nu^g\Sigma_f^g\varphi^g - \lambda_i C_i\ , \quad 1\leq i\leq I \tag{3b}$$

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

1. The shape equation contains an additional term equivalent to a removal cross section, $\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}$.
2. The delayed neutron source term is divided by p.
3. The system of equations is now nonlinear due to the factorization.
4. An equation is needed to obtain the amplitude p.

To derive the amplitude equation (also known as the PRKE equations), the shape/precursors equations are weighed by a space-dependent function (typically the initial adjoint flux, ϕ^{*g}) and integrated over the phase-space. The final expressions are given below:

$$\frac{dp}{dt} = \left[\frac{\rho-\bar{\beta}}{\Lambda}\right]p + \sum_{i=1}^{I}\bar{\lambda}_i\xi_i \tag{4a}$$

$$\frac{d\xi_i}{dt} = \frac{\bar{\beta}_i}{\Lambda} - \bar{\lambda}_i\xi_i \quad 1\leq i\leq I \tag{4b}$$

where we have defined the reactivity, effective delayed-neutron fraction, and delayed-neutron precursor decay constant as follows:

$$\frac{\rho-\bar{\beta}}{\Lambda} = \frac{\sum_{g=1}^{G}\left(\phi^{*g},\frac{\chi_p^g}{k_{eff}}(1-\beta)\sum_{g'=1}^{G}\nu^{g'}\Sigma_f^{g'}\varphi^{g'} + \sum_{g'\neq g}^{G}\Sigma_s^{g'\to g}\varphi^{g'} - \left(-\nabla\cdot D^g\nabla+\Sigma_r^g\right)\varphi^g\right)}{\sum_{g=1}^{G}\left(\phi^{*g},\frac{1}{v^g}\varphi^g\right)} \tag{5a}$$

$$\frac{\bar{\beta}}{\Lambda} = \sum_{i=1}^{I} \frac{\bar{\beta}_i}{\Lambda} = \sum_{i=1}^{I} \frac{1}{k_{eff}} \frac{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^{g} \beta_i \sum_{g'=1}^{G} \nu^{g'} \Sigma_f^{g'} \varphi^{g'})}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g\right)} \tag{5b}$$

$$\bar{\lambda}_i = \frac{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^{g} \lambda_i C_i)}{\sum_{g=1}^{G} (\phi^{*g}, \chi_{d,i}^{g} C_i)} \tag{5c}$$

The following inner product definition has been used: $(\phi^{*g}, f) := \int_D \phi^{*g}(\vec{r}) f(\vec{r}) d^3r$. Additionally, in order to impose uniqueness on the factorization and to derive the PRKE, the following normalization condition is imposed: $\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g\right) = constant$. This condition is later used as the convergence criteria when solving the nonlinear system of equations resulting from the IQS method.

1.2 Operator Notation and Extension of IQS to Multigroup Transport Equations

For simplicity, the previous section only described the IQS implementation using the neutron diffusion equation. However, deriving the IQS form for other neutron balance equations (e.g., transport, simplified transport, etc.) is very similar. To this end, we re-write the neutron conservation equations in operator form, shown in Equation 6.

$$\frac{\partial}{\partial t} \left(\frac{\Psi^g}{v^g} \right) = \sum_{g'} \left(H^{g' \to g} + P_p^{g' \to g} \right) \Psi^{g'} - L^g \Psi^g + S_d^g \tag{6}$$

where Ψ^g is the multigroup neutron flux (angular flux in the case of transport), $H^{g' \to g}$ is the scattering operator, $P_p^{g' \to g}$ is the prompt neutron production operator, L^g is the loss operator, and S_d^g is the delayed neutron source operator. Using Equation 1a, the reader may easily obtain the functional form for these operators in the case of a diffusion approximation. Next, the factorization $\Psi^g(\vec{r}, \vec{\Omega}, t) = p(t) \psi^g(\vec{r}, \vec{\Omega}, t)$ is introduced, where the shape is denoted by ψ^g, leading to the following shape equations, Equation 7:

$$\frac{\partial}{\partial t} \left(\frac{\psi^g}{v^g} \right) = \sum_{g'} \left(H^{g' \to g} + P_p^{g' \to g} \right) \psi^{g'} - \left(L^g + \frac{1}{v^g} \frac{1}{p} \frac{dp}{dt} \right) \psi^g + \frac{1}{p} S_d^g \tag{7}$$

Finally, the PRKE parameters are defined by Equations 8 and 9, where $(\Psi^{*g}, f^g) = \int_{4\pi} \int_D \Psi^{*g}(\vec{r}, \vec{\Omega}) f^g(\vec{r}, \vec{\Omega}) d^3r d^2\Omega$.

$$\frac{\rho - \bar{\beta}}{\Lambda} = \frac{\sum_{g=1}^{G} \left(\Psi^{*g}, \sum_{g'} (H^{g' \to g} g + P_p^{g' \to g} - L^{g'} \delta_{g'g}) \psi^{g'} \right)}{\sum_{g=1}^{G} \left(\Psi^{*g}, \frac{1}{v^g} \psi^g \right)} \tag{8}$$

$$\frac{\bar{\beta}}{\Lambda} = \sum_{i=1}^{I} \frac{\bar{\beta}_i}{\Lambda} = \sum_{i=1}^{I} \frac{\sum_{g=1}^{G} (\Psi^{*g}, \sum_{g'} P_{d,i}^{g' \to g} \psi^{g'})}{\sum_{g=1}^{G} \left(\Psi^{*g}, \frac{1}{v^g} \psi^g \right)} \tag{9}$$

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

1.3 Solving the IQS System of Equations

Solving for the shape can become expensive, especially in two or three dimensions, and even more so when using the transport equations in lieu of the diffusion equations. Using IQS, one expects the time dependence of the shape to be weaker than that of the flux itself, thus allowing for larger time step sizes in updating the shape. The PRKE equations form a small ODE system and can be solved using a much smaller time step size. In transients where the shape varies much less than the flux, IQS can thus be very computationally effective. The two time scale solution process, a micro scale for the PRKE and a macro scale for the shape, is illustrated in Figure 1.

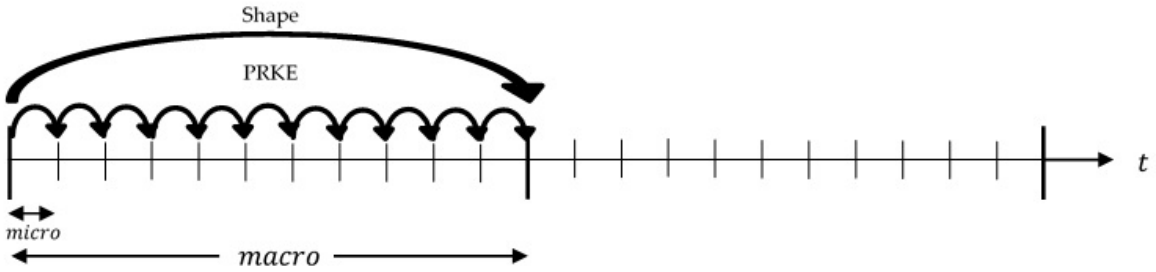


Figure 1: IQS method visualization

By this design, the benefits of IQS is most realized when using time adaptive techniques, one of these is described in Section 1.6. As we noted earlier, shape-PRKE equations are a nonlinear system and thus need to be solved in a iterative manner. Each macro time step can be iterated so the best shape is used to compute power at the micro time steps. 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.
The most convenient method to implement IQS in Rattlesnake is through Picard or fixed-point iterations. Over a given Picard iteration, the shape is evaluated first with the latest available amplitude. The updated shape is employed to compute the PRKE parameters (using a linear interpolation in time) and then the PRKEs are solved using micro time steps. This process is repeated until convergence of the normalization condition. Picard iterations can also be utilized for multiphysics simulations, where neutronics (e.g., IQS) is nonlinearly coupled to thermal-hydraulics, for instance. Thus, inlcuding the IQS nonlinear system solve into the Picard iteration of MOOSE is a straightforward process and is described in Figure 2.

1.4 A Predictor-Corrector version of IQS (IQS P-C)

The Predictor-Corrector (P-C) version of IQS factorizes the flux and derives the PRKE the same way as the standard version, but the solution of the coupled system of equations is different. In the IQS P-C version, the flux equations (not the shape equations) are solved (represented by Equations 1a and 1b) in order to obtain a predicted flux solution. This predicted flux is then converted to shape by rescaling as follows:

$$\varphi_{n+1}^{g} = \underbrace{\phi_{n+1}^{g}}_{\text{predicted}} \frac{K_0}{K_{n+1}} \tag{10}$$

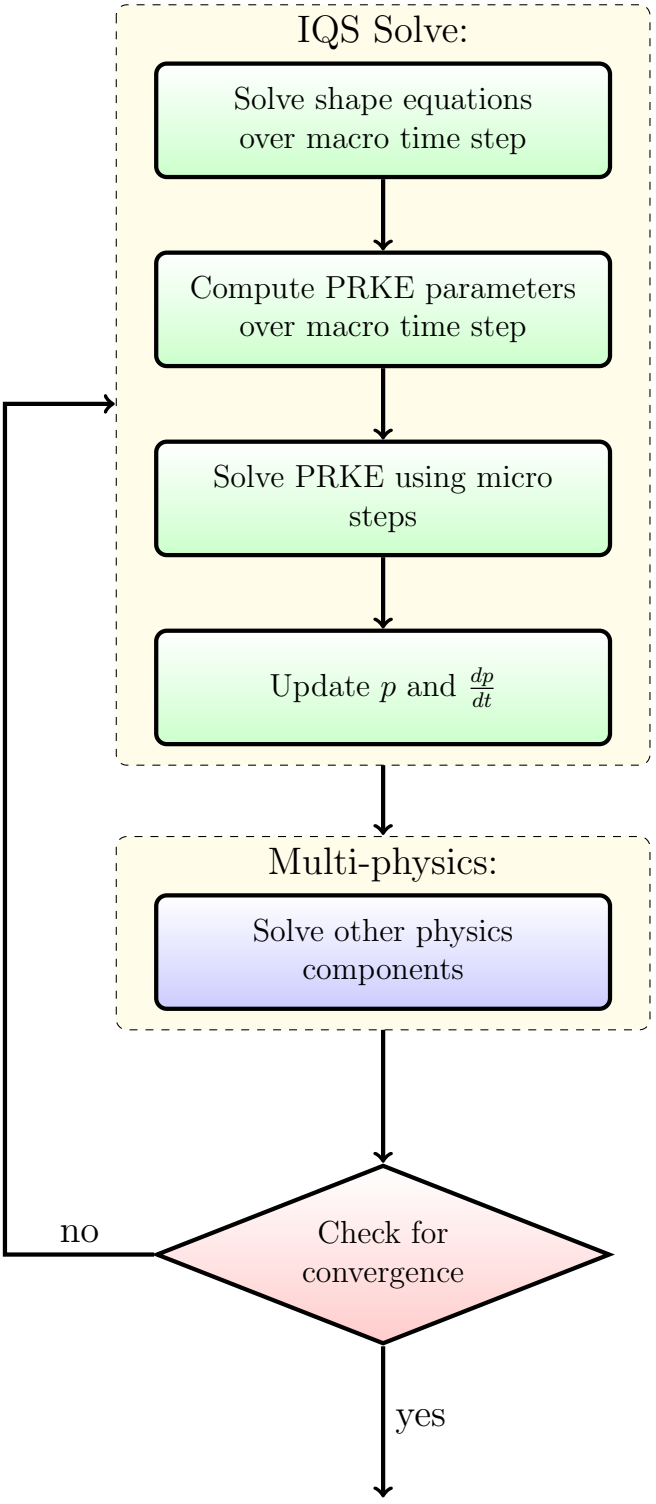


Figure 2: Visualization of Picard iteration process

where the scaling factors are given by

$$K_{n+1} = \sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \phi^g_{n+1} \right) \tag{11}$$

$$K_0 = \sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^g_{n+1} \right) = \sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \phi^g_0 \, . \right) \tag{12}$$

The PRKE parameters are then computed with this shape using Equations 5a - 5c and interpolated over the macro step, then the PRKE ODE system is solved on the micro time scale. With the newly computed amplitude, the shape is rescaled into a flux and the final corrected flux is given by:

$$\underbrace{\phi^g_{n+1}}_{\text{corrected}} = p_{n+1} \times \varphi^g_{n+1} \, . \tag{13}$$

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.

1.5 Precursor Integration

This section presents the time-integration method used to solve coupled IQS shape and precursor equations, represented by Equations 3a and 3b. First, we note that we could keep this system of two time-dependent equations and solve it as a coupled system. However, this is unnecessary and a memory expensive endeavor because the precursor equation is only an ODE and not a PDE. Instead, one may discretize in time the shape equation, which typically requires the knowledge of the precursor concentrations at the end of the time step. This precursor value is taken from the solution, numerical or analytical, of the precursors ODE.

Through prototyping, it has been found that typical implicit nor Crank-Nicholson time discretizations of precursors are not 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 method entails linearly interpolating the fission source ($S_f = \nu \Sigma_f \varphi$) and integrating the resulting continuous function analytically. Equation 14 is the equation to solve for the precursors and Equations 15 and 16 are integrating factors.

$$C^{n+1} = C^n e^{-\lambda \Delta t} + \beta_i \left(a_2 S_f^{n+1} + a_1 S_f^n \right) \tag{14}$$

$$a_1 = \int_{t_n}^{t_{n+1}} \left(\frac{t_{n+1} - t'}{\Delta t} \right) p(t') e^{-\lambda (t_{n+1} - t')} dt' \tag{15}$$

$$a_2 = \int_{t_n}^{t_{n+1}} \left(\frac{t' - t_n}{\Delta t} \right) p(t') e^{-\lambda (t_{n+1} - t')} dt' \tag{16}$$

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.

1.6 Time Adaptation

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

$$e_n = \frac{\left\| \sum_{g=1}^{G} \varphi^g_{\Delta t/2} - \sum_{g=1}^{G} \varphi^g_{\Delta t} \right\|_{L^2}}{\max\left(\left\| \sum_{g=1}^{G} \varphi^g_{\Delta t/2} \right\|_{L^2}, \left\| \sum_{g=1}^{G} \varphi^g_{\Delta t} \right\|_{L^2} \right)} \tag{17}$$

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

$$\Delta t_{new} = S \Delta t \left(\frac{e_{tol}}{e_n} \right)^{\frac{1}{1+q}} , \tag{18}$$

where q is the convergence order of the time integration scheme being used and $S \simeq 0.8$ is a safety factor.
If the error is larger than the user-specified tolerance, the time step is rejected. A new time step size is estimated using Equation 17 as well.
To sum up, each solution undergoes Picard iterations until convergence before the error is calculated; for IQS P-C, each solution is re-scaled by its corresponding amplitude p. Additionally, if the nonlinear iteration does not converge, the entire step doubling process is repeated with half the time original time step.

2 Rattlesnake Implementation

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

2.1 Executioner

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

$$Error_{IQS} = \left| \frac{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^{g,n} \right)}{\sum_{g=1}^{G} \left(\phi^{*g}, \frac{1}{v^g} \varphi^{g,0} \right)} - 1 \right| \tag{19}$$

2.2 Action System

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

$$\frac{\partial}{\partial t}\left(\frac{\varphi^g}{v^g}\right) = \underbrace{\frac{\chi_p^g}{k_{eff}} \sum_{g'=1}^{G} (1-\beta) \nu^{g'} \Sigma_f^{g'} \varphi^{g'}}_{FluxKernel} + \underbrace{\sum_{g' \neq g}^{G} \Sigma_s^{g' \to g} \varphi^{g'}}_{FluxKernel} - \underbrace{\left(-\nabla \cdot D^g \nabla\right) \varphi^g}_{FluxKernel} - \underbrace{\Sigma_r^g \varphi^g}_{FluxKernel}$$
$$\underbrace{- \frac{1}{v^g} \boxed{\overbrace{\frac{1}{p}\frac{dp}{dt}}^{FromExecutioner}} \varphi^g}_{IQSKernel} + \underbrace{\frac{1}{p} \sum_{i=1}^{I} \chi_{d,i}^g \lambda_i C_i}_{ModifiedFluxKernel} \tag{20}$$

To enable Rattlesnake to solve this equation, an IQS removal kernel was created to evaluate $\sum_{g=1}^{G} \frac{1}{v^g} \frac{1}{p} \frac{dp}{dt} \varphi^g$ and added when the IQS executioner is called. Also, the precursor kernel was modified to include the $\frac{1}{p}$ term. Finally, the precursor auxkernel that evaluates Equation 3b using the analytical integration method described in Section 1.5.

2.3 PRKE coefficients

In order to evaluate the PRKE coefficients, defined by Equations 5a - 5c, four postprocessors were created. The parameter calculations were separated by $\frac{\bar{\beta}_i}{\Lambda}$ numerator, $\bar{\lambda}_i$ numerator/-denominator, $\frac{\rho - \bar{\beta}}{\Lambda} / \frac{\bar{\beta}}{\Lambda}$ denominator, and $\frac{\rho - \bar{\beta}}{\Lambda}$ numerator. The first three are relatively simple, only relying on material properties and solution quantities, then computing the elemental integral. The $\frac{\rho - \bar{\beta}}{\Lambda}$ numerator requires the use of the MOOSE `save_in` feature. This feature saves the residual from a calculated kernel or boundary contribution in the shape evaluation to an auxiliary variable. The postprocessor then computes the inner product of this variable and the initial adjoint solution. After each of these postprocessors are evaluated, a user

object pulls together all the values and performs the numerator/denominator divisions. The resulting values are then passed to the executioner for the PRKE solve.

2.4 Other Action Systems

For simplicity, IQS implementation has only been described for CFEM diffusion. However, Rattlesnake has other action systems capable of transient simulation, where IQS can be implemented and be effective. One of these action systems is DFEM diffusion, where the only major difference from CFEM is the diffusion term in Equations 1a and 3a. However, in the derivation for IQS, this term is unaffected between shape and flux evaluation. So saving the residual for this diffusion kernel in the `save_in` variable is the only alteration to this action for IQS to function.

Additional action systems involve transport, but it is evident from Section 1.2 that IQS implementation in these is straightforward as well. The main differences between a diffusion implementation and a transport implementation are outlined below:

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

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

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

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

2.5 Predictor-Corrector Modification

In order to preserve the already implemented standard version of IQS, an option in the IQS executioner was created to specify which method is desired. Because the diffusion solve is flux instead of shape, when predictor-corrector option is specified, the IQS removal kernel ($\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}$) and the modified precursor kernel are bypassed, while all the postprocessors are still executed. However, it is difficult to rescale the flux to shape before the PRKE

parameter postprocessor are executed. So the parameters are computed using the full flux, but amplitude is space independent and comes out of the integrals. As seen in parameter definitions, when shape is replaced with flux, the amplitude comes out of the integral and cancels out. So the conversion of the predicted flux in Equation 10 to shape is unnecessary if the corrected flux is solved with Equation 21. After obtaining the corrected flux, the precursors are re-evaluated using a `EXEC_LINEAR` statement.

$$\underbrace{\phi^g_{n+1}}_{\text{corrected}} = \underbrace{\phi^g_{n+1}}_{\text{predicted}} \frac{K_0}{K_{n+1}} p_{n+1} \tag{21}$$

2.6 Input

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

Below is the syntax for the executioner block for an IQS input file. The `predictor_corrector` logical determines whether to do IQS P-C or regular IQS. The `IQS_error_tol` is the tolerance for the IQS error represented by Equation 19 and will at most `picard_max_its` iterations until convergence. The `prke_scheme` defines the time discretization used for the PRKE solution. Where `RK` is SDIRK33, `CN` is Crank-Nicholson, and `IE` is implicit-Euler.

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

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

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

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

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

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

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

3 RESULTS

This section describes results of examples that test 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. For time adaptation, backward-Euler time discretization was used.

3.1 One-Dimensional Custom Example

The example is very simple and computes quickly, it entails a one dimensional, homogeneous 400 cm slab with a heterogenous perturbation in absorption cross section. Figure 3 shows how the regions of the slab are divided and Table 1 shows the initial material properties. Regions 2, 3, and 4 have slope perturbations at different points in time, Table 2 shows the values of the absorption cross-section in each region at the times of interest. The values of Σ_a between these times of interest are linear interpolations between the given values.

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

Figure 3: 1-D slab region identification

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

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

Table 2: 1-D slab absorption cross-section at times of interest

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

Figure 4 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 5 shows that IQS is not only consistent for this example, but also has a better error constant in the convergence study. Figure 5 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. The IQS performance in this example are not exceptional because it is not an easy problem. As seen in Figure **??**, the shape undergoes significant change through the transient, which requires many shape evaluations to retain accuracy.

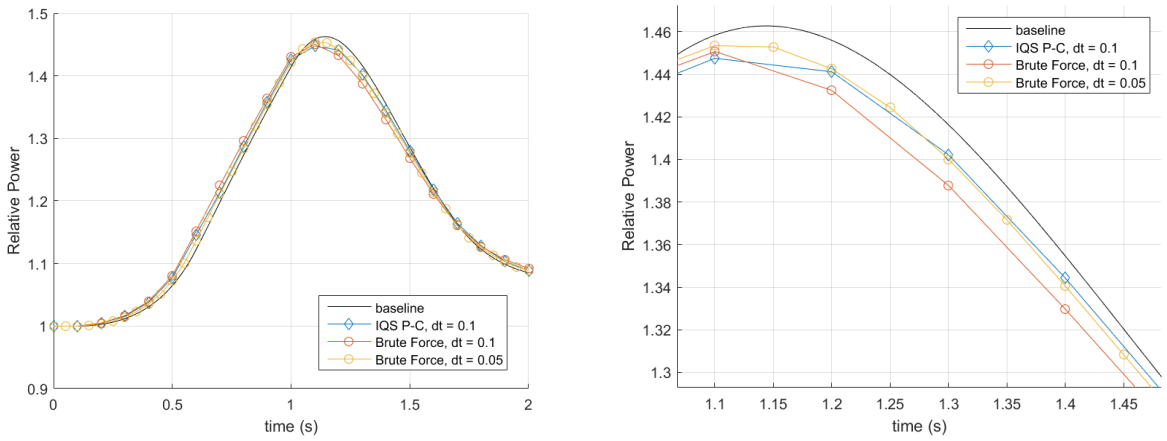


Figure 4: Power level of 1D example

Table 3: 1-D slab step doubling results

		Brute Force			IQS			IQS P-C		
Test	e_{tol}	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

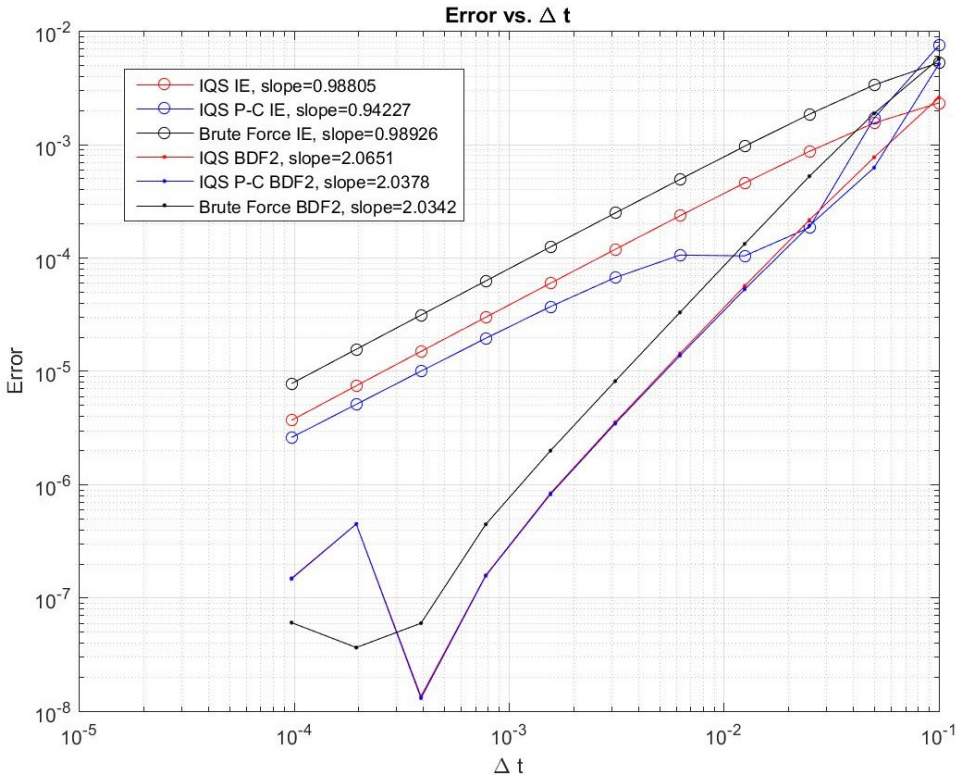


Figure 5: Error convergence comparison of 1D example

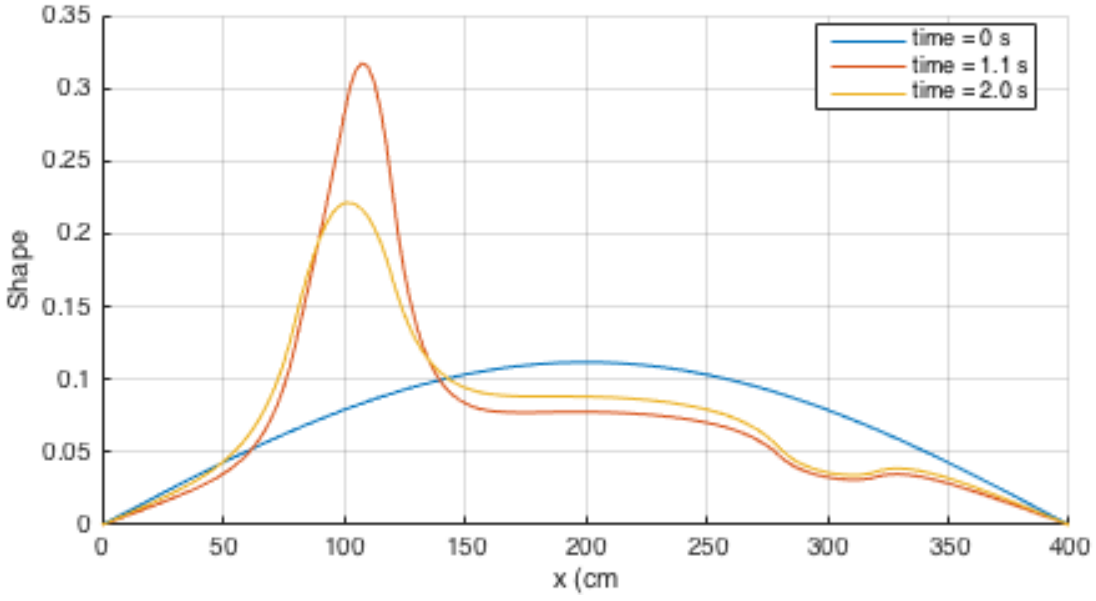


Figure 6: Flux profile of 1D slab at various times

3.2 TWIGL Benchmark

This benchmark problem originates from the Argonne National Lab Benchmark Problem Book [1]. It is a 2D, 2-group reactor core model with no reflector region shown in Figure 7. Table 4 shows the material properties of each fuel region and the ramp perturbation of Material 1.

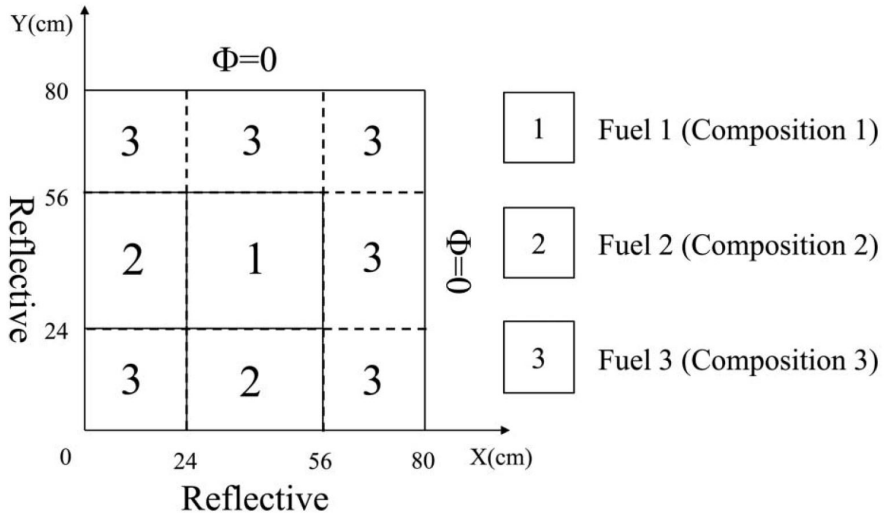


Figure 7: TWIGL benchmark problem description

Table 4: TWIGL benchmark material properties and slope perturbation

Material	Group	$D(cm)$	$\Sigma_a(cm^{-1})$	$\nu\Sigma_f(cm^{-1})$	χ	$\Sigma_s(cm^{-1})$ $g \to 1$	$\Sigma_s(cm^{-1})$ $g \to 2$
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:

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

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

Figure 8 shows the IQS solution as compared with the Brute Force solution. These plots show that IQS is consistent in more complex, higher dimensional problems in Rattlesnake. Finally, Figure 9 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 and Figure 10 show 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_{tol}, while IQS P-C was better with smaller e_{tol}.

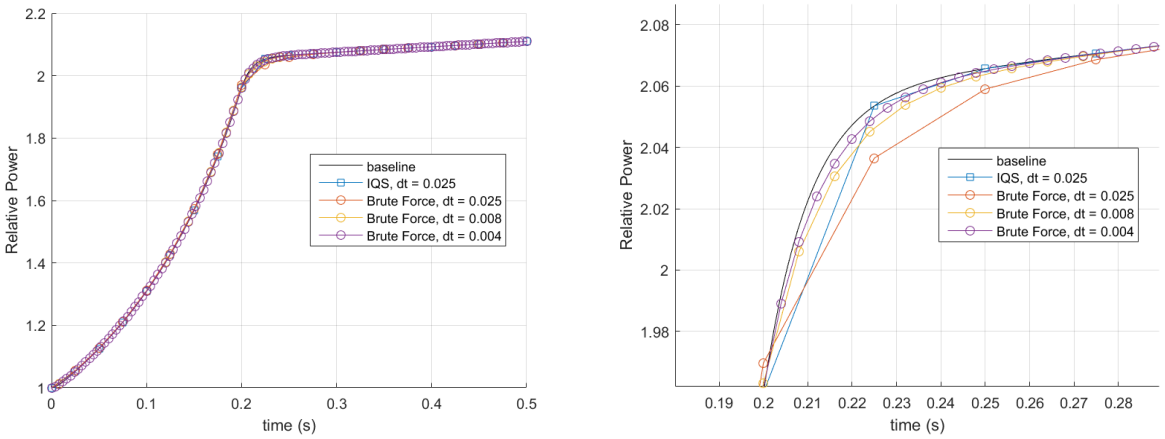


Figure 8: Power level comparison of TWIGL Benchmark

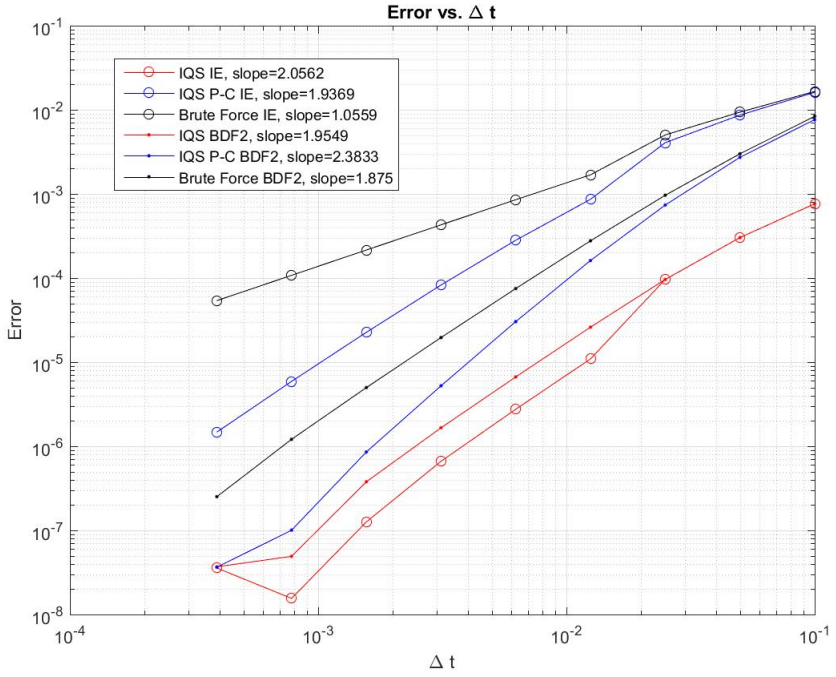


Figure 9: Error convergence comparison of TWIGL Benchmark

Table 5: TWIGL step doubling results

		Brute Force			IQS			IQS P-C		
Test	e_{tol}	Error	Steps	Solves	Error	Steps	Solves	Error	Steps	Solves
1	0.05	0.00012677	9	29	0.03380433	4	20	0.00323100	4	9
2	0.01	3.5555e-05	11	35	0.00166991	5	40	0.00263068	5	12
3	0.005	4.0364e-05	11	31	0.00886584	5	40	0.00160486	6	21
4	0.001	0.00294822	33	122	0.02976305	5	36	1.7527e-05	10	35
5	0.0005	0.00099778	39	131	0.00143781	6	55	1.4185e-05	16	74
6	0.0001	0.00019510	78	236	0.00016175	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

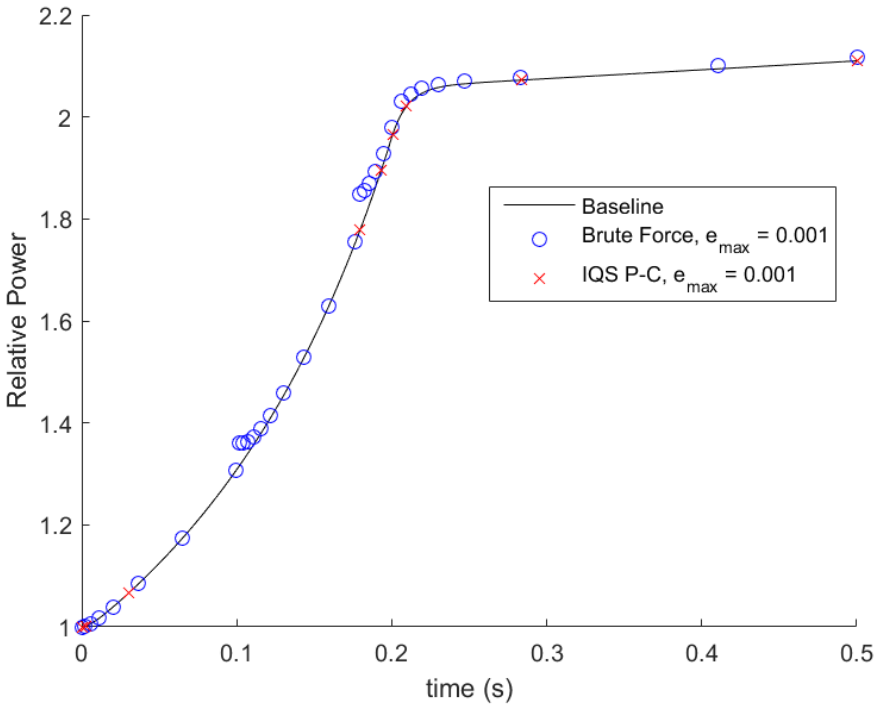


Figure 10: Power level comparison of TWIGL Benchmark with time adaptation

3.3 LRA Benchmark

The LRA benchmark is a two-dimensional, two-group neutron diffusion problem with adiabatic heat-up and Doppler feedback in thermal reactor [1]. It is a super prompt-critical transient. To have better understanding on the cross sections given later, we present the equations here:

$$-\frac{1}{v_1}\frac{\partial \phi_1}{\partial t} = -\nabla\cdot D_1\nabla\phi_1 + (\Sigma_{a,1}+\Sigma_{s,1\to 2})\phi_1 - \nu(1-\beta)S_f - \sum_{i=1}^{2}\lambda_i C_i, \tag{22a}$$

$$-\frac{1}{v_2}\frac{\partial \phi_2}{\partial t} = -\nabla\cdot D_2\nabla\phi_1 + \Sigma_{a,2}\phi_2 - \Sigma_{s,1\to 2}\phi_1, \tag{22b}$$

$$S_f = \sum_{g=1}^{2}\Sigma_{f,g}\phi_g, \tag{22c}$$

$$\frac{\partial C_i}{\partial t} = \nu\beta_i f - \lambda_i C_i, \quad i = 1,2, \tag{22d}$$

$$\frac{\partial T}{\partial t} = \alpha f, \tag{22e}$$

$$\Sigma_{a,1} = \Sigma_{a,1}(\vec{r}, t=0)\left[1+\gamma\left(\sqrt{T}-\sqrt{T_0}\right)\right], \tag{22f}$$

$$P = \kappa S_f, \tag{22g}$$

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_f is the fission reaction rate; P is the power density; T is the temperature; κ is the averaged power released per fission; α is the combination of κ and the specific heat capacity; γ is the Doppler feedback coefficient; $T_0 = T(\vec{r}, t=0)$. The two-group diffusion equation are solved with zero flux boundary conditions on external surfaces, reflecting conditions at symmetry boundaries and steady state initial conditions which are obtained by solving

$$-\nabla\cdot D_1\nabla\phi_1 + (\Sigma_{a,1}+\Sigma_{s,1\to 2})\phi_1 = \frac{1}{k}\sum_{g=1}^{2}\nu\Sigma_{f,g}\phi_g, \tag{23}$$

$$-\nabla\cdot D_2\nabla\phi_1 + \Sigma_{a,2}\phi_2 = \Sigma_{s,1\to 2}\phi_1. \tag{24}$$

$$\tag{25}$$

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

$$\bar{P} \equiv \frac{\int_{V_{core}} P(\vec{r}, t=0)d\vec{r}}{\int_{V_{core}} d\vec{r}}, \tag{26}$$

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 11.

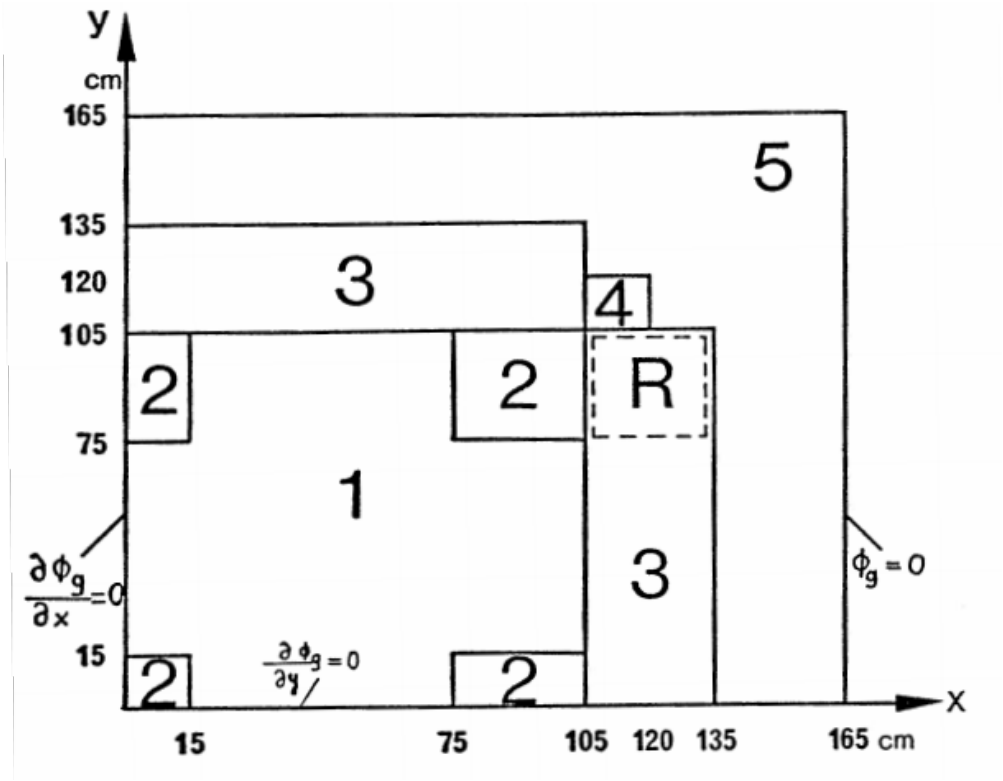


Figure 11: LRA benchmark geometry with region assignment.

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:

$$\Sigma_{a,2}(t) = \Sigma_{a,2}(t=0) \begin{cases} 1 - 0.0606184t, & t \leq 2 \\ 0.8787631, & t > 2 \end{cases} \tag{27}$$

where t is time in seconds.

The Figure 12 plots show the IQS solution as compared with the Brute Force solution. These plots show that IQS is consistent with Doppler feedback problems in Rattlesnake. Figure 13 plots the error convergence of IQS and the Brute Force methods. Table 9 shows the time adaptation performance of brute force and IQS P-C. These results show that IQS, again, performs much better than the brute force method by taking significantly fewer diffusion solves while retaining similar accuracy.

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.

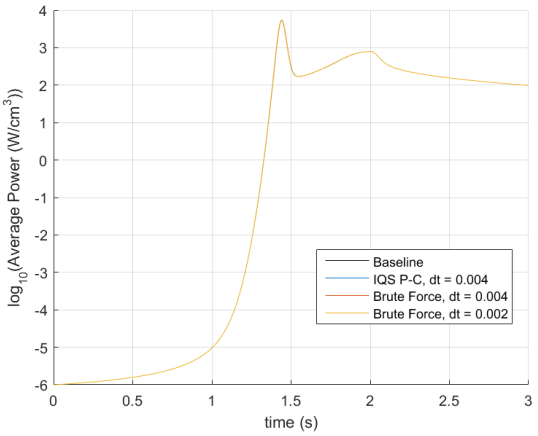
Group i	β_i	λ_i (s^{-1})	$\chi_{d,i,1}$	$\chi_{d,i,2}$
1	0.0054	0.0654	1	0
2	0.001087	1.35	1	0

Table 8: LRA benchmark scalar values.

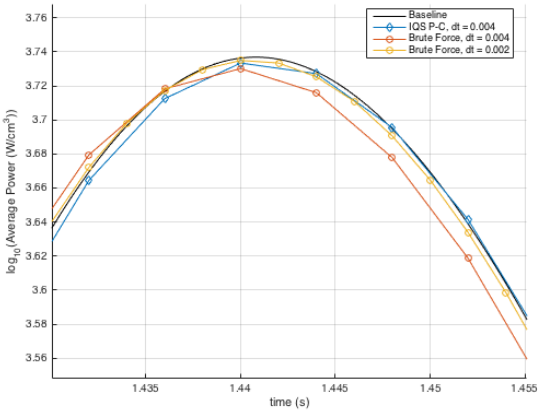
Meaning	Notation	value
Axial buckling for both energy groups	B_g^2	10^{-4} (cm^{-2})
Mean number of neutrons per fission	ν	2.43
Conversion factor	α	3.83×10^{-11} $(K \cdot cm^3)$
Feedback constant	γ	3.034×10^{-3} $(K^{1/2})$
Energy released per fission	κ	3.204×10^{-11} $(J/fission)$
Initial and reference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

Table 9: LRA step doubling results

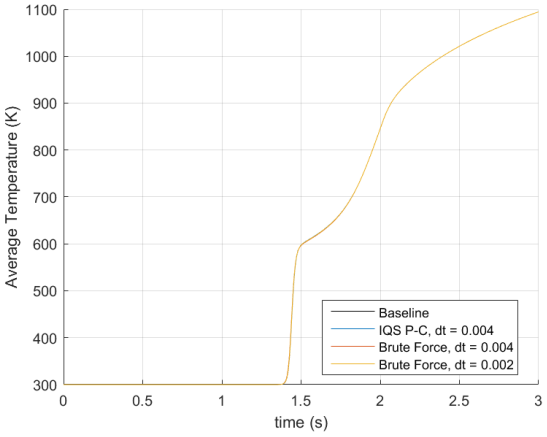
		Brute Force			IQS P-C		
Test	e_{tol}	Error	Steps	Solves	Error	Steps	Solves
1	0.01	0.0437566	145	735	0.143855	105	616
2	0.005	0.0309732	266	948	0.104782	157	814
3	0.001	0.158242	585	2046	0.155676	218	1062
4	0.0005	0.270949	833	2908	0.211994	480	2038
5	0.0001	0.280468	1793	6155	0.0437761	434	1458
6	5e-05	0.212274	2485	8511	0.0385312	542	1719



(a) Power Profile



(b) Power Profile at Peak



(c) Temperature Profile

Figure 12: LRA Benchmark power and temperature comparison

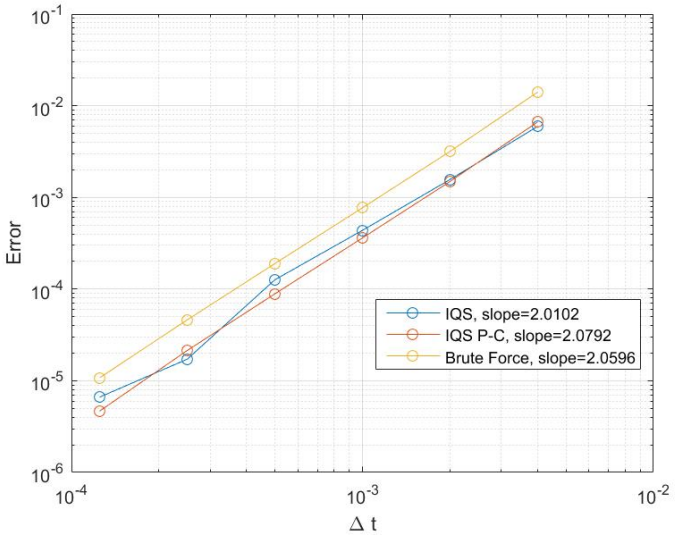


Figure 13: Error convergence comparison of LRA Benchmark

4 Conclusions

The purpose of this report is to briefly review the improved quasi-static method for neutronic transient simulations, to describe its implementation in Rattlesnake, the radiation transport application based on MOOSE, and to show its effectiveness using several test cases. Two IQS implementations are available in Rattlesnake. The first one is the traditional IQS approach, where the shape equation and PRKE are solved as a coupled nonlinear system with Picard iterations. The second approach, IQS P-C, is a predictor-corrector linearization, where the flux is solved and then rescaled using the amplitude obtained through the PRKE (Point Reactor Kinetic Equations).

IQS's implementation in Rattlesnake involves a new kernel, an auxiliary kernel, a new executioner derived from the current transient, four postprocessors, as well as alterations to the action system. The new kernel implements the $\frac{1}{v^g}\frac{1}{p}\frac{dp}{dt}$ term in the shape equation. The auxiliary kernel alters how the precursors are solved, the method is outlined by Section 1.5. The executioner is where the PRKE are evaluated after shape solves are performed. The postprocessors are used to evaluate the parameters for the PRKE and a user object is used to pass them to the executioner. IQS's implementation the action system made sure that all the correct kernels and postprocessors are executed when the IQS executioner is specified in the input.

The IQS implementations have been verified and comparing against the traditional "brute force" temporal discretization of the governing laws. Error convergence plots and time adaptation performance are provided. Three examples were used including a custom one-dimensional example, the TWIGL benchmark, and LRA benchmark. The one-dimensional example shows large shape changes in the flux and proves consistency of the shape evaluation. IQS shows proper error convergence for first and second order time discretization. IQS P-C shows marginal improvement with time adaptation, but traditional IQS shows poor performance. The TWIGL benchmark shows the effectiveness of the PRKE in IQS because there is very little shape change through the transient. IQS shows very satisfactory performance with both constant time steps and time adaptation. The LRA benchmark tests IQS when utilized in the context of a multiphysics simulation. Again, IQS shows good performance by requiring half the time steps as brute force for a given error.

5 Future Work

Now that we are confident in the transient modeling capabilities with IQS, our next step in this research will be to compare results with a direct Rattlesnake solution for TREAT. INL has performed a number of transient tests for various TREAT configurations with thermal feedback. Use of the MAMMOTH reactor physics app will allow coupling of Rattlesnake (including the new IQS routines) with BISON. In this application, we will need to investigate the temporal coupling of BISON with the IQS time stepping process. It may or may not be necessary to couple each micro time step to BISON updates, but further research in this area is needed. Another interest to improve IQS's capability is to develop a smarter time adaptation scheme. The adaptation would entail predicting the perturbation of the shape based on transient changes in material properties. The ability to update PRKE parameters within macro steps may also be a valuable improvement.

References

[1] Argonne Code Center. Benchmark problem book, anl- 7416, suppl. 2. Technical report, Argonne National Laboratory, 1977.

[2] Y. Ban, T. Endo, and A. Yamamoto. A unified approach for numerical calculation of space-dependent kinetic equation. *Journal of Nuclear Science and Technology*, 49:496–515, 2012. doi: 10.1080/00223131.2012.677126. URL http://dx.doi.org/10.1080/00223131.2012.677126.

[3] S. Dulla, E. H. Mund, and P. Ravetto. The quasi-static method revisited. *Progress in Nuclear Energy*, 50(8):908 – 920, 2008.

[4] D. R. Ferguson and K. F. Hansen. Solution of the space dependent reactor kinetics equations in three dimensions. *Nuclear Science and Engineering*, 51:189, 1973.

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

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