NONLINEARLY CONSISTENT SCHEMES FOR COUPLED PROBLEMS IN REACTOR ANALYSIS

A Thesis

by

VIJAY SUBRAMANIAM MAHADEVAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2006

Major Subject: Nuclear Engineering

NONLINEARLY CONSISTENT SCHEMES FOR COUPLED PROBLEMS IN REACTOR ANALYSIS

A Thesis

by

VIJAY SUBRAMANIAM MAHADEVAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Jean C. Ragusa
Committee Members,	Marvin L. Adams
	Jean-Luc Guermond
Head of Department,	William E. Burchill

December 2006

Major Subject: Nuclear Engineering

ABSTRACT

Nonlinearly Consistent Schemes for Coupled Problems in Reactor Analysis.

(December 2006)

Vijay Subramaniam Mahadevan, B.Tech., Bharathidasan University

Chair of Advisory Committee: Dr. Jean C. Ragusa

Conventional coupling paradigms used nowadays to couple various physics components in reactor analysis problems can be inconsistent in their treatment of the nonlinear terms. This leads to usage of smaller time steps to maintain stability and accuracy requirements thereby increasing the computational time. These inconsistencies can be overcome using better approximations to the nonlinear operator in a time stepping strategy to regain the lost accuracy.

This research aims at finding remedies that provide consistent coupling and time stepping strategies with good stability properties and higher orders of accuracy. Consistent coupling strategies, namely predictive and accelerated methods, were introduced for several reactor transient accident problems and the performance was analyzed for a 0-D and 1-D model. The results indicate that consistent approximations can be made to enhance the overall accuracy in conventional codes with such simple non-intrusive techniques.

A detailed analysis of a monoblock coupling strategy using time adaptation was also implemented for several higher order Implicit Runge-Kutta (IRK) schemes. The conclusion from the results indicate that adaptive time stepping provided better accuracy and reliability in the solution fields than constant stepping methods even during discontinuities in the transients. Also, the computational and the total memory requirements for such schemes make them attractive alternatives to be used for conventional coupling codes.

ACKNOWLEDGEMENTS

I offer my sincerest gratitude and appreciation to my advisor, Dr. Jean C. Ragusa, and committee members, Dr. Marvin L. Adams and Dr. Jean-Luc Guermond, without whom this thesis would not have been possible. Thanks especially to Dr. Jean C. Ragusa, who has provided me with valuable guidance, inspiration and time, making himself available whenever needed throughout the duration of the research.

I would also like to acknowledge my family for their understanding and providing continued support during tough times. Last but not the least, I thank my friends from India, College Station, and everywhere in between.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

Reliable and accurate simulations of physical phenomena often require the simultaneous description of several physics components. In most cases, these physics are coupled in a non-linear fashion, making it intricate to find the solution efficiently and accurately. There are several examples of physical phenomena that are non-linearly coupled thereby raising a need to develop stable and accurate schemes to find the solutions. Three examples of such phenomena are:

- Radiation diffusion where the radiation energy is strongly coupled with the temperature field,
- Nuclear reactor analysis where the neutronics and the power are strongly coupled with the thermal-hydraulics field,
- Blood / Vein Fluid Structure Interaction (FSI) problems where fluid and structural vibrations are coupled together.

For the past decade, high fidelity modeling of nonlinear multi-physics problems has been subdivided into several distinct domains of physics and solved individually as mono disciplinary blocks without rigorous coupling between the different physics. Although naïve, this is the most widely used coupling strategy for nonlinear multi-physics. This kind of modeling is based on coupling several existing specialized mono disciplinary codes with a "black-box" strategy, where the input of one code is the output of other, thereby producing solutions that are weakly coupled. This coupling strategy, denoted hereafter as Nonlinearly Inconsistent Coupling (NIC) is based on the explicit linearization of the problem.

This conventional coupling strategy is based on a more commonly known technique called as the operator-splitting method. This method decomposes the system of PDEs into simpler sub-problems and solves the resulting system individually using specialized numerical schemes. This conventional strategy is non-iterative and hence the

---

This thesis follows the style of *Nuclear Science and Engineering*.

nonlinearities in the system due to the coupling are not resolved, reducing the accuracy in the time stepping procedure to first order. Although it does allow parts of the problem to be treated implicitly and others explicitly, the lack of iterations over the nonlinear coupling terms leads to less accurate solutions. Despite these drawbacks, this is still one of the major coupling paradigms used today for solving nonlinear multi physics systems.

The fundamental inefficiency and essential drawback of this strategy is that it does not resolve nonlinearities between physics over a time step and hence is inherently inaccurate. Such an inconsistent treatment of the nonlinear terms usually results in a loss of the convergence order in the final solution and requires the use of excessively small time steps due to stability constraints and loss of order of convergence. The direct implication of using smaller time steps to achieve a reasonable accuracy is that the computations need greater CPU time and resources. The attractive feature of such a naïve coupling strategy is that the legacy of many man-years of mono disciplinary code development and V&V (validation and verification) are preserved.

As mentioned before, nuclear reactor analysis is a good example of highly non-linear, coupled multi physics problem and the nonlinearities are embedded at the heart of reactor design, analysis and safety calculations. It is then of prime importance to develop coupling strategies that can produce highly accurate solutions even in the complex scenarios usually encountered in reactor analysis safety. Often physical phenomena, such as the ones found in reactor accidents, involve rapidly varying transients which are represented by a stiff system of differential equations. Stiff problems are characterized by solutions having fast varying modes together with slower modes, requiring time integrators that can handle such disparate time scales. Such stiff problems necessitate the use of implicit time discretization for stability reasons, as will be seen in the next chapter.

The physics of nuclear systems are usually sub-divided into 3 domains for extensive and rigorous calculations based on the nature of the physics. They are given as:

- **Neutronics** - Deals with the neutron population distribution in the reactor core
- **Thermal hydraulics** - Deals with the calculation of fluid density and temperature fields in the coolant
- **Heat transfer** - Deals with the temperature fields within the nuclear fuel

It needs to be noted that even though these domains 'seem' distinct, these various physics are intertwined and rely heavily on the solution field of one another. The coupling of the various physics involved in reactor analysis hence requires stable and accurate schemes to yield reliable results and the nonlinearly inconsistent coupling (NIC) strategy will not yield very accurate results for complex scenarios such as accident behaviors, a Loss of Coolant Accidents (LOCA), a Main Steam Line Break (MSLB) and other safety analysis calculations performed in nuclear reactor analysis.

Present and future simulations of nuclear reactors, either for the design of new cores or for core performance analysis will rely increasingly on multi-dimensional, multi-physics computations. These coupled simulations should be performed in amenable wall-clock times and should yield accurate solution. In that respect, the fundamental inefficiency of the current nonlinearly inconsistent coupling must be solved in an efficient way without affecting the current coupling of the existing codes, if possible.

The current research devises simple, robust, and CPU-effective acceleration methods for the nonlinear coupling of neutronics and thermal-hydraulics so that the coupled simulations of transients and accidents can be performed with better confidence and with smaller computation effort.

On the whole, the aim of the current research is to strive to do the following:

- *A*nalyze the current nonlinearly inconsistent coupling procedures and assess their stability and accuracy,
- *P*ropose improvements[1] to the nonlinearly inconsistent operator-splitting coupling paradigm in order to achieve a Nonlinearly Consistent Coupling (NCC) scheme,
- *D*evelop stable schemes that will have minimal impact on existing mono-disciplinary codes but improve the nonlinear coupling by slightly modifying the interfaces between the codes (Non-intrusive modification),
- *V*alidate the accuracy improvements on real-life physical scenarios; Core transients[2], MSLB[3] benchmarks.

Based on the above overall goals, this research thesis is then organized as follows:

- Review the coupling strategies currently in use and put forth their deficiencies.

- Explain the basic theory for our new coupling strategy and how the deficiencies in current schemes can be overcome
- Discuss some stable numerical schemes that correct the flaws in the current strategy and aid in achieving a higher order of convergence preserving stability for higher time steps
- Apply these schemes to a nuclear reactor analysis transient problem, in both 0-D and 1-D scenarios
- Measure and analyze the effectiveness of the proposed methods by discussing the implication of the results.

Further in this chapter, a brief introduction to the numerical analysis of schemes being performed in this research will be discussed.

I.1 Nonlinearities and time discretization schemes

Multi-physical problems in mathematical terms are a coupled set of partial differential equations requiring both spatial and temporal discretization. After spatial discretization, this general system of nonlinear ordinary differential equations can be written as

$$\begin{aligned} y_1' &= f_1(y_1, y_2, \ldots y_n) \\ y_2' &= f_2(y_1, y_2, \ldots y_n) \\ &\vdots \\ y_3' &= f_n(y_1, y_2, \ldots y_n) \end{aligned} \tag{1.1}$$

Where f_i are nonlinear differential operator representing physical conservation laws, equilibrium conditions etc., and y_i represent the unknown field variables.

A single set of ordinary differential equations can then be formulated as an Initial Value Problem (IVP)

$$y' = Ly + N(y) + b = f(y) \text{ and } y(0)=y_0 \tag{1.2}$$

where y is the vector of unknowns,

y' represents time derivative of the unknown vector,

L is a matrix representing a strictly linear operator, and

$N(y)$ is a vector representing a nonlinear operator

The matrices L and $N(y)$ may result from spatial discretization of the partial differential equations in higher dimensions or due to coupling between the sub-problems obtained in an operator splitting methodology.

For many physical systems, the nonlinear vector function $N(y)$ may be factorized as $G(y)y$, where $G(y)$ is a matrix whose elements depend on y. Using this factorization, Equation (1.2) can be re-cast as

$$y' = Ly + G(y)y + b \tag{1.3}$$

Most coupled phenomena can be cast in the form of Equation (1.3). Generally speaking, Equation (1.3) is solved using many mature time integration techniques. Some of the more popular, time discretization schemes are the Backward Euler, Crank-Nicholson, Implicit Runge-Kutta (IRK), linearly implicit Rosenbrock-Wanner family of methods (ROW) and Adams-Moulton (implicit multi-step) methods with either a constant time-step strategy or a variable step control with step-doubling or embedded strategy. Schemes of various orders of accuracy can be obtained depending on the implicit procedure chosen to solve the system of nonlinear equations.

For instance, in order to advance the numerical solution in time from time t^n to time t^{n+1}, the very popular θ discretization method can be used. When this scheme is applied to Equation (1.3), we obtain the following equation.

$$\frac{y^{n+1} - y^n}{t^{n+1} - t^n} = \theta\,[L^{n+1}y^{n+1} + N(y^{n+1}) + b^{n+1}] + (1-\theta)\,[L^n y^n + N(y^n) + b^n] \tag{1.4}$$

with $\theta \subset [0; 1]$. For $\theta = 1$, we obtain the backward Euler scheme (first-order in time if enough regularity is present in our problem) and for $\theta = 1/2$, we obtain the Crank-Nicholson scheme (second-order in time if enough regularity is present in our problem).

Other time discretization schemes can be considered here instead of the θ-Scheme and a similar procedure can be followed to discretize Equation (1.2) depending on the need for higher orders of accuracy or increased stability as appropriate for the problem under investigation. But it is important to remember that the goal of this research is to improve the treatment of the nonlinear term $N(y^{n+1})$ in Equation (1.4) and discretize Equation (1.2) in a consistent fashion so as to preserve the order of accuracy of the solution scheme.

I.2 Current coupling schemes

In the domain of reactor physics, several mono-disciplinary codes have been developed for decades and their reliability are established. It is then of paramount importance that the research be aimed at developing algorithms which will re-use as much as possible from existing codes, rather than proposing the merging of codes into one monolithic meta-code. Such an effort will then produce more accurate results based on the existing codes without a need to replicate the code development cycle again and preserving all the time spent on these codes. It is then important to understand the "black-box" approach that is usually employed for mimicking the coupling of multi-physics and the inherent deficiencies that are present with such a strategy before we delve in devising better strategies to solve coupled systems.

A simple schematic showing the first order operator-splitting model for a coupled, two-physics system is shown below. So, for a transient problem, the solution procedure at every time step can be described by the following figure.

In current coupling schemes as shown in Figure (1), the mono-disciplinary codes solve a given physics, and data between codes is exchanged at *rendezvous* points, usually at the end of each time interval. The concept of this model springs from the fact that the different physics have previously already been solved using dedicated, specialized codes and using suitable message passing paradigms such as Parallel Virtual Machine (PVM) or a Message Passing Interface (MPI), these codes can be made to communicate with one another to solve (inconsistently) the coupled physics problem.

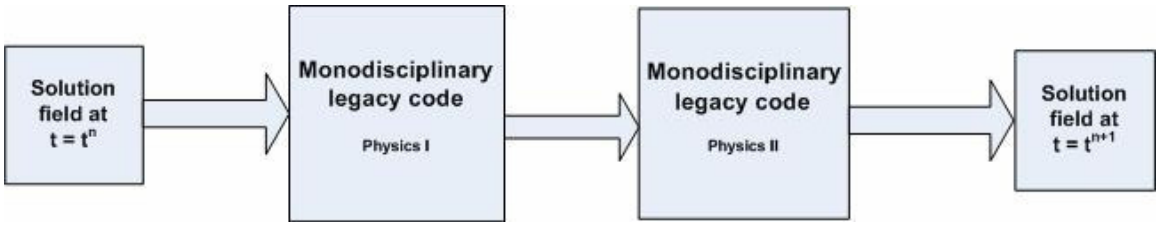

Figure 1: Operator-split model

In the operator-split model, the mono-disciplinary codes that are coupled together, work on the principle of a lagged updation procedure. The solution from one physics code becomes the input of another physics code, without any iteration over the time step

or extrapolation in the exchanged values. In these schemes, the nonlinear problem is linearized and the nonlinearities over the time step are never converged and hence the terminology 'Weakly coupled schemes' is also usually used to refer to such a strategy.

To be more specific, a weakly coupled system could be characterized by the condition

$$\left|\frac{\partial f_i}{\partial x_j}\right| \leq C_i \left|\frac{\partial f_i}{\partial x_i}\right| \tag{1.5}$$

for some $C_i << 1$ and for every $j \neq i$ in the neighborhood of the exact solution. In this case, the value of f_i depends mostly on x_i and even a significant change in x_j causes only a small change in f_i. Even though this might not be true physically but the way the problem is solved in a weakly coupled system, f_i is not sensitive enough to x_j or in other words, the system is diagonally dominant. This is due to the aforementioned reason that over a time step, the coupling between the different physics is not completely resolved and hence the primary operator f_i in a physics depends strongly only on the field variable x_i.

The idea behind this weak coupling is to use an inconsistent approximation for the nonlinear term at the end of the time step, $N(y^{n+1})$. This approximation consists in treating $N(y^{n+1})$ explicitly in time as follows, leading to the following substitution :

$$N(y^{n+1}) = N(y^n) + \frac{\partial N(y^n)}{\partial y}(y^{n+1} - y^n) \approx N(y^n) + O(\delta y) \tag{1.6}$$

or when using the factorized expression for $N(y)$ as given in Equation (1.3),

$$N(y^{n+1}) = G(y^{n+1})\, y^{n+1} \approx G(y^n)\, y^{n+1} + O(\delta y) \tag{1.7}$$

The nonlinear system has therefore been explicitly linearized but at the cost of a gross approximation wherein the nonlinearities are treated explicitly in time. Another way of looking at this approximation is to view it as a first-order prediction as below.

$$y^{n+1} \approx y^{n+1,P} = y^n + O(\Delta t) \tag{1.8}$$

where the superscript P denotes the prediction. It is clear from Equation (1.8) that such a scheme will only lead to a global convergence rate of $O(\Delta t)$, no matter what time integration techniques are used in each of the separate physics. We already emphasize that nonlinearly inconsistent schemes will require small time steps to achieve reasonable accuracy due to their convergence of $O(\Delta t)$, hence taxing the CPU time and slowing down the engineering analysis work.

It is also important to know that there are several small digressions in the actual implementation of the weakly coupled schemes. Some of the various cases when two physics components, say Neutronics and Thermal-hydraulics are coupled, are shown below.

- Both codes advance simultaneously from time t^n to t^{n+1}, and the data is exchanged at the end of the time step (simultaneous update procedure) which can be represented by a block Jacobi scheme
- Or one code advances first and sends its results to the second code before the latter starts the calculations for the time-step (serial staggered procedures) which can be represented by a block Gauss-Seidel scheme

The schemes above can be extended to couple more than 2 physics in a similar fashion. A descriptive figure to explain the above variations in the weakly coupled schemes is shown below for a Neutronics/Thermal-Hydraulics coupled nuclear reactor system.

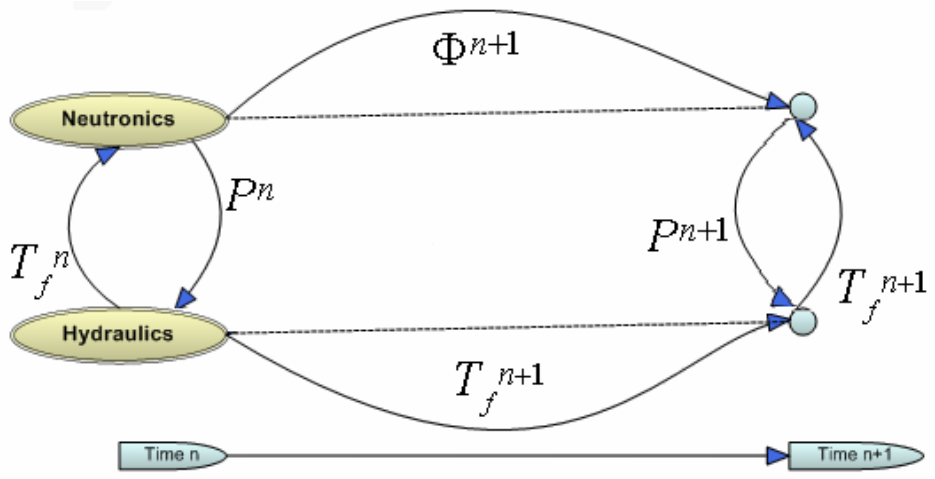

Figure 2: Conventional simultaneous update procedure

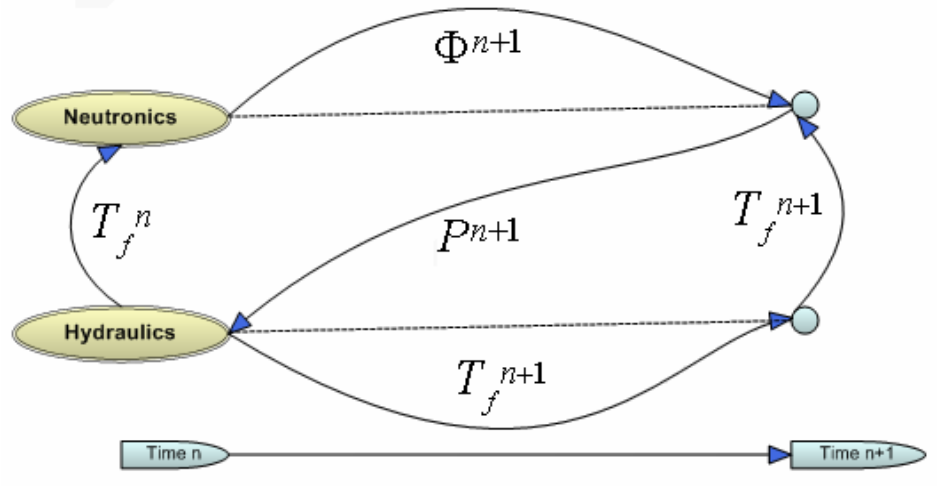

Figure 3: Conventional serial staggered procedure (Neutronics first)

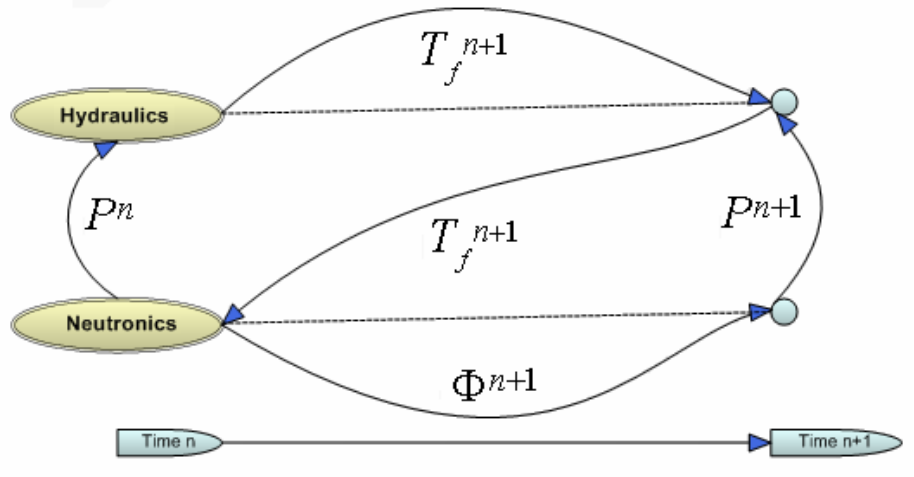

Figure 4: Conventional serial staggered procedure (Thermal hydraulics first)

The inaccuracy in this type of conventional operator-splitting approaches which do not converge the nonlinearities at every time step, necessitating the use of very small time steps, can be analyzed using an illustrative example shown below.

I.3 An illustrative example

At this point in the discussion, we are going to present a simple derivation to illustrate the issues that were discussed earlier on in this section.

Let *y* be the vector of unknowns in a nonlinear multi-physics problem. The system of nonlinear differential equations can be written as follows:

$$y' = f(t,y) \tag{1.9}$$

where f is a nonlinear vector-function.

We can then split the standard problem into a linear term and a nonlinear term similar to Equation (1.3) as follows:

$$y' = f(t,y) = L(t)\, y + N(t,y) \tag{1.10}$$

For illustration, applying the implicit Crank-Nicolson discretization scheme to Equation (1.10),

$$y^{n+1} = y^n + \frac{\Delta t}{2}[f(t^{n+1}, y^{n+1}) + f(t^n, y^n)] + O(\Delta t^3)$$

$$[I - \frac{\Delta t}{2} L(t^{n+1})]y^{n+1} = [I + \frac{\Delta t}{2} L(t^n)]y^n + \frac{\Delta t}{2}[N(t^{n+1}, y^{n+1}) + N(t^n, y^n)] + O(\Delta t^3) \tag{1.11}$$

Equation (1.11) is nonlinear in the unknown y^{n+1} and has to be solved or approximated to some known precision to get the correct and accurate solution. Usually, a time explicit linearization of the nonlinear term is performed using a crude approximation

$$N(y^{n+1}) = N(y^n) + O(\Delta t)$$

yielding

$$[I - \frac{\Delta t}{2} L(t^{n+1})]y^{n+1} = [I + \frac{\Delta t}{2} L(t^n)]y^n + \frac{\Delta t}{2}[N(t^{n+1}, y^n) + N(t^n, y^n)] + O(\Delta t^2) \tag{1.12}$$

Typically, for linear system of differential equations, the above implicit Crank-Nicholson scheme yields $O(\Delta t^2)$ globally. Unfortunately, the approximation in Equation (1.12) is only $O(\Delta t)$ globally in time and consequently reduces the overall global accuracy of the C.N scheme to $O(\Delta t)$ from $O(\Delta t^2)$. This observation is quite general and holds for any time integration technique for nonlinear systems for which the nonlinear terms have been explicitly linearized using the approximation in Equation (1.6), which is what most existing codes use.

I.4 Drawbacks and remedies

The standalone mono-disciplinary codes provide higher global convergence accuracy order in time while solving the system of nonlinear ODE for a decoupled system. But the above illustration elucidates that the conventional operator-splitting methodology yields only $O(\Delta t)$ global accuracy in the solution fields. Hence, higher global orders of accuracy

e.g., $O(\Delta t^2)$ or higher, could improve the efficiency and performance of current schemes by many fold. Also higher order accurate schemes are more intricate and can be very complicated to code.

Unfortunately, end users are not necessarily aware of the huge price paid for the loss of one or more orders of convergence. The direct implication of the loss of an order of accuracy is that we need smaller time steps to reach reasonable levels of accuracy and to ensure that the scheme remains stable. Also the calculation and computation of such nonlinearly inconsistent schemes would then require comparatively more CPU time and resources to obtain reasonable levels of accuracy. This delays the actual engineering analysis and design process which are heavily dependent on the solution from such coupled systems for short transients or accident simulations.

To workaround the problems we have outlined, simple and robust numerical schemes need to be derived which will be stable and accurate even for larger time steps in comparison to the current schemes. It has now become clear that the nonlinear equations resulting from the time discretization of nonlinear multi-physics systems should be solved or approximated in an efficient way. This nonlinear solve could be performed in different ways by using better approximations which will be covered in the following section. Also, if stable numerical schemes can be combined with adaptive step size control strategies, then the performance and the desired accuracy in the solution can be controlled and improved. Based on some previous work[4] even explicit time stepping would be a viable option if the necessary extended stability conditions and enough damping is included in the scheme. Such explicit schemes will eliminate the need to solve the nonlinear system iteratively and hence has the advantage of reducing the total number of function evaluations at the cost of few stabilizing small steps.

Going further, with the observations made and understanding gained from the current schemes, we will look at devising new robust, stable and accurate schemes as mentioned before and look at the properties and implementation of each method in the next section.

CHAPTER II

STABLE AND ACCURATE HIGHER ORDER NUMERICAL SCHEMES

Nonlinear, coupled systems, as mentioned before, require stable schemes to yield accurate results. Optimizing such schemes to minimize the time taken for computation involves analysis and modification of existing schemes so as to preserve the effort towards the mature legacy codes that are primarily used to solve the mono-physics blocks. There are several parameters that need to be born in mind before deciding on the scheme to be used for computation. They are as follows:

1. **Stability** – The numerical scheme under consideration should be unconditionally stable for all time steps so that we are not constrained by the step size used. This is important to reduce the overall computation time in obtaining the solution fields with a prescribed accuracy and limit the accumulation of round-off errors.
2. **Order of accuracy** – The numerical scheme should have a high order of convergence which will enable the usage of larger time steps when necessary, given that the scheme is within the stability region. Higher order schemes can then be used to obtain the solution fields with a specified accuracy in significantly fewer steps than the lower order schemes. Also, the coupling scheme should not cause any loss of accuracy when linking the different physics solvers together while obtaining the solution fields.
3. **Computational cost** – The numerical scheme should require minimal computational effort to evaluate the linear and the nonlinear operators at each time step as possible in order to reduce the calculation time. For instance, a higher order scheme requiring many 10000 flops/step will be less efficient than a slightly lower order scheme with 5000 flops/step. This is comparatively a minor factor but still plays an important role in deciding the best numerical scheme for the proposed problem.

4. **Implementation ease** – The numerical scheme should not be too cumbersome to implement since the aim of coupling different mono-physics codes is primarily to improve the interface driver. Hence complex schemes requiring extensive man hours to program, test and verify the changes are disadvantageous from a practical stand point.

Bearing these parameters in mind, numerical analysis was done on various popular and mature methods that have been used previously in different coupled physics scenarios by Robert Lowrie[5] and Knoll et al.,[6].

Based on preliminary observations of the numerical schemes on a nonlinear problem of the form shown in Equation (1.3), the variable θ time discretization using constant stepping procedure along with several higher order explicit and implicit Runge-Kutta embedded adaptive time control schemes were chosen for solving the multi-physics problem in nuclear core analysis. We shall discuss about these methods in the next sections of this chapter. But first, let us review the concepts of accuracy and stability of numerical schemes.

II.1 Numerical solution procedures for solving nonlinear systems

The current techniques for nonlinear dynamical systems offer insights into the numerical behavior of the systems. This result is due to the combination of a well posed initial continuous differential problem and a discrete numerical scheme employed to resolve it. The numerical properties of time-stepping algorithms require an extension to classical analysis performed in the linear domain, especially when we deal with applications to a large class of nonlinear problems arising in coupled physics. Based on the consolidated work concerning the theory of nonlinear properties of numerical methods by several researchers like Hairer E, Wanner G on solving non-stiff and stiff ODEs[7,8], Dekker K, Verver J.D on Runge-Kutta solvers[9], the non-linear problem can be linearized about a given point and analyzed to obtain a global evaluation of the reliability and capability of the selected numerical scheme. Also, the restrictive nature of the nonlinear stability analysis procedures can limit discretization schemes that are not suited for stiff problems to use large time step values.

For illustrative purposes, let us consider a general nonlinear problem of the following form

$$\frac{dy}{dt} = Ly + N(y) + b(t) = f(y,t) \tag{2.1}$$

where $f(y,t)$ is continuous in the problem domain D_f

and L, N(y) are the linear and nonlinear operators of the problem.

A numerical method may be applied in order to solve the nonlinear equation which in general leads to a family of equations $f(y_h) = 0$. The subscript h indicates the dependence on a small parameter such as mesh size and that $h \in (0, \infty)$. Then, we can define the convergence of the approximating solution y_h* to y*, as

$$\lim_{h \to 0} \| y^* - y_h^* \| = 0 \tag{2.2}$$

We are interested in making the error in the solution $\varepsilon = \| y^* - y^n{}_h \|$ minimal, in as few steps as possible for a given step size h. To accomplish this, we must numerically analyze the schemes under consideration and determine the *accuracy* and *stability* of our approximation schemes.

II.2 Stability

A numerical scheme is considered to be stable if applied for the linear IVP $y'=\lambda y$ with $\lambda<0$, for any step size h, it results in a non-increasing sequence of approximations i.e., $y^{n+1} \leq y^n$. In other words, if the true solution remains bounded then the numerical solution should be bounded for all step sizes.

In linear stability analysis, the ODE of the form $y' = \lambda y$ where $\lambda < 0$ is analyzed to find the stability region of a given numerical scheme. In the case of a nonlinear problem of the form given in Equation (2.1), we can linearize f in a neighborhood of the solution φ(t) as follows

$$y'(t) = f(t, \varphi(t)) + \frac{\partial f}{\partial y}(t, \varphi(t))(y(t) - \varphi(t)) + \ldots \tag{2.3}$$

Now let us introduce $y(t) - \varphi(t) = \bar{y}(t)$ to then get

$$\bar{y}'(t) = \frac{\partial f}{\partial y}(t, \varphi(t))\bar{y}(t) + \ldots = J(t)\bar{y}(t) + \ldots \tag{2.4}$$

At first approximation we consider the Jacobian $J(t)$ as constant and neglect the higher order error terms. Omitting the bars, we then arrive at,

$$y'(t) = Jy(t) \tag{2.5}$$

Now, we can apply the numerical scheme of interest to Equation (2.5) to obtain an equation of the form

$$y_{n+1} = R(hJ)\, y_n \tag{2.6}$$

where the function $R(hJ)$ is the *stability function* of the numerical method.

For illustration, let us apply the implicit Backward Euler scheme to Equation (2.5) and analyze the stability of the system.

$$y_{n+1} = y_n + hJy_{n+1} \rightarrow (I - hJ)\, y_{n+1} = y_n \rightarrow y_{n+1} = R(hJ)\, y_n \tag{2.7}$$

Where $R(hJ) = (I - hJ)^{-1}$. To have a stable numerical solution, $|R(hJ)|<1$. For this to be possible, the following condition needs to be satisfied.

$$\frac{1}{1 - h\,|\,\lambda_{\max}\,|} < 1 \tag{2.8}$$

Where $|\lambda_{max}|$ is the largest Eigenvalue of the Jacobian matrix

A similar procedure can be applied when using different numerical schemes to analyze and find the stability regions. By definition, a numerical method is called *A-stable* if and only if the stability domain satisfies

$$S \supset C^{-} = \{\; z = h\lambda_{max};\;\; \mathrm{Re}\, z \le 0 \;\} \tag{2.9}$$

For methods with this property the step size is never restricted by stability regardless of the stiffness ratio $S=|\lambda_{max}/\lambda_{min}|$, where λ is any Eigenvalue of $(I - hJ)$. Taking *A*-stability into account, one can develop several different schemes that are implicit since only these methods have a proper rational stability function.

The stability analysis of explicit schemes have been performed by Dahlquist[10] and has been proven that these methods have a bounded stability region and hence do not satisfy *A*-stability. Such conditionally stable methods require extensive step-size restrictions to resolve all the modes in a problem, especially when it is stiff. On the other hand, the *A*-stable implicit schemes solve stiff differential equations efficiently without any step-size restrictions but require solution of nonlinear algebraic equations at each step repeatedly, which results in an increase of computational cost. Hence, if the solution

remains well behaved for arbitrarily large values of time steps, ample CPU time can be saved by using the implicit methods instead of the conditionally stable explicit schemes that require very small step-sizes throughout the transient period.

In some situations, it may also be desirable to damp the very stiff components of the numerical solution. This leads to the concept of *L*-stability of numerical schemes apart from *A*-stability, which requires that $R(z) \to 0$ as $z \to -\infty$. The L-stability is important while solving fast changing transients, especially in stiff systems where the fast modes have to be damped to avoid unwanted oscillations.

II.2.1 Stability of conventional schemes

The introduction provided in Chapter I for the operator splitting technique offers several advantages over solving the monolithic block system of equations as a whole. If the split is accomplished as shown in Figures 2, 3 or 4, it provides good flexibility in the staggered and non-staggered procedures to use implicit or explicit schemes for either or all the physics. But, an important issue that needs to be addressed in such an operator split partitioning is the stability of the partitioned system.

For illustration, let us assume that the system of nonlinear equation representing two coupled physics is split as

$$\frac{dY_1}{dt} = f_1(t, Y_1, Y_2), \text{ and } \frac{dY_2}{dt} = f_2(t, Y_1, Y_2) \tag{2.10}$$

Now let us apply the implicit C.N scheme to the above system of equations.

$$Y_{i,n+1} = Y_n + \frac{h}{2}[f_i(t_n, Y_{1,n}, Y_{2,n}) + f_i(t_{n+1}, Y_{1,n+1}, Y_{2,n+1})] + O(h^3)\text{, } i=1,2 \tag{2.11}$$

To solve for the solution fields $Y_{i,n+1}$ would require solving the nonlinear set of equations. Considering the Taylor series expansion of $f_i(t_{n+1}, y_{i,n+1})$ about the known solution $Y_{i,n}$,

$$f_i(t_{n+1}, Y_{n+1}) = f_i(t_{n+1}, Y_n) + \frac{\partial f_i(Y_n)}{\partial Y_i}(Y_{i,n+1} - Y_{i,n}) + \sum_{\substack{j=1 \\ j \neq i}}^{2} \frac{\partial f_i(Y_n)}{\partial Y_j}(Y_{j,n+1} - Y_{j,n}) + \ldots$$

But from the general system of equation given above, it can be seen that $Y_{i,n+1} - Y_{i,n} = O(h)$; Then, replacing $f_i(t_{n+1}, Y_{i,n+1})$ yields

$$
\begin{aligned}
Y_{1,n+1} &= Y_{1,n} + \frac{h}{2}\left[\begin{array}{l} f_1(t_{n+1}, Y_{1,n}, Y_{2,n}) + f_1(t_n, Y_{1,n}, Y_{2,n}) + \\ \dfrac{\partial f_1(Y_{1,n}, Y_{2,n})}{\partial Y_1}(Y_{1,n+1} - Y_{1,n}) + \dfrac{\partial f_1(Y_{1,n}, Y_{2,n})}{\partial Y_2}(Y_{2,n+1} - Y_{2,n}) \end{array}\right] + O(h^3) \\
Y_{2,n+1} &= Y_{2,n} + \frac{h}{2}\left[\begin{array}{l} f_2(t_{n+1}, Y_{1,n}, Y_{2,n}) + f_2(t_n, Y_{1,n}, Y_{2,n}) + \\ \dfrac{\partial f_2(Y_{1,n}, Y_{2,n})}{\partial Y_1}(Y_{1,n+1} - Y_{1,n}) + \dfrac{\partial f_2(Y_{1,n}, Y_{2,n})}{\partial Y_2}(Y_{2,n+1} - Y_{2,n}) \end{array}\right] + O(h^3)
\end{aligned}
\tag{2.12}
$$

After a little rearranging, the above equation in the general form becomes:

$$
Y_{i,n+1} = Y_{i,n} + \frac{h}{2}\left[I - \frac{h}{2}\frac{\partial f_i(Y_n)}{\partial Y_i}\right]^{-1}\left[\begin{array}{l} f_i(t_{n+1}, Y_n) + f_i(t_n, Y_n) + \\ \displaystyle\sum_{\substack{j=1 \\ j\neq i}}^{2} \frac{\partial f_i(Y_n)}{\partial Y_j}(Y_{j,n+1} - Y_{j,n}) \end{array}\right] + O(h^3)
$$

$$
Y_{i,n+1} = Y_{i,n} + \gamma h\left[I - \gamma h J_{i,i}\right]^{-1}\left[f_i(t_{n+1}, Y_n) + f_i(t_n, Y_n) + \sum_{\substack{j=1 \\ j\neq i}}^{2} J_{i,j}\delta Y_j \right] + O(h^3) \tag{2.13}
$$

Where $J_{i,j} = \dfrac{\delta f_i(Y_n)}{\delta Y_j}$ is the Jacobian matrix. The above scheme would yield a $O(h^2)$ globally with an iterative procedure to converge the solution.

Now if the nonlinearity is not converged iteratively, then the conventional scheme involves an explicit solution procedure which leads to conditional stability. Such an approximation involves the assumption that $J_{i,j} = 0$ for $i \neq j$. Then the numerical stability of the applied numerical scheme is restricted by the stability condition $h < \dfrac{2}{|\lambda_{max}|}$ where λ_{max} is the absolute maximum eigenvalue of the Jacobian matrix J and h is step size used by the scheme.

In a simultaneous update procedure, the stability conditions may be more stringent due to explicit procedures in all physics but in staggered coupling strategy, some of the physics use an implicit nonlinear procedure. Hence, the stability conditions will depend on the nonlinear operator of the corresponding physics with an explicit approximation.

II.3 Accuracy

Numerical errors arise from two sources: round-off and truncation errors. Round-off errors arise from the limited precision of computer arithmetic. It can be defined as the difference between the calculated approximation of a number and its exact mathematical

value. The relative error of a single operation is at most equal to the unit round-off error u. This is a small number, but if the computer carries out millions of operations, the round-off errors can accumulate and become significant. Numerical software writers strive to ensure that this accumulation is kept under control by making sure that the number of operations performed in calculating the solution is minimal. This in turn means that a scheme with unconditional stability and higher orders of accuracy allows the use of bigger time steps leading to lesser computation and lesser round-off errors.

The second source of error is called truncation error. This error arises when we make discrete approximations of continuous functions. This error can be, to a certain extent, limited by making the step-sizes in the discrete function as small as possible. The Taylor expansion series, which provides a means for creating approximate functions, also allows us to evaluate the truncation error. We often evaluate the quality of a numerical solution by estimating the error incurred with our functional approximations.

The accuracy of a scheme to solve a nonlinear system is really just a matter of minimizing the error term in the Taylor approximation for the scheme. Because of its simplicity and its applicability, it can be extended to the general nonlinear problem shown in Equation (2.1). Numerically solving a differential equation requires an initial condition and an algorithm for extending the solution. The idea is to expand the solution space around the initial condition and use the Taylor series to guide the approximation of the solution.

An important observation is that after each step of the numerical method a new initial value problem is approximated. This brings in the notion of a local error where after each step the incoming data is assumed to be exact. Therefore, accuracy is measured by comparing the numerical solution over one step with the corresponding Taylor series expansion of the exact solution as

$$y(t+h) = y(t) + \sum_{i=1}^{\infty} f^{(i-1)}(t)\frac{h^i}{i!} \tag{2.14}$$

The above equation is accurate without any kind of approximations. But, calculation using the formula requires approximations since it is not practical to get all the terms till infinity. Then, Equation (2.14) can be approximated as

$$y(t+h) = y(t) + \sum_{i=1}^{p} f^{(i-1)}(t)\frac{h^i}{i!} + \varepsilon^{p+1} \tag{2.15}$$

Where the local error is $\varepsilon^{p+1} = \sum_{i=p+1}^{\infty} f^{(i-1)}(t)\frac{h^i}{i!} = O(h^{p+1})$ (2.16)

Error analysis tells us that if the right hand side function is sufficiently smooth (p times continuously differentiable), and if the scheme is stable, then the local truncation error at a fixed time increases as $O(h^{p+1})$. When a solution is calculated for n steps, the local truncation error accumulates in the solution, which can be measured to give the true order of the discretization scheme. Then for a given IVP of the form given in Equation (2.1), the approximation in Equation (2.16) yields a p order accurate scheme globally.

$$Global\ error = \sum_{i=1}^{n} \varepsilon_i^{p+1} \approx n\varepsilon^{p+1} = \frac{T}{h} O(h^{p+1}) = O(h^p)$$

Where T is the total time of calculation.

If we assume that the solution is smooth enough then the truncation error converges to zero with decreasing step size. This property is known as *consistency* and is essential for all nonlinearly convergent schemes. The new global error consists of the action of the numerical scheme on the previous error and the error committed in the approximation of the derivatives according to the scheme. Also, the concept of "stiffly accurate schemes" is often used in the context of finding the solution fields to very stiff problems. This means that the scheme does not lose its theoretical order of accuracy when the fast modes in a stiff system dominate and hence is consistently accurate. Such schemes are optimal for our problem under consideration, which is both stiff and nonlinear in nature.

Summarizing the above, when a numerical method is used with a sufficiently large step size, the numerical solution may become unbounded, even though the exact solution is bounded. For certain methods applied to stiff problems, the step size necessary for stability may be excessively small in relation to the smoothness of the exact solution. This means stability rather than accuracy is restricting the step size. To ensure there is no restriction on the step size used, the numerical scheme needs to be stiffly accurate and both the A and L stability criteria needs to be met. Hence, for stiff systems, it is the

stability that is of prime importance and higher orders of accuracy are desired only when the former conditions are met.

It is important to understand the stability properties of the numerical schemes being used to choose the right step-sizes to obtain convergent solutions with desired accuracy for coupled physics systems. We will now look at some nonlinear iterative procedures that will be used to resolve the nonlinearities at each step when using implicit time discretization schemes.

II.4 Nonlinear iterative solution procedures

It is important to have consistent and stable schemes for solving nonlinear transient problems. For illustrative purposes, let us consider the nonlinear IVP given in Equation (2.1) where the nonlinear operator needs to be calculated approximately. In order to achieve this, let us expand the nonlinear operator using Taylor series expansion as

$$N(t_{n+1}, y_{n+1}) = N(t_{n+1}, y_n) + \frac{\partial N(y_n)}{\partial y}(y_{n+1} - y_n) + \ldots$$

$$\rightarrow N(t_{n+1}, y_{n+1}) \approx N(t_{n+1}, y_n) + \frac{\partial N(y_n)}{\partial y}\frac{\partial y}{\partial t}\Delta t + O(\Delta t^2)$$

It is evident from the above equation that the nonlinear resolution is vital to achieve the right convergence order for the time integration scheme used. If the nonlinearities are not resolved completely, then the local error in the nonlinear expansion $O(\Delta t^2)$ accumulates and destroys the convergence order gained using higher order time discretization schemes. Hence, it is important to iteratively converge the nonlinear problem in order to retain the accuracy in the solution.

There are several popular iterative schemes that have been used successfully to solve nonlinear problems. For our purposes, we shall consider only those schemes that have the advantages of ease of implementation and good convergence properties. It should be remembered that it is important to resolve the nonlinearities at each time step due to the undesirable effect that any inconsistent approximation used can reduce the order of accuracy of the overall time discretization scheme. The following discussion will detail about the Picard and Newton iterative procedure that are used to solve the nonlinearities to obtain the converged solution.

II.4.1 Picard iterations

Picard iteration also known as the fixed point iteration (FPI) method is a viable, easy method to implement but it is not very effective to iterate at every time step to converge the nonlinearities in order restore the convergence order. Such a solution procedure will have a high CPU cost and therefore, may take longer in computation time. Hence, few modifications might be needed on top of the θ time discretization scheme and Picard iterations to make it a potential candidate among other numerical scheme for reactor analysis.

In solving differential equations, Picard iteration is a constructive procedure for establishing the existence of a solution to a differential equation $y' = f(t,y)$ that passes through the fixed point (t_o, y_o). The method of successive approximations used in Picard iterations uses the equivalent integral equation for Equation (2.1) and an iterative method for constructing approximations to the solution.

The solution to the IVP in Equation (2.1) is found by constructing recursively a sequence of functions as follows.

$$y(t) = y_o, \ and \ \ y_{n+1}^{l+1}(t) = y_o + \int_{t_n}^{t_{n+1}} f(s, y_n^l(s))ds \tag{2.17}$$

Then the solution to Equation (2.1) is given by the limit

$$y_{n+1}^{CV}(t) = \lim_{l \to \infty} y_{n+1}^{l}(t) \tag{2.18}$$

This is one of the simplest numerical techniques to implement and that explains the reason why fixed point iterations can be used in the current nonlinearly inconsistent coupling schemes. Also, acceleration schemes could be used to improve the convergence rate of the nonlinear iteration. A brief discussion on this is provided below.

II.4.2 Newton methods

The family of Newton-like methods is another appropriate choice for solving nonlinear problems. Unlike other methods, the Newton methods require the computation of the Jacobian matrix for the nonlinear system, which can be very expensive if calculated numerically. Unless the problem under consideration is simple enough, analytical Jacobians cannot be explicitly found and hence pose a severe problem. To circumvent this overhead, a modified Newton iteration called the Inexact Newton can be

utilized where the calculation of the Jacobi can be done only once in several *m* steps, if the variation of the Jacobian with respect to solution field is weak.

One point iterations are those schemes where the solution y_{i+1} is related to y_i in some way but independent of *i* itself. An important example of a one-point iteration is Newton's iterative method. For an IVP of the form Equation (2.l), a suitable time discretization can be applied first and then the resulting equation can be solved to get the solution at the end of the time step by Newton's method.

Newton's method is a generalized process to find an accurate root of a system of equations $f(x)=0$. Suppose that *f* is a C^2 function on a given interval, then using Taylor's expansion near *x,*

$$f(y+\delta y) = f(y) + \delta y f'(y) + O(\delta y^2) \tag{2.19}$$

Now if we stop at first order then we obtain,

$$f(y+\delta y) = 0 \approx f(y) + \delta y f'(y) \rightarrow \delta y = -\frac{f(y)}{f'(y)} \tag{2.20}$$

or more generally

$$y_{l+1} = y_l - J^{-1} f \tag{2.21}$$

where *J* is the Jacobian of *f*.

Equation (2.21) is the recursive form of Newton's iterative formula which can be to find the complete solution of the IVP. Also, it should be noted that the Newton's iterative scheme is *quadratically convergent*.

The Picard iteration procedure offers the advantage of being easy to implement and has a cheap computational cost per iteration but it has been known to fail or converge slowly. The Newton's method on the other hand, is more expensive computationally but its higher convergence rate makes it an attractive alternative for strong nonlinear problems.

Now, we will review the different time discretization schemes and methods that can be used to obtain the solution fields for the coupled, multi-physics transient problem.

II.5 Time discretization strategies

There are two types of discretization strategies that were tried out in the current research. They are given as:

1) Constant step – In this strategy, the time step sizes are maintained constant and do not change dynamically as the transient calculation proceeds.
2) Adaptive step – In this strategy, the time step is predicted and controlled to yield solution within user specified tolerance. This methodology maximizes the performance of the time integration scheme by using larger time steps when allowed and reduce it when sudden discontinuities occur in the system.

The details on these strategies are given below.

II.5.1 Constant time-stepping strategy

As briefly described before, the constant step strategy is straightforward and makes use of a constant time step to solve problems of type Equation (2.1). The advantage of using such a simple procedure is that it allows us to understand the intricacies involved in implementing a numerical scheme for a given problem since the effort needed is lesser compared to an adaptive stepping strategy.

In the current research, only the θ discretization scheme has been implemented using both the Picard and Newton iterative solvers for converging the nonlinearities in each physics. We will also propose modifications to the classic C.N scheme which will improve the order of convergence of the scheme without the expensive iterative procedure at every time step.

The order of a scheme in this strategy can be measured by computing a reference solution first using a very fine step size and then computing solutions with larger step sizes. Then the global error in the solution can be expressed as

$$E(h) = \| y_{ref}^{T} - y_h(T) \| = Ch^p \tag{2.22}$$

Equation (2.22) can then be used to obtain the order of the scheme p and matched with the theoretical accuracy to find if the scheme is nonlinearly consistent for the problem at hand.

II.5.2 Adaptive time-stepping strategy

Adaptive time discretization is essential when constant time step methods cannot capture the irregular variations in the solution. For most stiff problems, as benchmarked by several researchers like Butcher and Hairer, an adaptive method would provide significant improvement on the number of steps for a given accuracy of the solution.

The adaptive algorithm basically consists of primarily two steps.

1) Error estimation procedure
2) A time step selection strategy based on the error estimate

A brief overview of each of the steps is given below.

II.5.2.1 Error determination procedure

Typically, two different error determination strategies based on the error estimation principle are commonly used. The details of these strategies are given below.

II.5.2.1.1 Time step doubling method

This is a straightforward method and is usually effective when the number of function evaluations needed for a numerical scheme is low. Another advantage of this method is that the implementation requires the use of only one numerical method of a fixed order rather than 2 schemes of different orders as in the case of embedded scheme.

The idea is that a solution is calculated using a step size $2h$ and then again using 2 half steps of size h. The difference between the calculated solution values gives a measure of the local truncation error in the solution. The equations representing the error for a method of order p can be derived easily as follows.

$$Step\ size\ h: y_{n+1}^{h} = y_n^h + Ch^p + O(h^{p+1})$$

$$Step\ size\ h: y_{n+2}^{h} = y_{n+1}^h + Ch^p + O(h^{p+1}) = y_n^h + 2Ch^p + O(h^{p+1})$$

$$Step\ size\ 2h: y_{n+2}^{2h} = y_n^h + C(2h)^p + O((2h)^{p+1})$$

Then, the local error estimate can be given as

$$Local\ Error\ \varepsilon = \frac{y_{n+2}^{2h} - y_{n+2}^{h}}{(2^p - 1)} = Ch^p + O(h^{p+1}) \tag{2.23}$$

II.5.2.1.2 Embedded methods

This method has been used for a variety of problems in adaptive time stepping strategy and relies on using two schemes with different orders of accuracy to find the local truncation error. For example let us consider two schemes with orders p and $p+1$ and let us denote the exact solution as $y(t_{n+1})$.

$$Lower\ Order: y(t_{n+1}) = y_{n+1}^{1} + C_1 h^p + O(h^{p+1})$$

$$Higher\ Order: y(t_{n+1}) = y_{n+1}^2 + C_2 h^{p+1} + O(h^{p+2})$$

Then the local error can be measured as

$$Local\ Error\ \varepsilon = y_{n+1}^2 - y_{n+1}^1 = O(h^{p+1}) \tag{2.24}$$

Once the local truncation error from either A or B method is measured, the next procedure would be to modify the time-step accordingly.

II.5.2.2 Time-step control strategy

Several step control procedures have been devised for nonlinear dynamical systems by researchers K. Gustafsson[11, 12], H. Watts[13] and G. Soderlind[14]. In the current research, the ideas proposed by Gustafsson have been taken as the basis to implement the step control procedures. Now some of the different options available in this procedure are shown and discussed below.

II.5.2.2.1 Standard step size controller

$$h_{n+1} = Sh_n \left(\frac{\tau}{|\varepsilon|} \right)^{\frac{1}{p+1}} \tag{2.25}$$

Where p is global order of the method,

ε is the estimated local truncation error,

τ is the user specified tolerance,

S is a safety factor τ .

II.5.2.2.2 PI step size controller

The above standard controller does not resolve the instabilities. To overcome this, Gustafsson and Soderlind devised a new step size controller based on the Proportional-Integral (PI) control. The controller is given as

$$h_{n+1} = Sh_n \left(\frac{\tau}{|\varepsilon_n|} \right)^{k_I} \left(\frac{|\varepsilon_{n-1}|}{|\varepsilon_n|} \right)^{k_P} \tag{2.26}$$

where the method specific parameters with typical values given by $k_I = \frac{0.3}{p_e}$ and $k_I = \frac{0.4}{p_e}$ and p_e is the local order of the method (i.e., p_e=p+*1*).

II.5.2.3 Measuring order of accuracy

Unlike the constant stepping strategy, the measurement of the order of accuracy for the adaptive stepping strategy is not as simple since the step sizes are all different. But since the step sizes are controlled based on the local truncation error, it provides us a way to relate the order of the method with the total CPU time needed to obtain the solution at time *T*. Assuming that each step requires about the same amount of calculation effort to compute the solution, we can then relate from the above standard step control mechanism that

$$Steps \propto \left(\frac{1}{\tau}\right)^{\frac{1}{p_e}} \tag{2.27}$$

Hence, a plot of -log(τ) (Y axis) and log(Steps) (X axis) will give a slope of p_e. This procedure will be used to obtain the convergence order results for both the time-doubled and the embedded schemes in this research.

II.5.3 Implications of the choice of integration method

When a numerical solution to the general nonlinear problem is found, the step size will be chosen based on the modes that currently dominate the error estimates. Fast modes that have converged during the main transient period will move toward the asymptotic region and the contribution of the fast mode during longer periods will be negligible. Should such a mode be excited and become active again, the step size has to be decreased substantially in order to resolve the new transient.

It is important that the integration method does not erroneously excite the modes that have converged. This makes the behavior of $R(z)$ as $z \to -\infty$ of great importance. Although $|R(z)| < 1$ is enough for stability, this requirement is not entirely sufficient for good performance. The value of $R(z)$ governs how much of a specific mode is propagated from step to step, and $R(z) \to 0$ as $z \to -\infty$ (a property shared by all L-stable methods). If this is not the case, errors in the fast modes will propagate from step to step, and if $|R(z)|$ gets close to 1, the errors may accumulate and eventually become large enough to cause a step rejection. It then takes a drastic step size decrease in order to continue the integration. The fast mode must be returned to the asymptotic region, i.e., where the *local* error is small enough, and control authority is regained. This behavior has been observed

in practical codes and has been referred to as the "Hump phenomenon" in Hairer and Wanner (1991, p. 113).

Even when using stiffly accurate integration methods there may be error contributions from modes in the non-asymptotic region. This may be experienced as the true value of the order is different from the one expected. If the experienced variation is moderate, it may be modeled as changes in p although such a scheme has not been implemented in this research and is outside the scope of the document.

Next, we will review some basic theory behind time discretization schemes used in combination with the adaptive time step control strategy to obtain the solution field with user desired accuracy. All methods chosen are nonlinearly consistent schemes and hence should perform well for stiff systems.

II.6 Time discretization schemes

There are not many families of numerical schemes available for solving stiff problems efficiently and consistently. Some of the popular methods that can be used are the θ-discretization and Runge-Kutta family of methods. It is important to note that the conventional operator-split codes use θ-discretization to obtain the transient solution along with the constant time stepping strategy. Hence, we will follow the same path and try to improve the convergence of the conventional scheme by introducing better approximations in the scheme. For the adaptive time stepping strategy, the higher order variants of the RK scheme with good stability properties will be used. We shall now discuss each of the different methods in detail.

II.6.1 Theta discretization scheme

The theta method is a variable parameter (θ) time integration scheme, which permits the resulting difference equations to range from fully explicit to fully implicit. For a given value of the variable parameter θ, the solution of the time-dependent equations of the form given in Equation (2.1) reduces to a sequence of local problems in which the fixed term is composed of quantities computed from the solution of the previous time point. In each local problem, the unknown solution field is computed using a

conventional nonlinear iteration procedure with possibly an acceleration scheme applied on top of it.

Now discretizing Equation (2.1) with a θ–scheme gives

$$\frac{y_{n+1} - y_n}{\Delta t} = \theta f(y_{n+1}) + (1-\theta) f(y_n) \tag{2.28}$$

where a backward difference approximation $\frac{dy}{dt} = \frac{y_{n+1} - y_n}{\Delta t}$ has been applied and $\Delta t = t^{n+1} - t^n$.

Equation (2.28) can be expressed in an alternate form, where the solution field at $t=t^{n+1}$ is expressed in terms of the solution field at $t=t^n$.

$$(I - \theta\ \Delta t G(y_{n+1}^l))\, y_{n+1}^{l+1} = (I + (1-\theta)\ \Delta t F(y_n))\, y_n \tag{2.29}$$

Where $F(y_{n+1})$ is expressed as $G(y_{n+1}^l)\, y_{n+1}^{l+1}$ where l is the nonlinear iteration index

Equation (2.29) can also be expressed more concisely in the form, $y_{n+1} = A^{-1} B y_n$.

Note that when $\theta=0$, the θ–scheme gives the first order explicit Euler or the Forward Euler scheme. Since this scheme is explicit, we will not use this for the stiff coupled problem due to restrictive stability reasons.

When $\theta=1$, the θ–scheme represents the first order implicit Euler or Backward Euler scheme. From Equation (2.29), it can be seen that the implicit Euler then is of the form $y^{n+1} = A^{-1} I y^n$. Since this scheme is both A and L stable, it is stiffly accurate and can yield an exact estimation of the solution. The only drawback is that this scheme is only first order globally $O(h)$ and hence requires smaller steps to reach a desired accuracy.

Now when $\theta=1/2$, the θ–scheme represents the implicit second order Crank-Nicolson scheme. It is important to note that the CN scheme is $O(h^2)$ which makes it a very attractive candidate for the purpose of creating higher order accurate, stable numerical schemes for coupled systems. Even though the CN scheme is A-stable, it is not L-stable and hence when the step sizes become larger, we can observe oscillations in the solutions that will be damped out very slowly. This region should be monitored and carefully avoided in order to obtain meaningful solution fields.

In general, the order of accuracy of the θ discretization scheme can be expressed as $O(h^2)$ when $\theta=2$ and $O(h)$ otherwise. This is one of the simplest time discretization

schemes that are popular due to the fact that there is only one parameter involved in the definition of the scheme which controls the stability of the resulting scheme and the order of accuracy. Thus for $\theta \geq 1/2$, we obtain stable implicit schemes which are easy to implement, with a few modification if necessary to find the solution.

II.6.2 Runge-Kutta family of methods

Runge(1895) and Kutta(1901) formulated the general scheme of methods which is now called as the Runge-Kutta method. The general definition of a RK scheme can be represented as follows.

Definition

A Runge-Kutta (RK) method with s stages is defined by

$$k_i = f\left(t_{n-1} + hc_j, y_{n-1} + h\sum_{j=1}^{s} a_{i,j}k_j\right), \; i = 1,2 \ldots s$$

$$y_n = y_{n\text{-}1} + h\sum_{j=1}^{s} b_j k_j \tag{2.30}$$

In this general formula, k_i represents the internal stage values and y_n is the update at the n^{th} step which is the numerical approximation to $y(t)$ at $t=t_n$. h denotes the step-size and the coefficients $a_{i,j}$, b_j, c_j are constants which can be constructed to yield the desired approximation to the solution.

Now if $a_{i,j} = 0$ for $j \geq i$, then the Runge-Kutta method is fully *explicit* (*ERK*). And if $a_{i,j} = 0$ for $j > i$, then the method is called *Diagonally Implicit RK* (*DIRK*). Else the method is fully *implicit* (*IRK*).

A simpler representation of the s stage RK method with the coefficients listed in the tableau was first introduced by Butcher.

Butcher Tableau – General formulation

c_1	a_{11}	a_{12}	...	a_{1s}
c_2	a_{21}	a_{22}	...	a_{2s}
.	.	.	...	.
.	.	.	...	.
c_s	a_{s1}	a_{s2}	...	a_{ss}
	b_1	b_2	...	b_s

A RK method of order p can be compared to the actual Taylor series expansion, to derive the order conditions. For higher order methods, it gives the user great flexibility in deriving a scheme with optimal order and stability properties to fit the needs of the problem. This plasticity of the method and the ease of adjusting the coefficients to obtain embedded formulas make them attractive to the adaptive time-stepping problem at hand. Also, the implication of all this is that the coefficients can be adjusted for the embedded formulas to satisfy the following goals:

1) Assured accuracy
2) Minimal CPU time by using lesser stages and via adaptively using larger steps when necessary

We shall discuss several higher order RK schemes and explain briefly on the accuracy and stability of the embedded schemes.

II.6.2.1 Explicit RK schemes (ERK)

Explicit methods are easy to implement and have cheap computational cost because the internal stages depend only on previous stages and hence require no nonlinear iterative procedure or matrix inversions. But since they have poor stability properties and are unable to resolve very fast changing modes (explicit schemes are not suitable for stiff equations). To solve this problem and to utilize the advantages of these one step schemes, modifications to the existing ERK schemes as shown by Eriksson et al., [4] can be made to extend the stability region to resolve the modes smoothly.

One common procedure usually used to accomplish this is sub-stepping where a given step found adaptively can be divided into several smaller steps. This is in a way the same as using a smaller time-step although we do not explicitly store the solution fields at the intermediate stages. Another procedure that can be used to extend the stability region of a scheme for stiff problems is to introduce an additional damping term to resolve the fast modes (dominant eigenvalue).

Based on these conclusions it seems feasible to efficiently use an explicit method, if it is adaptively stabilized with a relatively low number of small time steps, so that we gain the desired combination of a low cost per time step and with the possibility of using large time steps beyond rapidly changing transients.

In order to test the suitability of the explicit methods for the nonlinear, stiff nuclear reactor analysis problems, two classical Explicit RK (ERK) schemes ERK3, ERK4 of global order 3 and 4 respectively were chosen. These methods could be used with both the step-doubled and embedded strategies to adaptively control the time-steps used. More analysis on the usability of these schemes will be discussed in Chapter V.

The RK coefficients for the explicit schemes are shown below.

Explicit RK – Order 3

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array} \qquad (2.31)$$

Explicit RK – Order 4

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \qquad (2.32)$$

These schemes are very simple in nature and are inferior in terms of stability to the more established RK-Fehlberg (RKF) and Dormand-Prince (DOPRI) higher order schemes. Also, since the above methods are not truly embedded, the cost of evaluation in an adaptive time stepping scheme will be slightly higher than DOPRI. But, in the current research, the idea is to find out a strategy to adapt the above schemes to stably find accurate solutions which directly implies the usability of the other specialized RK schemes.

II.6.2.2 Rosenbrock-Wanner (ROW) methods

The main idea behind the family of Rosenbrock-Wanner methods is to linearize Equation (2.30) to obtain a new set of equations of the form

$$k_i = hf(t_{n-1} + hc_i, \sum_{j=1}^{i-1} \alpha_{i,j} k_j) + hJ \sum_{j=1}^{i} \gamma_{i,j} k_j, \; i = 1,2 \ldots s$$

$$\rightarrow (I - hJ\gamma_{i,i}) k_i = hf(t_{n-1} + hc_i, \sum_{j=1}^{i-1} \alpha_{i,j} k_j) + hJ \sum_{j=1}^{i-1} \gamma_{i,j} k_j \qquad (2.33)$$

$$y_n = y_{n-1} + \sum_{j=1}^{s} b_j k_j$$

Where $\alpha_{i,j}$, $\beta_{i,j}$, b_j are the method specific coefficients and $J = \left.\frac{\partial f}{\partial y}\right|_{y_{n-1}}$ is the Jacobian of f evaluated at the moment $n-1$.

This can be interpreted as the application of one Newton iteration to each stage in Equation (2.30) with starting values $k^{(0)}_i = 0$. Instead of continuing the iterations until convergence, we consider Equation (2.33) as a new class of methods whose stability properties have been analyzed and documented by Hairer and Wanner.

The advantage of this method is obvious since the unknown k_i can be found in one step by inverting the matrix $I - h\gamma_{i,i}J$. Now if we find coefficients such that $\gamma_{i,i} = \gamma$, we can obtain a special class of methods called the Singly Diagonally Implicit RK scheme (SDIRK) which requires for instance only one LU-decomposition of the matrix per time step.

For solving nuclear reactor transient problems, we have specifically chosen the Generalized *A*-stable RK methods of order four introduced by Kaps and Rentrop[15]. Now based on the order conditions, two different γ were chosen to obtain different stability properties while minimizing the local truncation errors. These generalized RK schemes are called the GRK4A and GRK4T schemes.

The GRK4A scheme is *A*-stable but not *L*-stable but the GRK4T scheme is both *A* and *L* stable. Hence these schemes are expected to perform well for stiff problems if a suitable adaptive time stepping control procedure is chosen. The different coefficients for the above schemes which can be used to obtain the solution with a global accuracy of $O(h^4)$ are given below.

The coefficients for the schemes given below can be used to calculate the 3rd order and 4th order accurate solution which can be used to estimate the local error. Since the scheme is embedded, the higher order solution evaluation does not involve any extra

function evaluation. Once Equation (2.33) is used to find the stage function values, the solution can be calculated as

$$y_n = y_{n\text{-}1} + \sum_{j=1}^{4} b_j k_j$$

And the local error using

$$e_n = \sum_{j=1}^{4} (b_j - b_j^{'}) k_j + O(h^5)$$

The coefficients for the GRK4A scheme are listed below.

$\gamma=0.395$, $\gamma_{21}=-0.767672395484$,

$\gamma_{31}=-0.851675323742$, $\gamma_{32}=0.522967289188$

$\gamma_{41}=0.288463109545$, $\gamma_{42}=0.0880214273381$, $\gamma_{43}=-0.337389840627$

$\alpha_{21}=-0.438$,

$\alpha_{31}=0.796920457938$, $\alpha_{32}=0.0730795420615$,

$b_1=0.199293275701$, $b_2=0.482645235474$, (2.34)

$b_3=0.0680614886256$, $b_4=0.25$,

$b'_1=0.346325833758$, $b'_2=0.285693175712$, $b'_3=0.367980990530$

These coefficients yield a local truncation error $\varepsilon \leq 1.08/4!$

The coefficients for the GRK4T scheme are listed below.

$\gamma=0.231$, $\gamma21=-0.270629667752$,

$\gamma31=-0.3112844/3294$, $\gamma32=0.00852445628482$,

$\gamma41=0.282816832044$, $\gamma42=-0.457959483281$, $\gamma_{43}=-0.111208333333$,

$\alpha_{21}=0.462$,

$\alpha_{31}=-0.0815668168327$, $\alpha_{32}=0.961775150166$, (2.35)

$b_1=0.217487371653$, $b_2=0.48622903799$,

$b_3=0.0$, $b_4=0.296283590357$,

$b'_1=-0.717088504499$, $b'_2=1.77617912176$, $b'_3=-0.0590906172617$

These coefficients yield a local truncation error $\varepsilon \leq 0.461/4!$

An adaptive control strategy based on the local truncation error found out using these embedded schemes can then be used to solve the stiff coupled system. Although these schemes are stable, a curious phenomenon called the “Hump” occurs while calculating

the solution fields where the step sizes drop by a factor of 10^{-3} which further leads to huge number of step rejection. This phenomenon observed by Hairer can be overcome by imposing drastic step reductions when consecutive step rejections are encountered. The results obtained by this procedure have been discussed in Chapter V.

II.6.2.3 Singly Diagonally Implicit RK (SDIRK)

The SDIRK methods are a special class of DIRK methods where the order conditions and coefficients are calculated so as to obtain $\gamma_{i,i}=\gamma$, the free parameter. This leads to a simplified form of the DIRK method which requires the inversion of a single matrix $(I - h\gamma J)$ per time step.

In search of a higher order embedded IRK scheme, we have chosen the SDIRK(3,4) scheme proposed by Hairer (p.100) which is both *A* and *L* stable with an additional property of being stiffly stable. The coefficients for this scheme are shown below in the form of a Butcher tableau.

This implicit scheme is of high order accuracy and is also a stiffly accurate. Hence, they provide another attractive alternate to finding the solutions for stiff, nonlinear systems. The local error for the scheme is then given as $\varepsilon_n = \sum_{j=1}^{5}(b_j - b_j^{'})k_j + O(h^5)$.

$$
\begin{array}{c|ccccc}
\frac{1}{4} & \frac{1}{4} & & & & \\
\frac{3}{4} & \frac{1}{2} & \frac{1}{4} & & & \\
\frac{11}{20} & \frac{17}{50} & \frac{-1}{25} & \frac{1}{4} & & \\
\frac{1}{2} & \frac{371}{1360} & \frac{-137}{2720} & \frac{15}{544} & \frac{1}{4} & \\
1 & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \\
\hline
b & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \\
b' & \frac{59}{48} & \frac{-17}{96} & \frac{225}{32} & \frac{-85}{12} & 0 \\
\hline
e & \frac{-3}{16} & \frac{-27}{32} & \frac{25}{32} & 0 & \frac{1}{4}
\end{array}
\qquad (2.36)
$$

The actual nonlinear system to solve per time step can simply be written as

$$k_i = f(t_{n-1} + hc_i, y_n + h\sum_{j=1}^{5} a_{i,j}k_j)\ ,\ i = 1\ldots5$$

Using the notation $k := (k_1; k_2; : : : ; k_s)^T$, we can formulate the nonlinear system of equations as an s-dimensional fixed-point equations

$$k = F(k), F_i(k) \ = f(t_{n-1} + hc_i, y_n + h\sum_{j=1}^{s} a_{i,j}k_j) \qquad (2.37)$$

It seems reasonable to assume that Picard iterations can be used to converge the nonlinearities and to obtain the accurate stage solution fields. But it has been known that sometimes fixed point iterations destroy the stability properties of the IRK schemes. Instead, Newton's method is usually employed to resolve the nonlinearities in the SDIRK schemes. Using the simplified Newton's procedure described by Hairer (Stiff ODE : Chapter IV.8), the solution reliability and accuracy can be improved drastically. The cost of computation in such a procedure/iteration/step is usually *s* function evaluations. Also, suggestions to improve the step estimation procedure by replacing the traditional local error term ε_n by a more stable estimate in the form of $(I - h\gamma J)^{-1}\varepsilon_n$ where the term $(I - h\gamma J)^{-1}$ acts as a filter to damp out the stiff parts. This additional error estimation does not cost much except for a linear solve since the LU decomposition of the matrix has already been calculated for the step.

Based on the stability properties and the high global accuracy orders associated with this method, it is predicted that this scheme will perform well for the nonlinear, stiff nuclear transient problem.

II.6.2.4 RADAU IIA

Implicit RK schemes are stable schemes which contain a full coefficient matrix typically requiring more computational effort to solve the nonlinearities at each time step. Since the cost associated with such a procedure is very high when function evaluations are complex, the need to use efficient Newton procedures and step control strategies to resolve the problem is important.

Butcher (1964) introduced the RK schemes based on RADAU quadrature formulas. He called them processes of type I, II or III based on certain order conditions being

satisfied by the coefficients. Several other researchers including Axelsson (1969) worked on producing schemes with strong stability properties.

The RADAU scheme of type IIA is both *A*, *L*-stable. It has been theoretically proved that the *s*-stage RADAU-IIA scheme is of order 2*s*-1 with the stability function given by the (*s*-1, *s*) Pade' approximation.

We are interested in the embedded scheme of RADAUIIA or simply RADAU5 which contains 3 stages of computation per time step and is globally $O(h^5)$ accurate. The actual coefficients of this scheme are provided below in a Butcher tableau which is based on the code that was successfully implemented and tested by Hairer for several stiff problems.

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array} \qquad (2.38)$$

Most of the questions regarding the correct implementation of the simplified Newton solver, the starting values and the stopping criteria for the iterations and the selection of step sizes have been discussed by Hairer (Stiff ODE - Chapter IV.8). Apart from the implementation details, he also points out some interesting conclusions on the RADAU5 adaptive scheme. Some of those that are relevant to this research are mentioned below.

RADAU5 is a good stiff integration and had very good stability properties. Since it is *L*-stable, the stability domain also covers the imaginary axis. This means that high oscillations in the solution may be damped by the numerical method. It is also interesting to note that the local error of RADAU5 schemes behaves as $O(h^6)$ globally for $h \leq \epsilon$ and for large *h* which means that for coarse tolerances, the scheme converges faster than the theoretical order of convergence. Also, the improvement in the estimation of the error based on the idea shown in the SDIRK scheme can lead to significantly precise solution fields which are always accurate to the user specified tolerance.

All of the properties mentioned for the GRK, SDIRK and RADAU5 schemes do seem attractive from the perspective of higher orders of accuracy and better stability properties. Also based on previous usage of these schemes to solve nonlinear stiff problems like the Van der Pol equation, the performance in comparison to any of the conventional schemes should be multi-fold. A detailed analysis of the gain will be discussed when the solution from the simulation to the coupled problem is shown in results section.

II.7 Overcoming drawbacks in current schemes

The current *nonlinearly inconsistent coupling schemes* have several disadvantages that have been discussed before. The aim of this research is to devise methods that are robust, stable and accurate along with the flexibility to reuse existing mono-physics legacy codes as much as possible to create *a nonlinearly consistent (NC) coupling strategy*.

There are several strategies that can be used to obtain a consistent coupling. Some of those strategies are described below.

II.7.1 Solution prediction

A solution prediction is an idea derived from extrapolation. Instead of solving for an unknown quantity by iterative methods, it obviously makes sense to extrapolate the solution based on the history of solution and use the particular value as the starting point. In our current research, we will introduce 2 predictions; one with a 2^{nd} order accuracy using Taylor series approximation and another with a 3^{rd} order accuracy using step-doubling strategy to improve the convergence order of the conventional schemes. These procedures however bear an assumption that the transient solution is smooth and the first, second derivatives are constant over a reasonably small time step. Now let us discuss each of these methods in detail.

II.7.1.1 Second order prediction

As mentioned before, the notion behind the solution prediction procedure is to approximate the solution value at $t+h$ based on the history of the solution. Hence for a nonlinear problem of the form given in Equation (2.1), the Taylor series expansion for the nonlinear term N yields

$$y(t+h) = y(t) + hy'(t) + O(h^2)$$

This equation can then be simplified by applying BDF, to get

$$y(t+h) = y(t) + (y(t) - y(t-h)) + O(h^2)$$

$$y^p(t+h) = 2y(t) - y(t-h) + O(h^2) \tag{2.39}$$

We can see that with the prediction of the solution by extrapolation and by using it as the starting value, the order that is lost by not converging the nonlinearities in the conventional scheme by iteration can be restored. Hence, we can have a modified fixed point scheme which resembles PEC (Predict-Evaluate-Correct) schemes that require only one iteration with the nonlinear function resolved using the predicted value.

The primary advantage of such a modification is that, for just 1 corrector iteration after a prediction, we gain an extra order without the drawback of having to resolve the nonlinearities iteratively. This proposed scheme in the current research for constant step strategy to find the solutions for multi-physics problems has been tested for transient reactor accidents.

It should be obvious by now that this idea is an extension from multistep methods and more terms can be used to get prediction values of higher order and can be corrected to get overall higher orders of accuracy. But before considering the gain in the accuracy, it is also imperative to find the effect of such an approximation on the stability of the scheme being used.

Due to the explicit linearization of the nonlinear term, this procedure will yield only conditional stability ($h < \frac{2}{|\lambda_{max}|}$) which can severely restrict the usage of larger time steps for stiff problems.

II.7.1.2 Third order prediction

There is also a variant to the prediction scheme that can be used to obtain higher order of global accuracy $O(h^3)$ than the previously described local explicit extrapolation. This idea is based on the Time-doubling strategy previously discussed for the adaptive time stepping scheme. Here the solution at a particular time-step is calculated with 2 different step sizes : h and $2h$ with the 2nd order solution prediction applied at every step. The local error is then calculated and the solution is corrected accordingly[16] to yield the desired global *cubic convergence*.

For illustrative purposes, let us consider the second order accurate CN scheme over which the 3rd order prediction procedure is performed. Now, since this procedure is exactly the same as the step-doubling strategy in adaptive time stepping, we could use the estimate of the local error derived and shown in Equation (2.23). Then, based on that, the local error is given as

$$\Delta = \frac{y_{n+2}^{2h} - y_{n+2}^{h}}{6} = Ch^3 + O(h^4) \tag{2.40}$$

Where the value C remains approximately constant over the time step and the magnitude of which calculated using the actual Taylor series expansion gives the magnitude as $y^{(3)}(x)/3!$. Now expanding Equation (2.38) using the value for C, we then get

$$y_{n+2}^{h,c} = y_{n+2}^{h} + h^3 \frac{f''(x)}{6} + O(h^4) = y_{n+2}^{h} + \Delta + O(h^4) \tag{2.41}$$

It can be seen that the corrected solution is locally $O(h^4)$ and hence converges globally as $O(h^3)$. This procedure although simple does increase the CPU cost by 50% since we do achieve the accuracy of the smaller time step even without the $2h$ step size calculation. But again, similar to the second order prediction method, such a procedure only leads to conditional stability in the scheme.

II.7.2.1 Picard iterations

Fixed point or Picard iterations as mentioned before are nonlinear iterative schemes which can be used to converge the nonlinearities over the different physics when an operator split technique is used to couple multi-physics. Picard iterations can restore the convergence order of a higher order scheme and eliminates the loss of accuracy due to the crude explicit approximation $N(y^{n+1})=N(y^n)$;

There are several disadvantages of using such a strategy to restore the accuracy which include the increase in computational and memory usage in the existing codes. But it is essential to stress that the stability of the higher order discretization scheme can be maintained using this procedure unlike the explicit linearization method where the solution is only conditionally stable.

II.7.2.2 Accelerated Picard iterations

There are several acceleration techniques available to improve the speed of convergence of an iterative procedure. It should be remembered that the order of convergence does not change when acceleration techniques are used for a particular iterative scheme but the number of iterations to reach final convergence can be reduced significantly.

For the numerical schemes and methods discussed in the preceding sections, a technique called the Aitken's Δ^2 process satisfies all the criteria. The Aitken's method can be used to speed up convergence for any sequence that is linearly convergent. In order to proceed, we define the Aitken's method.

Given the sequence $\{p_n\}_{n=0}^{\infty}$, we define the forward difference formula as $\Delta p_n = p_{n+1} - p_n$ for $n=1,2,3\ldots$ Higher powers $\Delta^k p_n$ are defined recursively by $\Delta^k p_n = \Delta^{k-1}(\Delta p_n)$ for $k \geq 2$. When $k=2$ we have the useful formula $\Delta^2 p_n = \Delta(\Delta p_n) = (p_{n+2} - p_{n+1}) - (p_{n+1} - p_n)$ which simplifies to $\Delta^2 p_n = p_{n+2} - 2p_{n+1} + p_n \quad for \quad n = 2,\ 3,\ \ldots$

Now assume that the sequence $\{p_n\}_{n=0}^{\infty}$ converges linearly to the limit p and that $p_n \neq p$ for all $n \geq 0$. If there exists a real number A with $|A| < 1$ such that

$$\lim_{n\to\infty} \frac{p - p_{n+1}}{p - p_n} = A \tag{2.42}$$

then the sequence $\{q_n\}_{n=0}^{\infty}$ defined by

$$q_n = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n} = p_n - \frac{(p_{n+1} - p_n)^2}{(p_{n+2} - 2p_{n+1} + p_n)} \tag{2.43}$$

converges to p faster than $\{p_n\}_{n=0}^{\infty}$, in the sense that

$$\lim_{n\to\infty} \left| \frac{p - q_n}{p - p_n} \right| = 0$$

II.7.2.2.1 Steffensen's acceleration

When Aitken's process is combined with the fixed point iteration, the result is called Steffensen's acceleration[17]. Starting with p_0, two steps of fixed point method are used to compute p_1 and p_2. Then Aitken's Δ^2 process is used to compute the accelerated value, q_o.

$$q_0 = p_0 - \frac{(\Delta p_0)^2}{\Delta^2 p_0} \tag{2.44}$$

Once q_o is calculated, $p_o = q_o$ and the whole process is repeated again. This method is effective and is specific to Picard iterations.

II.7.3 Implicit solution of the monolithic block

Current operator-split strategies offer flexibility in the way the different physics are solved but involve complexities in terms of resolving the nonlinearities and finding a consistent solution. Instead, the coupled problem can be tackled as a whole using higher order implicit RK schemes which require the calculation of the Jacobian matrix.

Jacobian matrix calculations can be expensive for higher dimensional problems and alternatives using Jacobian-free methods can be used in such cases. But in a simpler nonlinear problem, like in the case of a 0-D model in reactor analysis, the solution fields can be found out by solving the system of equations implicitly, eliminating the loss of convergence orders in the coupling strategy thereby creating a NCC strategy.

II.7.4 Summary

Summarizing the above discussion, current schemes perform a crude approximation for the nonlinear term and hence the solution is only first order accurate in time. Now, by including explicit solution prediction in the conventional staggered schemes, the lost order can be gained and we can create more accurate coupling at no extra cost. The only change is to modify the interface between the coupling blocks and perform the appropriate extrapolation to get the predicted values.

Apart from this strategy, we can also aim to use Picard iterations with and without acceleration where the nonlinearities are completely converged at each time step. The Aitken Δ^2 process is very efficient and requires storing only 2 preceding solution vectors, which is trivial, given the configuration of current computers. In the results, we will discuss the performance gain of such acceleration and the amount of CPU time saved with such a procedure.

The advantages of using an adaptive time-stepping strategy provides better estimation of the solution accuracy and results in resolving the transient modes efficiently i.e., using small time steps during rapid transients and larger time step during slow transient period.

Also, the usage of higher order RK schemes detailed above provide several benefits over the classical θ-discretization scheme with a NCC strategy with respect to stability and stiff integration properties. Such specialized methods with a good step control method (PI) could lead to stable, consistent and highly accurate coupling solution procedures.

Given the mathematical introduction to the various schemes that are to be used in multi-physics problems, especially in transient reactor analysis scenarios, let us delve into the physics and see how deeply the different physics in nuclear reactor analysis are coupled.

CHAPTER III

NUCLEAR SYSTEM – A COUPLED MULTI-PHYSICS PROBLEM

Nuclear systems are complex, stiff, nonlinear systems, which require stable numerical schemes to obtain high fidelity solution fields. The primary source of power production in a nuclear reactor is by means of the fission reaction that occurs in the fuel rods present in the core. But in order to determine the set of parameters, which will yield a safe, reliable, and economical reactor operation, it is vital to analyze the nuclear systems as a whole. The nuclear analysis of the core cannot be performed in an independent manner but rather it must interact strongly with other aspects of the core design. In reactor design, the core should be designed in such a way that it does not break any of the safety temperature limits on core components that might lead to failure and release of dangerous material into the coolant.

In this section, an overview of the interaction between different physics in nuclear reactor analysis is discussed in brief which is important to solve the reactor transient problems effectively.

III.1 Physics in nuclear systems

Nuclear systems can be broadly subdivided into 3 different physics that interact strongly with one another. They are

1) Neutron physics or 'Neutronics'
2) Thermal-Hydraulic physics
3) Heat-Conduction physics

Let us now discuss each of the physics and analyze the interaction between each of them along with the effect it has in making accurate calculations.

III.2 Neutronics

Neutronics is the branch of physics that deals with the calculation of neutron flux and neutron reaction rates in the different materials inside the core. It is very important to

determine the amount of fission reaction occurring in the fuel by means of which we can calculate the energy produced in the core. These reaction rates need to be calculated accurately in order to determine the power produced in a nuclear reactor and to calculate the temperature solution fields, which are strongly coupled to power.

The fission reaction that is responsible for producing power in the core is related to the neutron flux and the fission cross-section of the fuel material. The relation is given as

$$R.R_{fission} = \int_{Vol} \int_{g} \sum_{f}(r,E)\phi_{g}(r,E)dEdV \tag{3.1}$$

where $\sum_f$ is the fission cross-section of the fuel, and

ϕ_g is the neutron flux for the energy group g.

The core is composed of hundreds of different materials and isotopes, each with different cross-sections. The cross-section of a material is hugely affected by the temperature of the material and also dependent on the energy of the incident neutron. Different reaction rates other than those responsible for fission reaction can act as parasites and inhibit the rate of fission reaction in the material. Some others might be lost due to leakage from the core. The neutron flux at a particular point for a specific energy group is then obtained by writing a balance equation considering of all these different reaction rates.

The neutron balance equation or the '*neutron continuity equation'* accurately describes the neutron flux in the phase-space reactor domain.

$$\begin{aligned}\frac{1}{v_g}\frac{\partial \phi_g}{\partial t} + \nabla . J_g(r,t) + \Sigma_{t,g}(r,t)\phi_g(r,t) &= \sum_{g'=1}^{G} \Sigma_{s,g'->g}(r,t)\phi_{g'}(r,t) \\ &+ \chi_{p,g}\sum_{g'=1}^{G}(1-\beta_{g'})\upsilon\Sigma_{f,g'}(r,t)\phi_{g'}(r,t) + \sum_{k=1}^{K}\chi_{d,k,g}\lambda_k C_k(r,t)\end{aligned} \tag{3.2}$$

We can see that the neutron flux is dependent on the position in the core and the energy of the neutrons and varies with time; the angular dependence has been eliminated by integrating the neutron transport equation over all angles. Apart from the absorption, fission reaction rates and leakage rates that we talked about earlier, the neutron balance also includes the neutron scatter reaction, which essentially changes the energy group of the neutron inside the spatial domain of interest.

Finding a numerical solution to the neutron flux from Equation (3.2) is very difficult, especially when the domain is large and heterogeneous along with lot neutron energy and delayed groups. The neutron continuity equation is exact without any approximations, but contains an additional independent variable, the neutron current *J*, which is not simply related to the scalar flux. In order to solve for the neutron flux in Equation (3.2), we can close the system by making the diffusion approximation given by *Fick's Law*

$$J(r,E,\ t)\ = -D(r,E,\ t)\nabla\phi(r,E,\ t) \tag{3.3}$$

If we substitute Equation (3.3) in Equation (3.2), we then get the time-dependent Multi Group Diffusion (MGD) equation.

$$\begin{aligned}\frac{1}{v_g}\frac{\partial\phi_g}{\partial t}-\nabla.(D_g(r,t)\nabla\phi_g)+\Sigma_{t,g}(r,t)\phi_g(r,t)&=\sum_{g'=1}^{G}\Sigma_{s,g'->g}(r,t)\phi_{g'}(r,t)\\ &+\chi_{p,g}\sum_{g'=1}^{G}(1-\beta_{g'})\upsilon\Sigma_{f,g'}(r,t)\phi_{g'}(r,t)+\sum_{k=1}^{K}\chi_{d,k,g}\lambda_k C_k(r,t)\end{aligned} \tag{3.4}$$

For clarity, the diffusion equation can be expressed in operator notation. In practice, Equation (3.4) would be discretized and the operators are denoted as matrices that are dependent on the cross-section of the materials as a function of time.

The above equation can then simply written as

$$\frac{1}{v}\frac{d\phi}{dt}=(F_p-M)\phi+S_d \tag{3.5}$$

where - F_p is the prompt fission source ;

- S_d is the delayed neutron source;
- M is the net removal of neutrons (operator $\mathrm{M}\phi$) via absorption and scattering plus net leakage of neutrons to other points in the reactor;

The delayed neutron source results from the radioactive decay of the precursors. Assuming that there are *K* precursor groups, with respective decay constants λ_k , we can then write the delayed neutron source as

$$S_d(r,E,t)=\sum_{k=1}^{K}\chi_{dk}(E)\lambda_k C_k(r,t) \tag{3.6}$$

where χ_{dk} is the delayed neutron emission spectrum (different from the prompt neutron emission spectrum χ_p). The precursor concentrations are given by the precursor depletion equations as

$$\frac{\partial C_k(r,t)}{\partial t} = -\lambda_k C_k(r,t) + \int_0^\infty dE' \nu_{dk} \Sigma_f(r,E',t) \phi(r,E',t) \qquad (3.7)$$

We note that the delayed neutron source is not completely independent of the scalar flux at a particular time. The precursor equations are therefore coupled to the diffusion equation, and must be solved simultaneously to obtain the solution field for power profile in the core.

Even though the Multi Group Diffusion equation shown in (3.5) is easier to solve than Equation (3.2), it still can be complex to solve for multi-dimensional problems. Hence stable and accurate spatial and temporal discretization schemes need to be used to solve for flux and precursor fields.

For a 0-D model, the above system of equations can be collapsed into a simpler system allowing us to solve for the power and the precursor concentrations. The theory behind the derivation of the final 0-D Point Reactor Kinetics Equations (PRKE) are available in many introductory nuclear reactor theory textbooks[18] and hence will not be shown here. However, it is important to understand the definitions of the kinetics parameters since the analytical solution to PRKE for a constant reactivity case is easy to derive, which could be used to benchmark the solution procedure in the multi-dimensional cases, if all the assumptions of PRKE are met.

III.2.1 Point Reactor Kinetics Equations (PRKE)

The point kinetics equations can be derived in a very general fashion without making questionable assumptions such as the one speed diffusion approximation and a time independent spatial flux shape. This is done directly by collapsing the multi-group transport equation itself leading to a more formal definition of the kinetics parameters namely β, Λ *and* ρ. Hence provided that the general expressions are used for these parameters, the PRKE can be regarded as having a much broader domain of validity

The formal definitions and the point kinetics equations are shown below.

$$\text{Delayed neutron fraction } \beta(t) = \frac{\int_0^\infty dE \int_{Vol} dV \phi^*(r,E,0)[F_d(t)\phi(r,E,t)]}{\int_0^\infty dE \int_{Vol} dV \phi^*(r,E,0)[F(t)\phi(r,E,t)]} \qquad (3.8)$$

$$\text{Mean generation time } \Lambda(t) = \frac{\int_0^\infty dE \frac{1}{v(E)} \int_{Vol} dV \phi^*(r,E,0)\phi(r,E,t)}{\int_0^\infty dE \int_{Vol} dV \phi^*(r,E,0)[F(t)\phi(r,E,t)]} \tag{3.9}$$

$$\text{Reactivity } \rho(t) = \frac{\int_0^\infty dE \int_{Vol} dV \phi^*(r,E,0)[F(t)-M(t)]\phi(r,E,t)]}{\int_0^\infty dE \int_{Vol} dV \phi^*(r,E,0)[F(t)\phi(r,E,t)]} \tag{3.10}$$

Where ϕ^* is the adjoint flux or more generally neutron importance weight function

The final PRK equations can then be written as

$$\frac{dP}{dt} = \frac{(\rho(t)-\beta(t))}{\Lambda(t)} P(t) + \frac{1}{\Lambda(t)} \sum_{k=1}^{K} \lambda_k \zeta_k(t) + \frac{S_{ext}}{\Lambda(t)} \tag{3.11}$$

$$\frac{d\zeta_k}{dt} = \beta(t) P(t) - \lambda_k \zeta_k(t) \tag{3.12}$$

Where the modified precursor concentrations are $\zeta_k(t)$

and S_{ext} is the external source present in the system

Eventhough the above equations are general, to simplify the numerical simulation, the kinetics parameters β, Λ are taken to be constants instead of functions of time. The reactivity is calculated and updated to find the power and precursor solution fields during the transient.

Now depending on whether the model under consideration is 0-D or 1-D, Equation (3.11, 3.12) or (3.5, 3.7) respectively can be discretized to obtain the flux profile and power profile as a function of time. Further detailed analysis and discretization of the kinetics equations for both the 0-D and 1-D scenarios are explained in the next chapter.

III.3 Thermal hydraulics

Thermal hydraulics is the physics dealing with the calculation of enthalpy and temperature fields of the coolant. The coolant flowing outside the clad of the fuel pin gains enthalpy by convection and traps the heat from the core, which is then used to generate power by way of an associated steam thermal cycle. The thermal hydraulics physics and heat conduction are nonlinearly coupled due to the heat transferred from the

fuel to the coolant by means of forced convection. The temperature of the coolant is directly dependent on the temperature of the outer clad surface, which in turn is a direct function of the fission reaction rate thereby making all physics coupled to one another.

Before starting the discussion about the thermal hydraulics in nuclear analysis, it is important to understand that the current research has been suited particularly for a PWR system. Even though accommodating the changes to fit the scenarios of a BWR are not that complicated, they are outside the purview of this research but can be easily extended to.

It was important to mention that the system we are dealing with is a PWR because of the fact that there is no phase change in the coolant and heat is transferred only as sensible heat in the coolant, which requires a secondary system to generate steam in order to produce power. Two-phase flow calculations are slightly more complicated than single-phase calculations since we need to use two sets of equations governing each phase and for each flow regime in the coolant.

For all purposes in our current research in the 1-D model, we will assume a single channel flow in the core for thermal hydraulics and all calculations are performed at steady state only. This is based on the assumption that the short accident transients that we are of primary interest do not change the moderator temperature significantly since the response time for the convection process is larger than say for fuel conduction or fission reaction rate change. Hence, we will use the moderator profile at S.S and assume that the moderator properties remain invariant throughout the transient.

Now for the steady state calculation, an energy balance over the channel yields the following enthalpy profile as a function of the linear power.

$$h(z) = h_{in} + \frac{q'(z)}{\dot{m}} \tag{3.13}$$

where $\dot{m}$ is the mass flow rate across the channel

Using the enthalpy of the bulk liquid, the temperature of the bulk coolant at steady state can then be easily found. We can see that we require only the linear heat rate profile in order to find the bulk coolant temperature. Hence, the single-phase hydraulics model is linear and only depends on the fission reaction rate in the fuel and the outer surface temperature of the fuel.

The single-phase forced convection can then be employed to find out the clad outer surface temperature. We can use the Newton's law of cooling to find the heat transfer from the clad surface to the bulk coolant.

$$q''(z) = h_c(T_{wall}(z) - T_{bulk}(z)) \tag{3.14}$$

where $T_{wall}(z)$ is the temperature at the fuel pin outer surface

and h_c is convective heat transfer coefficient

By using the Dittus-Boelter correlation for a single-phase liquid, we can then find the heat transfer coefficient of the coolant using the following equation.

$$h_c = 0.023\, Re^{0.8} Pr^{0.4} \frac{k_f}{D_H} \tag{3.15}$$

where Re is the Reynolds number given as $\frac{\rho_f v D_H}{\mu_f}$

Pr is the Prandtl number given as $\frac{Cp_f \mu_f}{k_f}$

Once the heat transfer coefficient is found out, the outer clad temperature can be found using which the fuel temperature profile can be determined as will be seen in the heat conduction physics.

Then the transient energy conservation equation over the moderator sub-channel yields the following equation which can then be discretized to find moderator transient solution field.

$$(\rho C_p)_m A_h \frac{dT_m(z)}{dt} = R_{th}(T_f(z) - T_m(z)) - (\rho C_p)_m A_h \frac{v}{H}(T_m(z) - T_{m,in}) \tag{3.16}$$

Where R_{th} is the global exchange resistance between the fuel and fluid $(W/m\text{-}C)^{-1}$

v is the bulk coolant velocity , and A_h is the hydraulic area of coolant flow

In the 0-D model, the energy balance equation can be collapsed with an assumption $T_m = (T_{m,out} + T_{m,in})/2$ which can then be expressed as

$$(\rho C_p)_m A_h \frac{dT_m}{dt} = R_{th}(T_f - T_m) - 2(\rho C_p)_m A_h \frac{v}{H}(T_m - T_{m,in}) \tag{3.17}$$

From Equation (3.17), it can be seen that the moderator temperature solution field is coupled to the average fuel temperature (heat flux term) and the power (enthalpy change term). Hence, the hydraulics equations are nonlinearly coupled to the other physics. Now,

we will discuss about the heat conduction physics and see how a lumped model can be derived to combine the interdependence of these physics.

III.4 Heat conduction

Heat conduction physics deals with the conduction of the thermal energy that is stored in the fuel element to the coolant flowing around the fuel pin due to the temperature difference. The energy released by the nuclear fission reactions appears primarily as kinetic energy of various fission products. The bulk of this fission product energy is rapidly deposited as heat in the fuel material, very close to the location of the fission event. The heat is then transported via thermal conduction across the fuel element, across the gap separating the fuel from the clad, and then across the clad to the outer fuel pin surface. It is then transferred from the clad outer surface to the coolant by forced convection. The bulk motion of the coolant then carries the thermal energy up and out of the reactor core, either as sensible heat (i.e., coolant temperature rise in PWR) or as latent heat (i.e., thermally induced phase change by boiling in BWR).

In thermal design, the total energy deposition over all materials is frequently reassigned to the fuel in order to simplify the analysis of the core. We can determine the volumetric fission heat source in the core $q'''(r)$ by multiplying the fission reaction rate density for each isotope $w_f(i)$, the recoverable energy released per fission event to find

$$J_p(r,t) = \int_{Vol} \sum_i w_f(i) \int_0^\infty dE\ \Sigma_f^i(r,E)\phi(r,E) \tag{3.18}$$

Since the flux and the cross-section of the fuel vary across the reactor core, there will be a corresponding variation in the fission heat source in the core.

Now using the Fourier's law of thermal conduction, we can find the temperature field in the fuel by solving the general equation

$$\frac{\partial}{\partial t}(\rho C_p T) - \nabla.k(T)\nabla T = J_p(r,t) \tag{3.19}$$

It should be noted that the thermal conductivity k is temperature dependent for all the fuel elements in consideration, which then makes this heat conduction equation nonlinear. In case of the gap or the clad, where the temperature variations are relatively small, the thermal conductivity can be assumed to be constant.

The heat conduction equation for a cylindrical fuel pin with a uniform volumetric heat source J_p in which axial heat conduction can be ignored takes the following form.

$$\frac{d}{dt}(\rho_f Cp_f T) + \frac{1}{r}\frac{d}{dr}k_f(T) r \frac{dT}{dr} = J_p \tag{3.20}$$

For the gap and the clad, the heat source term J_p will not be present. Based on this heat conduction model, the temperature profile $T(r)$ in the fuel pin can be calculated and determined. Once the fuel temperatures are accurately calculated by solving Equation (3.19) for all the regions, we can calculate the average temperature in the fuel by using a weighted mean of the fuel centerline temperature and the fuel surface temperature. The average effective fuel temperature is then given by

$$T_{eff}(t) = wT_{CL}(t) + (1-w)T_{Surface}(t) \tag{3.21}$$

where w is the weight factor (usually 4/9)

Once this calculation is repeated over the radial dimension, the average fuel temperature as a function of the axial position can be determined to get $T_{eff}(z)$. This gives the axial fuel temperature profile, which should follow the same shape as the power profile.

III.5 Lumped model approach

To simplify the interaction between the Heat conduction and the hydraulics physics, a lumped model approach can be tried. This model lumps the net thermal resistance in the fuel pin into a single parameter dependent on the average fuel temperature. This nonlinear equation can then be solved to determine the average fuel temperature in the pin at the specific axial position $T_{eff}(z)$. The associated equations are

$$T_{eff} - T_{\mathrm{mod}} = J_p \frac{\pi R_f^2}{2\pi} R = q' R \tag{3.22}$$

$$R = \frac{1}{2\pi R_{gap} h_{gap}} + \frac{1}{2\pi k_c}\ln\left(\frac{R_{clad}}{R_{gap}}\right) + \frac{1}{2\pi R_{clad} h_{conv}} + \frac{w}{4\pi k_f(T_{eff})} \tag{3.23}$$

With this simplification, the coupled transient heat conduction and hydraulics equations can be expressed as

$$(\rho C_p)_f \frac{dT_f}{dt} = J_p A_f H + (T_f - T_m) R_{th} H \tag{3.24}$$

$$(\rho C_p)_m A_h \frac{dT_m}{dt} = R_{th}(T_f - T_m) - 2(\rho C_p)_m A_h \frac{v}{H}(T_m - T_{m,in}) \quad (3.25)$$

For the 0-D model in this research, this lumped model approach has been used extensively. The lumped temperature fields are linked to neutronics by means of the power density term J_p. This lumped model approach is based on solving the entire system of equations as a whole which is based on the monolithic coupling concept.

Now let us try to understand the interaction between the different physics that is critical to linearize the nuclear system and to obtain accurate solution.

III.6 Coupling between the different physics

It is worthy to note that the hydraulics model described above is simple in theory and is dependent on only the power profile. But the heat conduction model is nonlinear in itself and is coupled to both the hydraulics and the neutronics. And the neutron flux and reaction rates depend strongly on the cross-sections of the materials, which in turn are dependent functions of the temperature of material. Hence, as more fission occurs in the core, more thermal power is generated and the temperature in the core increases. This results in drastic changes in the material cross-sections affecting the flux and power.

The change in the macroscopic cross-section is partly due to the change in the microscopic cross-section and partly because of the change in the number density of the material with respect to temperature which in turn depends on the power distribution and hence the flux. These changes in cross-section are linked to the neutron flux deeply by means of reactivity represented by Equation (3.10). Such reactivity variation with temperature is the principal mechanism determining the inherent stability of a nuclear reactor with respect to short-term fluctuations in the power level.

Hence the reactivity can essentially be described as a sum of two contributions:

$$\rho_{total}(t) = \rho_{feedback}(t) + \rho_{external}(t) \quad (3.26)$$

where $\rho_{external}(t)$ is an external reactivity addition by means of either moving the control rods or by changing the flow rate of the moderator in the core etc. and is usually represented explicitly as a function of time.

And $\rho_{feedback}(t)$ is the feedback reactivity contribution from the different mechanisms that will be discussed in detail later.

III.6.1 External reactivity

The external reactivity is applied by means of moving the control rods in a systematic manner to control the change in reactivity to obtain the desired power level. Usually, the external reactivity applied is an explicit function of time in the form of a ramp or a step or a sinusoidal input. Due to the change in reactivity, the power will respond accordingly and hence we will observe changes in the power transient. Because of the increase or decrease in power, there is usually a feedback associated with the application of the external reactivity that stabilizes the power transient and brings the power back to a steady state.

As mentioned previously, some of the forms of external reactivity functions that have been performed in this research are

1) **Ramp input**

$$\rho_{external}(t) = at + \rho_0 \tag{3.27}$$

where a is the amplitude of the ramp reactivity in pcm/sec, and

ρ_0 is the initial reactivity

2) **Step input**

$$\rho_{external}(t) = a, \qquad t > t_0 \tag{3.28}$$

where a is the amplitude of the step reactivity in pcm

3) **Pulse input**

$$\rho_{external}(t) = a\delta(t - t_0) \tag{3.29}$$

where a is the amplitude of the pulse input in pcm

4) **Sinusoidal input**

$$\rho_{external}(t) = a\sin(\omega t), \qquad t > t_0 \tag{3.30}$$

where a is the amplitude of the sinusoidal input in pcm, and

ω is the frequency of oscillation in sec^{-1}

Based on these various mechanisms, we can simulate several transient scenarios in the computation of the solution in nuclear reactor analysis. Since the transient simulation nonlinearly couples all the involved physics, stable and accurate numerical schemes need to be developed in order to gain better insight on such scenarios.

III.6.2 Feedback reactivity

The interaction between Neutronics, the Heat conduction and Thermal hydraulics physics in the nuclear core that brings inherent stability to the system is called the *Feedback effect*. If the resulting feedback produces negative reactivity, the power in the core will decrease as a result and if it is positive, then power increases. Now let us try to understand about each of the important components in the total feedback reactivity.

III.6.2.1 Doppler feedback

Of most concern in the study of short-term feedback is the effect of the core temperature on the multiplication of the core. This can be expressed in terms of a *temperature coefficient of reactivity* α_T (pcm/^{o}C), which can be defined as

$$\alpha_T = \frac{\partial \rho}{\partial T_f} \tag{3.31}$$

Based on the above definition, the Doppler reactivity is more commonly[3] written as

$$\rho_{Doppler}(t) = \alpha_{Doppler}(\sqrt{T_{eff}(t)} - \sqrt{T_{eff,0}}) \tag{3.32}$$

If a reactor were to possess a positive α_T, then an increase in temperature would produce an increase in power causing a further increase in temperature and so on. In this sense, the reactor would be unstable with respect to temperature variations.

III.6.2.2 Moderator feedback

As the fuel temperature increases, the temperature of the moderator at the surface of the fuel element also increases. The direct implication of such a change is that the moderator density decreases leading to lesser efficient moderation. This results in the incident neutrons being at higher energies on average leading to a decrease in the fission rate and hence power. This dependence of the reactivity on changes in moderator fuel and density can be given as

$$\alpha_m = \frac{\partial \rho}{\partial T_m} < 0 \text{ (In under-moderated conditions)} \tag{3.33}$$

$$\alpha_m = \frac{\partial \rho}{\partial T_m} > 0 \text{ (In over-moderated conditions)}$$

where α_m is the moderator temperature coefficient of reactivity in pcm/^{o}C.

Using this definition, the moderator reactivity can be described as

$$\rho_{\text{mod}}(t) = \alpha_{\text{mod}}(T_{\text{mod}}(t) - T_{\text{mod},0}) \tag{3.34}$$

III.6.3 Other feedback mechanisms

The fuel and the moderator feedback are the primary effects that we are concerned in this research. But it should be remembered that the feedback effects are not limited to the above mentioned interaction. The other parameters that are responsible for producing feedback effects are

- Presence of voids in the moderator (BWR),
- Presence of Xenon –135 poison, which is one of the fission products,
- Presence of Samarium-149, which is another poisonous fission product,
- Presence of burnable poisons like Boron, dissolved in the moderator, or
- Reduction of the cross-section of materials due to burn-up

Every one of the above parameters contributes to the reactivity changes but the primary ones are due to the changes in fuel and moderator temperature. It is also important to note that in the current research, the void reactivity effect has not been considered but if needed, there would not be lot of modifications needed to include the effects.

Now, the total feedback reactivity in the system is given as

$$\rho_{feedback}(t) = \rho_{Doppler}(t) + \rho_{\text{mod}}(t) + \rho_{void}(t) + \rho_{boron}(t) + \rho_{Xe}(t) + \rho_{Sm}(t) + \ldots \tag{3.35}$$

The strong interaction between the various physics provides the physical safety and also introduces the complexity to analyze the system numerically. This strongly coupled, nonlinear, multi-physics problem involving complex transient accidents need robust and stable temporal and spatial discretization schemes to find the solution fields. This discussion will be provided in the next chapter.

CHAPTER IV

NEUTRONICS/HYDRAULICS COUPLED CODE IMPLEMENTATION

Numerical schemes require efficient implementation of the code in order to improve to determine the true performance of the method. Several non-trivial decisions regarding the interaction of the modules in the code can change the results for the scheme significantly. Hence, code design needs to performed with care to replicate existing conventional coupling codes to make improvements and introduce new design strategies for higher order time adaptation schemes.

For simplicity, the 0-D code was written in MATLAB whose performance was acceptable for the small system of equations we were dealing with. It also gave us the opportunity to use existing MATAB ODE solvers to test and benchmark the solution from other schemes.

For the 1-D model, since spatial and temporal discretization create an overhead in the computation process, C#.NET, a language introduced by Microsoft running on the Common Language Runtime (CLR) was used in order to improve performance and reduce computational times.

In the current section, we will go over some of the design choices and provide a basic overview of the code for both the 0-D and 1-D model.

IV.1 Code design

Coupling codes usually involve dedicated mono-physics solvers to handle each physics solution component separately and then use an interface to communicate with each other in a NIC strategy. This methodology derives its roots from the operator splitting technique where the single system of coupled equations is split into smaller problems which are attacked using specialized solvers.

In the current research, this serves us as the basis to start our code design. The coupled reactor code analysis package on a broader view point will include the Neutronics solver, Thermal hydraulics solver and Heat conduction solver. These solvers

will then be called upon by the interface or driver module which is responsible for supplying the required solution fields from other physics at each time step.

We will also need several utility functions that can handle the user input, data output and general linear algebra calculations that are essential to the working of the package. A detailed representation of the various modules that interact with the primary User Interface (UI) is shown in Fig. 5.

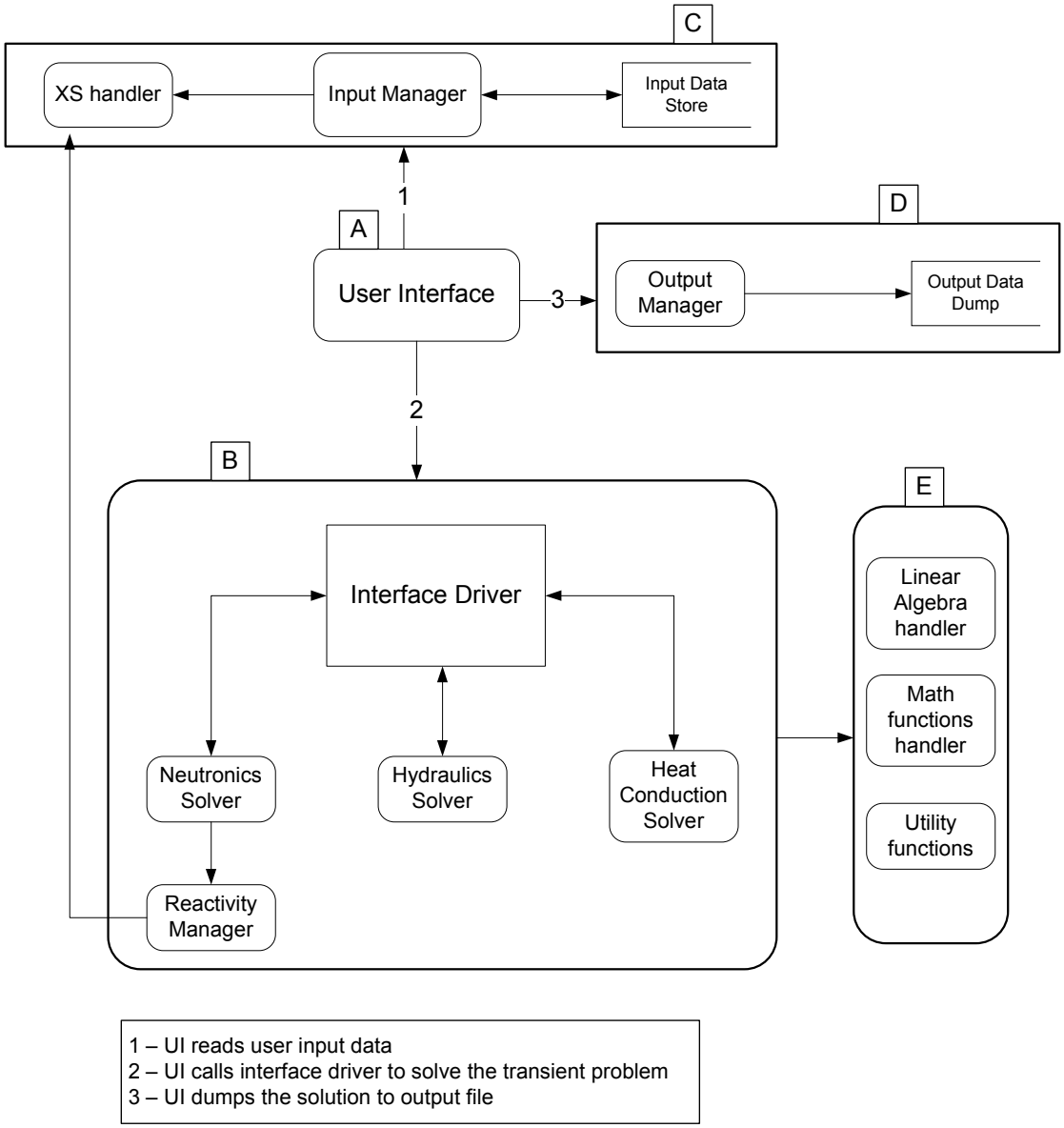

Figure 5: Data flow model

There are 5 important modules in the code design. They are

A. User Interface
B. Core solver
C. Input manager
D. Output manager
E. Math operation handler

The details given above are primarily based on the 1-D code which is more object oriented in nature. But, the code design for the 0-D model in MATLAB also follows the same pattern although being much simpler in nature.

In the following section, the internals of each of these modules will be discussed.

IV.2 User Interface

This is the primary interface to the user of the package. It allows the user to make the decision on the kind of transient to be applied (Ramp, Step or Sinusoidal etc.,) to the reactor configuration specified and several other functionalities of the package including Rod worth calculation, Adjoint flux calculation, Kinetics parameter calculations in 1-D model. In all these cases, the UI module interacts with the core solver to provide the necessary data to calculate the transient solution fields. Another functionality in the UI module is the plotting routines that give the user an idea of the average Power, flux and temperature distributions in the reactor.

IV.3 Core solver

This is the main code that is responsible for the calculation of transient solution. It includes the Interface driver, the solvers for Neutronics, Hydraulics and Heat conduction physics and several small utility functions.

IV.3.1 Driver

The core solver primarily contains the Interface driver module which acts as the key communicator between the various physics solvers. The interface is responsible for managing the time steps during transient calculation, making appropriate solution predictions for the different physics and calling the individual physics solvers in the order specified by the user. The details on whether feedback coupling is performed and whether a staggered or a simultaneous update operator-split coupling is used are handled by this driver.

In conventional coupling, an interface similar in nature is used to interface different codes. The proposed schemes to include Picard iterations and solution prediction that are handled by our current driver can efficiently be implemented in the same way for conventional interfaces also. This promises minimal change in existing coupling strategy with improvements in the solution field accuracy as will be shown in the results section.

IV.3.2 Neutronics solver

The neutronics solver is a dedicated solver for finding out the multi group flux solutions and the reaction rates (Power) in the reactor domain of interest. A Finite Element Discretization (FEM) is used to discretize space in this module. It can handle heterogenous configurations with varying refined meshes inside each physical mesh. The code uses piecewise linear basis functions and Gauss-Legendre quadrature for numerical integration.

In the steady state calculation, a power iteration to calculate the fundamental mode (K_{eff}) and its associated eigen vector (*Flux*) is performed. In the transient neutronics code, the system is always considered critical before transient happens. Hence, the total fission source F is normalized by K_{eff} found out in S.S calculation. Then a null transient simulation leads to a constant power ($K_{eff}=1$) as expected.

At $t=0^+$, a transient can be initiated by either moving the control rod in the reactor or by introducing a homogenous reactivity change by modifying the thermal absorption cross-section of the materials. This change in cross-section due to control rod is simulated using a separate cross-section library for rodded and unrodded materials. Then, the average cross-section in a mesh is computed using the weighted average formula given below.

$$\Sigma = H_r \Sigma_{rodded} + (1 - H_r)\Sigma_{unrodded} \tag{4.1}$$

And

$$H_r = \frac{Rod\ height\ in\ mesh}{Total\ height\ of\ mesh} \tag{4.2}$$

Once the cross-sections are calculated based on rod insertion height and temperature fields from other physics, the operator matrices to be solved are assembled accordingly and boundary conditions are applied. This assembly procedure needs to be done every time the cross-sections change during the transient.

In brief, the Neutronics solver is called by the interface driver with the fuel temperature and moderator density as input values. Using these, the cross-sections are found out from the table and matrices are constructed. Then an appropriate time discretization technique is used to solve the system to obtain the flux and precursor concentration values. Once this is done, the average power is computed and returned to the driver.

It is also important to remember that in a multi-group scenario, the group flux equations are coupled to one another by the scattering matrix. To resolve this, an iterative procedure more commonly known as *Thermal iterations* is performed to obtain converged flux solutions.

A detailed description of the spatial and temporal discretization procedure in neutronics is shown in Appendix (A).

IV.3.3 Cross-section manager

A dedicated cross-section manager module is implemented which is responsible for calculating the 2 groups material cross-sections for a given fuel temperature and moderator density. The cross-section table used in the current research is for the MSLB benchmark calculation[3] that provides data for several materials with K_∞ ranging from 0.8-1.3. A sample table for the material cross-section is shown in Appendix (B). The reactivity manager used by the Neutronics module calls the cross-section manager in the 1-D model to obtain the modified cross-sections based on the input from other physics. This is the prime mechanism by which feedback from other physics is included in Neutronics.

IV.3.4 Thermal hydraulics solver

The hydraulics solver is responsible for calculating the moderator temperature and density based on the power profile calculated by Neutronics, supplied as input by the interface driver. The spatial meshes needed are already determined by the Neutronics module which is used by the other solvers as the basis.

The functions of the hydraulics solver include the calculation of properties of the moderator at the specified temperature namely Density (ρ), Conductivity (k), Specific heat capacity (C_p) and Viscosity (μ). These properties are used to calculate the Heat transfer coefficient (HTC) using the Dittus-Boelter correlation for single phase fluid

which will be used to calculate the overall thermal resistance for heat transfer from fuel surface.

A simple energy balance at S.S can be performed to obtain the average moderator enthalpies at each mesh which is then used to calculate the moderator temperature and density. This data is given back to the interface driver to be supplied to other solvers as parameters.

In transient mode, the solver does not calculate the solution based on fuel temperature and power distribution since a simplifying assumption that moderator properties do not vary much during the short transient that are simulated in current research. Since it is not complicated to implement this feature, it could be added later to provide more accurate coupling between different physics.

IV.3.5 Heat conduction solver

The heat conduction solver is responsible for calculating the radial and axial fuel temperature distribution in a single fuel pin. Based on the power distribution and moderator temperature, thermal resistance supplied by the interface, the radial heat conduction equation can be solved.

The radial discretization is performed using a Finite Difference (FD) technique in the fuel pin and the axial meshes calculated by Neutronics are used as is to find axial average temperature distribution. The number of regions in the fuel and clad can be controlled by user inputs. The dimensions of the fuel pin along with the convection coefficient of the gap are also specified by the user.

Since the physics is inherently nonlinear in nature, a fixed point iteration is performed at each mesh to determine the converged radial temperature distribution in the fuel. The averaging of the fuel temperature is then performed using Equation (3.21) at each axial mesh based on the centerline and surface temperatures calculated from the radial discretization equation.

IV.4 Input manager

The input manager module is a simple data XML (Extensible Markup Language) reader which stores the user input data in an easily accessible data structure. This object is then propagated to the core solver to set all the required parameters for the calculation of the solution fields which include the reactor dimensions, tolerances for iterations,

maximum number of nonlinear iterations, interface iterations, material configuration, discretization details and run parameters. A sample Input file is shown in the Appendix (C).

IV.5 Output manager

The output manager module is a data writer which stores the solution field object structures to easily readable XML files that can be used later for analysis or plotting purposes.

IV.6 Math operation handler

The calculation of solution fields involves several matrix and vector operations. To handle these computations, the usage of an efficient and fast linear algebra package is important. Based on the performance of the LAPACK modules that are based on BLAS interfaces, an 'Interop' module to use CLAPACK routines from C# were implemented separately. In the 0-D model, the MATLAB code uses inbuilt matrix handlers based on LAPACK themselves eliminating the need for an additional module.

IV.7 Time adaptation solvers

The general design covered so far does not differentiate between constant time stepping and adaptive stepping strategies. Although major modifications are necessary in the code for the inclusion of time adaptation, the changes are concentrated on the driver module alone. This still opens up the possibility of usage of this strategy for coupling existing mono-physics codes in a consistent manner.

Since adaptive solvers can be optimized a lot more flexibly than constant step solvers for the problem at hand, the implementation of the solver has to be based on standard designs. Initial design for the code was based on the inbuilt MATLAB solver ode23s by Shampine which uses the standard error controller to adapt the time steps. After simulating various reactor analysis problems, a more efficient implementation for the Implicit RK and one step Rosenbrock methods was written based on the freely available RADAU5 code by Hairer and Wanner.

The implementation details of the RADAU5 code are described in Hairer[8]. Along with the standard controller, a PI controller was also implemented for the schemes under consideration.

IV.8 Pseudo code for reactor coupling

The actual code for the 0-D and 1-D model contains more than 8000 lines of code and hence it is not feasible to include the whole code in the current thesis. Instead, for researchers who are interested in performing the coupling of multi-physics on their own, we shall provide a pseudo-code or more generally a flow chart as shown in Fig. 6, that can be used to write the code.

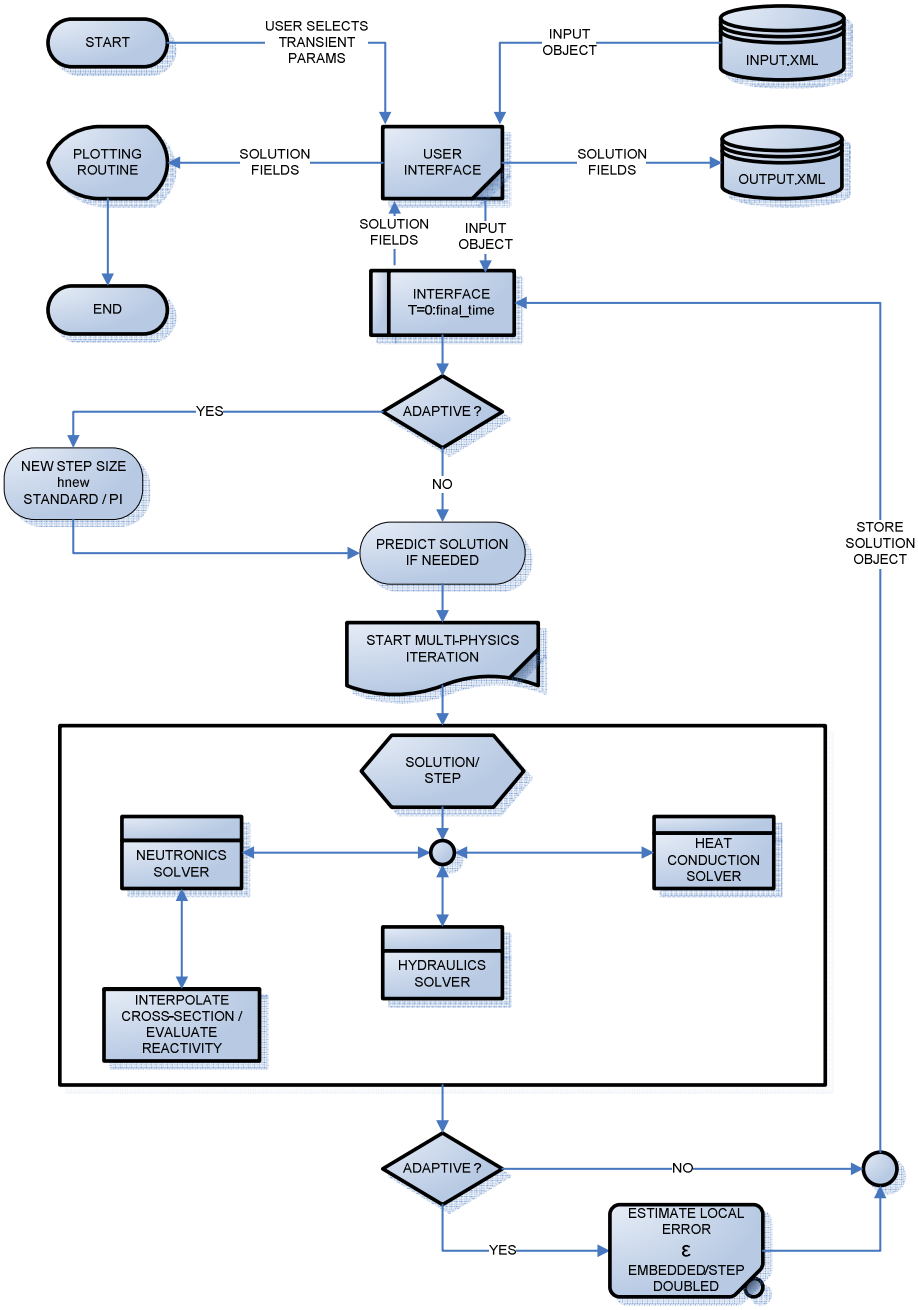

Figure 6: Algorithm for the coupled physics Neutronics/Hydraulics code

The figure is self explanatory and should provide an overview of the coupling code written for this research. The code verification and testing for the 0-D and 1-D models was performed by comparing the numerical solution to analytical solutions from simple problems. The reliability in the code was established prior to generating the results for both the models in constant and adaptive time stepping.

The results obtained are shown and discussed in the next chapter.

CHAPTER V

RESULTS AND DISCUSSION

In this chapter, the results obtained for the different models for nuclear reactor transient analysis solved using the numerical schemes discussed in Chapter II are shown and their implications are discussed. Also, the results for the adaptive time discretization schemes are elucidated with comparisons to MATLAB's standard ODE solvers for the 0-D model. It should be noted that an analytic solution exists only in the case of a constant external reactivity (Step) for 0-D and this gives a chance to benchmark the accuracy of the MATLAB solvers and solution from all other numerical schemes for 0-D and 1-D to the exact solution. Based on this, all further adaptive solution comparisons are made for the primary test cases, the Step and Ramp external reactivity additions to the system which can be simulated by changing the reactivity itself for the 0-D model and by changing the absorption cross-section as a function of space and time for the 1-D model.

All schemes implemented have a certain theoretical order of accuracy which might degrade when used for stiff, nonlinear systems if the nonlinearities in the system are not properly converged. The results shown for the convergence order of these methods match the theoretical order when completely converged or when a suitable solution prediction is used but is lost otherwise. We shall also discuss this when the results from certain strategies are observed.

V.1 0-D model

For the 0-D PRKE-Single channel hydraulics model, let us first look into the kind of transients that are analyzed. As mentioned before, for the primary test cases, the order of convergence for the different numerical schemes is found. Fig. 7 and Fig. 8 show a sample transient for a step and ramp reactivity input of 1.2$ respectively. The feedback between neutronics and hydraulics is evident due to the power turn in the transient where the negative Doppler reactivity acts as a limiting factor and brings the power back down (Power turning) to a final steady state value. The transient solution shown in the figures were calculated using a Picard iteration scheme where all the nonlinearities are fully

converged. For calculations involving order of convergence in the constant-step strategy, we shall use this particular solution method with a very fine time-step at the end of the transient, as the “Reference” solution.

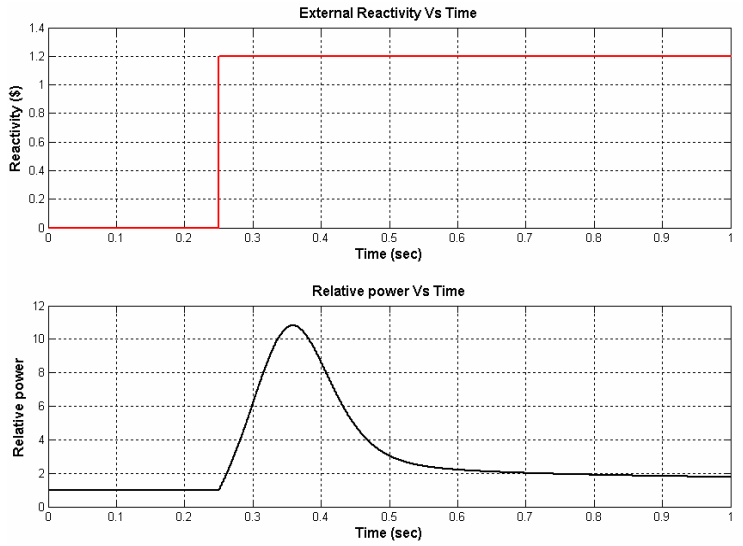

Figure 7: Step transient

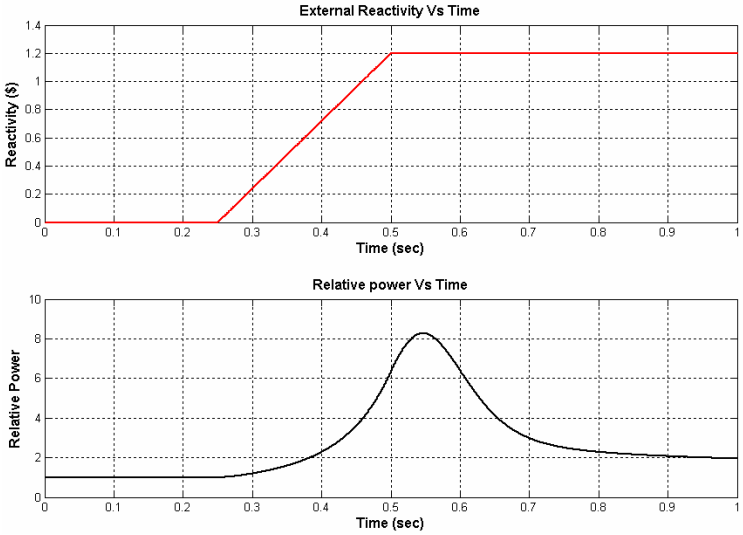

Figure 8: Ramp transient

Now, the results for both the constant and adaptive time stepping strategies are given below.

V.1.1 Constant time-step strategy

There are several discretization schemes that can be used to find out the solution in this strategy but we shall concentrate only on the theta discretization scheme. The primary idea behind usage of the B.E and C.N schemes is to make improvements in the existing conventional coupling strategy where the computational time required and the order of accuracy need to be improved.

To analyze the efficiency of convergence with the modifications suggested in Chapter II, we need to compare the transient solutions from different schemes. The aim is to find out the optimal scheme which has the lowest overhead and with a solution that is closer to the reference than conventional method.

Fig. 9 shows a calculation wherein at t=250 ms, a control rod is ejected. The ramp ejection duration is 250 ms with external reactivity amplitude of 1.2$. The reference computation for the transient was performed using a time step size of 0.5 ms. Three other computations were performed using a time step size of 10 ms with the following numerical schemes:

1) Conventional coupling paradigm – Nonlinearities are not converged between the different physics

2) Fixed-point iterations (i.e., conventional scheme iterated),

3) Explicit higher order treatment of nonlinear terms (improved prediction)

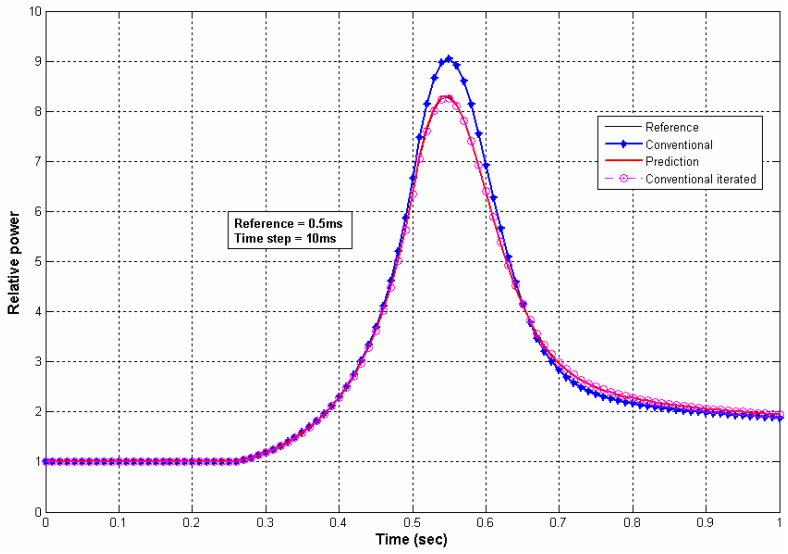

Figure 9: Comparison of power solution field between conventional and modified schemes

In Fig. 9, although not visible clearly, the FPI scheme with explicit solution prediction yields a solution that is much closer to the reference than the conventional scheme. An enlarged version of the power peak in Fig. 9 is shown in Fig. 10.

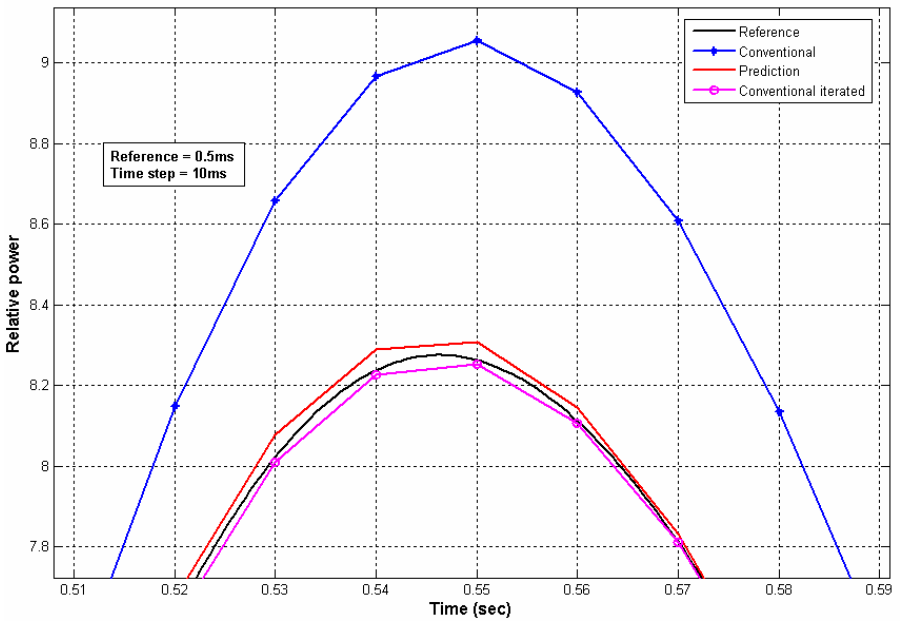

Figure 10: Comparison of power solution field: Enlarged at power peak

Fig. 10 shows that the conventional coupling scheme over predicts the power level by more than 10% whereas the other schemes are off by at most only 1%. Obviously, the improved prediction scheme was much cheaper than the Fixed-point iterative method because there are no iterations over all the physics within each time step calculation and hence the nonlinearities between the different physics are resolved by using explicit local extrapolation. The improvement in the solution field is impressive for the prediction case since the effort required to make the extrapolation is trivial but the effect of the modification, results in a solution closer to the reference even for a time-step that was much coarser (bigger by a factor of 20) than the one used to find the reference solution. But it should also be noted that the solution from improved predicted is more accurate than the conventional scheme only and not the converged solution itself.

V.1.1.1 Order of convergence

Apart from measuring the improvement in the solution from predictive methods, it is also clear that the predictive methods will restore the lost order of accuracy for the

coupled transient scenario even without outer iterations. To analyze this, a ramp transient, similar to the one plotted in Fig. 8 was simulated and the orders of accuracy of the various different schemes was found out. Remember that if the solution is not converged properly, then the observed order of convergence will be lower than the theoretical order.

Table I: Order of accuracy for the numerical schemes

Method	Converged	Predicted	Accelerated	Order
Conventional	No	No	No	1.0448
Backward Euler	Yes	No	No	1.017
Picard iterations	Yes	No	No	1.9975
Predicted FPI	Yes	Yes	No	1.9975
Improved-Acceleration	Yes	Yes	Yes	1.9975
Improved prediction (2^{nd} Order)	No	Yes	No	1.9675
Improved prediction (3^{rd} Order)	No	Yes	No	3.1165

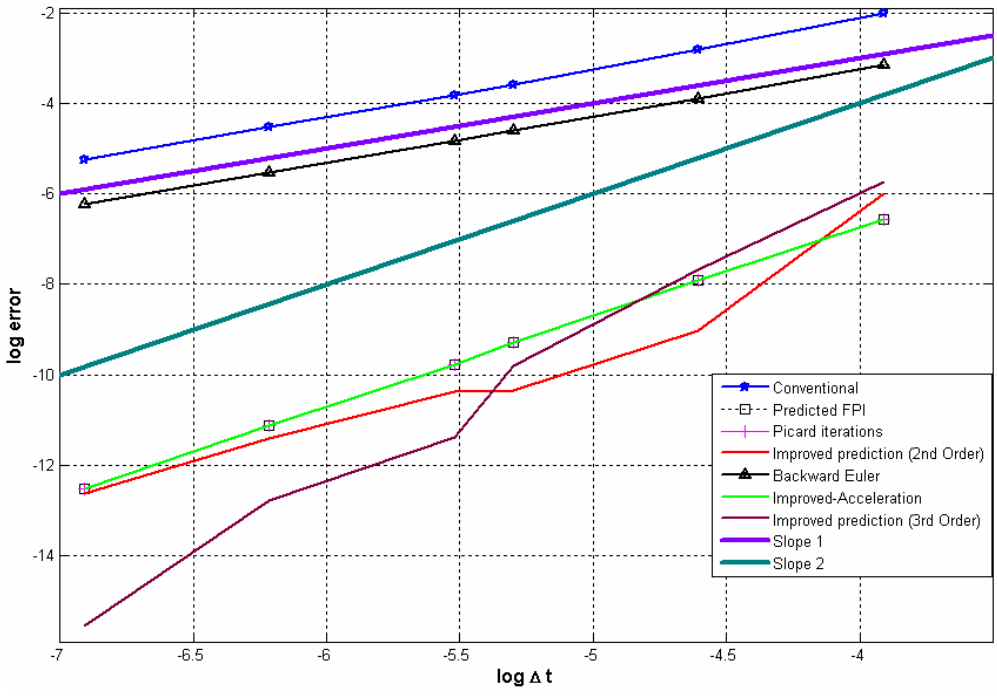

Figure 11: Order of accuracy of numerical schemes in constant stepping strategy in 0-D

Fig. 11 presents the accuracy order obtained for the different possible combination of the schemes using Picard iterations, solution prediction and acceleration. The time step sizes used to calculate the convergence order is varied from 0.8 ms to 100 ms. Table I shows the results for the order of convergence (Slope) from the Fig. 12 for the various schemes listed.

The super-convergence observed in some of the results is due to the fact being that the reference time step used is 0.5 ms and might not be accurate enough to yield the exact slope for the order of accuracy. As the reference time step is decreased, the order of convergence observed should tend towards the actual true order of the scheme.

The conventional coupling scheme only yields first order accuracy, whereas fixed-point iterations and improved prediction both yield the expected theoretical second order. Therefore these schemes are nonlinearly consistent. It is obvious that the improvement in number of iterations by Steffensen's acceleration technique does not change the order accuracy. Also, it is important to observe that whenever the nonlinearities are resolved and hence the Crank-Nicolson scheme yields second order convergence as long as the time steps used satisfy the *L*-stability criteria to avoid unwanted numerical oscillations.

The solution prediction method, both the 2^{nd} and 3^{rd} order schemes provide tremendous improvement in the convergence as compared to the conventional scheme. The 2^{nd} order prediction method only involves a trivial local extrapolation that can be handled in the driver, resulting in a superior convergence and higher accuracy even for the same time steps used for the converged solution.

Further the 3^{rd} order prediction with the time step doubling technique yields even better improvement over the 2^{nd} order prediction. The extra effort for the 3^{rd} order prediction is 50% more than the usual calculation with 2^{nd} order extrapolation done at each smaller time step. Since we are achieving the accuracy of the smaller (half) step size in the end, this technique leads to a more accurate solution field as compared to any of the other schemes discussed before. Hence if we weigh the 50% increase in CPU time to the increase in the convergence order over the Picard iteration method, this solution procedure proves to be very effective and a promising step towards achieving high fidelity transient solution fields.

V.1.1.2 Convergence acceleration

Another prime factor that has to be taken into account is the number of iterations needed to converge the nonlinearities between the physics. If there are only fewer number of iterations needed, then we could simply use the Picard iterations to find the solution since it is evident from Fig. 10 that the converged solution is always closer to the reference than the predicted solution since the nonlinearities are always resolved.

Table II: Average Fixed Point Iterations (FPI) / step

Step size	Noacc-Nopred	Noacc-Pred	Acc-Nopred	Acc-Pred
4.00E-05	3.669	2.816	3.833	2.752
8.00E-05	3.733	3.538	3.994	3.002
0.001	3.74	3.684	3.995	3.505
0.002	4.095	3.752	3.998	3.752
0.004	4.765	4.22	4.5	4.48
0.01	5.563	5.188	5.4	5.263
0.02	6.7	6.425	5.875	6.075
0.05	9.5	9.188	7.875	7.938
0.1	15.25	15	11.13	10.5

To illustrate the above factor, the number of Fixed Point Iterations/time step is plotted in Fig. 12 for different values of time steps and for each of the schemes mentioned above and is given in Table II. It is clear from the figure that using either solution prediction or Steffensen's acceleration definitely improves the number of fixed point iterations. Even though using only prediction does not yield considerable reduction in number of iterations, the synergistic effects of using both acceleration and prediction provides a reduction of more than 30% in the number of iterations per time step. On a large time scale, the total reduction in CPU time can then be considerable since on an average, only one third of iterations are needed to fully converge the nonlinearities between the different physics. Also from Fig. 12, it is clear that the usage of acceleration for finer time steps yields no perceptible improvement due to the fact that only lesser iterations are needed to converge while acceleration for coarser time steps results in a considerably

faster convergence, still retaining the order of accuracy. Hence, the acceleration techniques provide maximum boost in minimizing the number of iterations for bigger time steps and when used in conjunction with the solution prediction, the positive effects observed are very quite considerable in comparison to an un-accelerated, unpredicted converged Picard procedure.

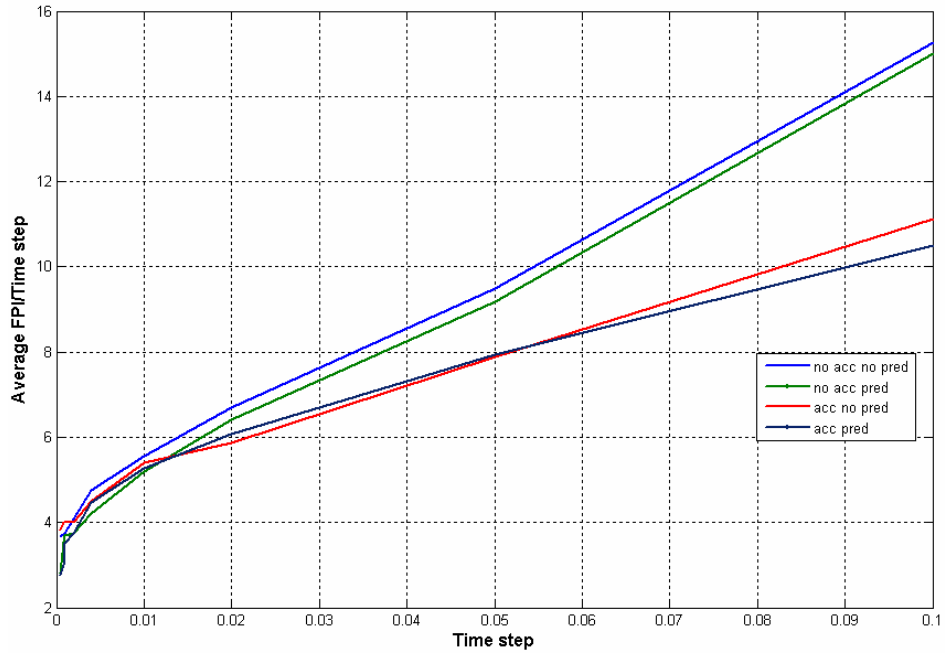

Figure 12: Efficiency of acceleration techniques

V.1.2 Adaptive time-step strategy

The results seen in the constant time-step strategy prove that the Backward-Euler and the Crank-Nicolson schemes are accurate for the stiff problems but the order of convergence is still low. The direct implication of this is that small time steps need to be used in the constant-step strategy to achieve minimal local truncation errors. Also, there is no confident method to monitor or control the truncation error that is accumulated in the solution fields. To overcome such a drawback and to have more control over the error in the solution, we need to choose the Adaptive time-step strategy with either step-doubling or higher order embedded methods involving stiffly accurate numerical schemes. We shall now discuss the results obtained using such a strategy for the primary test cases in the nuclear reactor accident transients.

V.1.2.1 MATLAB solvers – without feedback

Before the actual performance of the adaptive techniques can be analyzed, it is vital to find out the performance of the schemes for a preliminary test problem. To simplify the nuclear system, we assume a *no feedback scenario* where the neutronics is not coupled with the hydraulics and hence the reactivity feedback effects are completely neglected. This decoupling makes the problem linear and gives us a chance to analyze and benchmark the most efficient numerical scheme to be used when the feedback is introduced back in the system.

To obtain preliminary results and to make observations, a simple script in MATLAB was written to solve the PRKE equations to get the Power and 6 Precursor fields as a function of time for a given step or a ramp reactivity addition using the 7 different inbuilt ODE solvers.

Table III below provides some basic details on the MATLAB ODE solvers. For further information about the schemes and the strategies used in all the solvers, the MATLAB function reference[19] can be used as a manual.

Table III: MATLAB ODE solver details

Solver	Scheme details
ode45	Explicit one step RK (4,5) formula, the Dormand-Prince pair
ode23	Explicit RK (2,3) pair of Bogacki and Shampine
ode113	Variable order Adams-Bashforth-Moulton PECE solver
ode15s	Variable order solver based on the numerical differentiation formulas (NDFs). This is also a multistep solver like ode113
ode23s	Modified one step Rosenbrock formula of order 2
ode23t	Implementation of the trapezoidal rule using a "free" interpolant
ode23tb	An implementation of TR-BDF2, an implicit Runge-Kutta formula with a 1st stage TR and a second stage BDF2

It is important to note that the MATLAB solver estimates the local error e in the solution field and checks if this truncation error is less than or equal to the acceptable error

which is a function of the relative tolerance *RelTol* and the specified absolute tolerance *AbsTol*.

$$| e(i) | \leq \max(RelTol * Abs(y(i)), AbsTol(i))$$

Hence, when the *RelTol* or *AbsTol* is very crude, the explicit ODE solvers namely ode23, ode45 and ode113 yield unstable and physically meaningless solution fields. This is due to the fact that the crudeness in the tolerance allows the control of the time step to exceed the maximum stability step that is necessary to restrict the explicit schemes from becoming unstable. Hence, the tolerance is an important controlling factor to determine whether an explicit or an implicit scheme can be used to find the solution field. This disadvantage can be overcome by specifying a maximum time step Δt_{max} ('*MaxStep*' in MATLAB) which will restrict the time step in case of adaptive explicit time stepping thereby preserving stability and still achieving a one step solve over each step. The stability function for the explicit schemes usually provides us a good estimate of the step Δt_{max} value to be used. It should also be noted that when the Δt_{max} becomes a restricting factor, the number of steps taken to calculate the transient solution field will not be superior to the constant step strategy, although the final solution will still be accurate to the user specified tolerance. Hence, at crude tolerances, it is best to use the implicit schemes and at finer tolerances, explicit schemes offer a more viable and attractive option compared to the CPU hogging higher order implicit schemes which require multiple sub-steps inside a single step and an iteration on each step to solve the nonlinear problem.

To illustrate the above mentioned point, two figures are shown below for a very crude tolerance of *RelTol & AbsTol*= 1E-1 with an external step reactivity addition of 200 pcm.

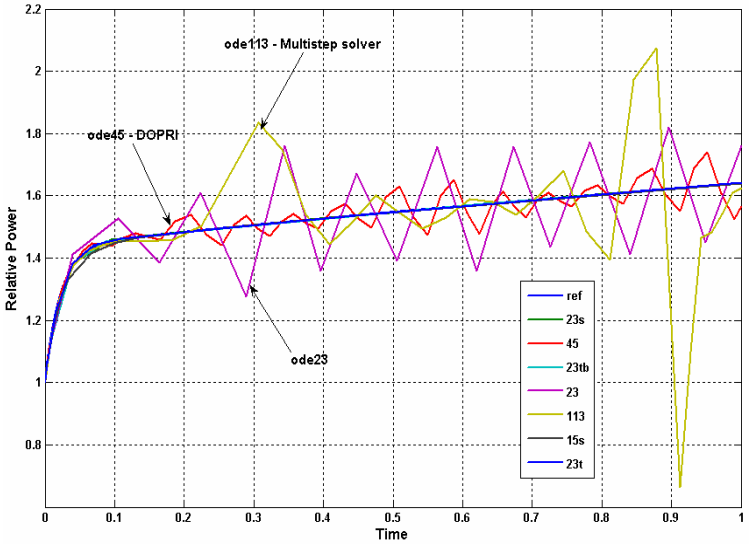

Figure 13: MATLAB solvers - unconstrained step-size

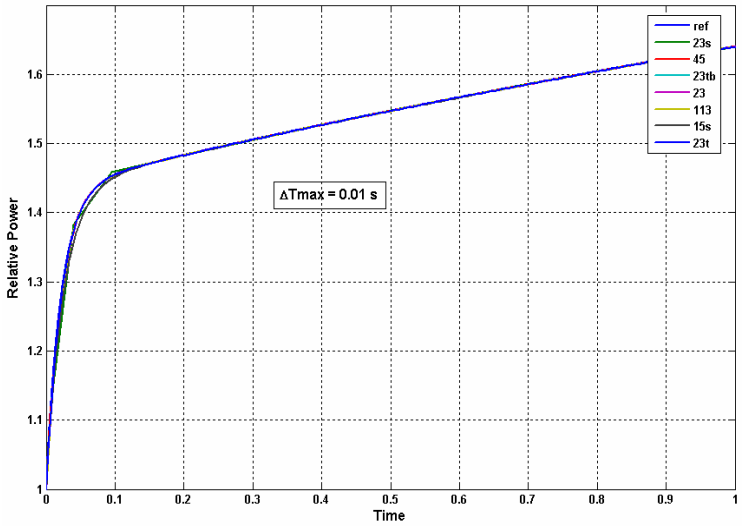

Figure 14: MATLAB solvers - constrained step-size

Fig. 13 shows the transient solution from the solvers with an unconstrained step-size where the Δt_{max} was not a limiting factor and hence the only the implicit schemes provide a stable solution. But in Fig. 14, when Δt_{max}=0.01s was an upper limiting factor for the adaptive control in all the explicit methods, the solution fields were convergent to the reference solution and the error with respect to the analytical solution was less than the user specified tolerance. This clearly proves that the explicit schemes are good candidates for solving such mildly stiff problems. If enough attention is paid on controlling the time

steps according to the stability and tolerance criteria, these methods could possibly even be used to calculate transient solutions which have local discontinuities e.g., a SCRAM event after relative power crosses the value 2.0.

Apart from proving the viable usage of the explicit schemes in an adaptive strategy, the experiment also provided other interesting results for the system under consideration. In Fig. 15, we have shown a plot of the user specified tolerance vs the number of steps needed by each of the solvers to provide a convergent solution and the corresponding data is shown in Table IV.

Fig. 15 shows that the number of steps for ODE solver 113 (variable order Adams-Bashforth-Moulton PECE solver) and 15s (variable order solver based on the numerical differentiation formulas) are both very efficient when we are dealing with higher tolerances while they do comparably well for lower tolerances if the Δt_{max} is adjusted to avoid the unstable region. Also, the ODE solvers 23 and 45 which are explicit, perform much better than the stiff solver 23s for fine tolerances.

Table IV: Number of steps required to achieve specified tolerance

Eps	ODE23S	ODE45	ODE23TB	ODE23	ODE113	ODE15S	ODE23T
1.00E-01	16	105	105	105	109	109	105
1.00E-02	18	105	105	105	109	109	108
1.00E-03	22	105	106	105	109	111	124
1.00E-04	32	105	113	108	110	118	134
1.00E-05	54	105	135	120	113	128	157
1.00E-06	107	107	205	153	118	141	236
1.00E-07	223	114	325	230	122	160	389
1.00E-08	473	127	600	402	143	191	761
1.00E-09	1013	150	1266	812	155	238	1623
1.00E-10	2177	190	2721	1745	172	307	3485

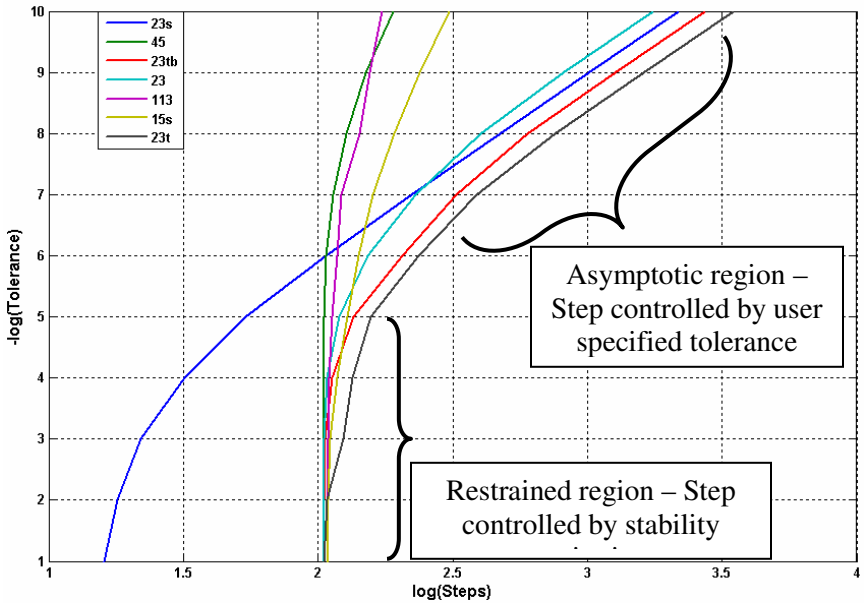

Figure 15: Plot of steps vs tolerance for MATLAB solvers

Fig. 16 shows the actual error in the solution fields with respect to the analytical solution for a 6-delayed group PRKE system without feedback. The tolerance used for the run was Eps=1E-6 with a Δt_{max}=0.01s for the explicit schemes. From the figure, we can clearly see that the stiff solvers ode23s, ode23tb and ode23t do not converge the solution to the right accuracy specified by the user during the crucial transient period but the solver ode15s provides an accurate and stable solution. Among the explicit solvers, ode113 and ode23 do converge very well and have a stable step-size control while the ode45 solver seems to calculate the local error in the solution wrongly which leads to smaller time-steps than needed thereby increasing the total CPU time actually necessary. Even though the solution from ode45 is always accurate to user specified tolerance, the step control mechanism has been poorly implemented which leads to the asymptotic Δt value being used, as clearly seen from Fig. 17.

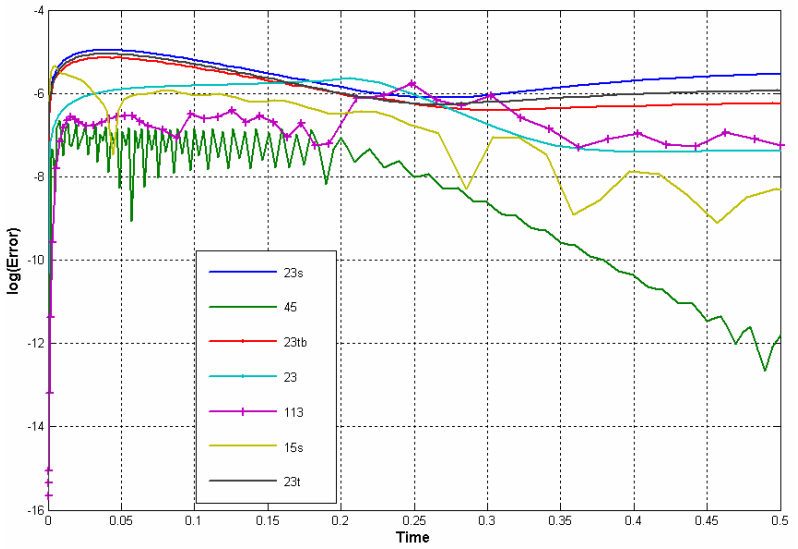

Figure 16: Plot of true error in solution from MATLAB solvers with Eps=1E-6

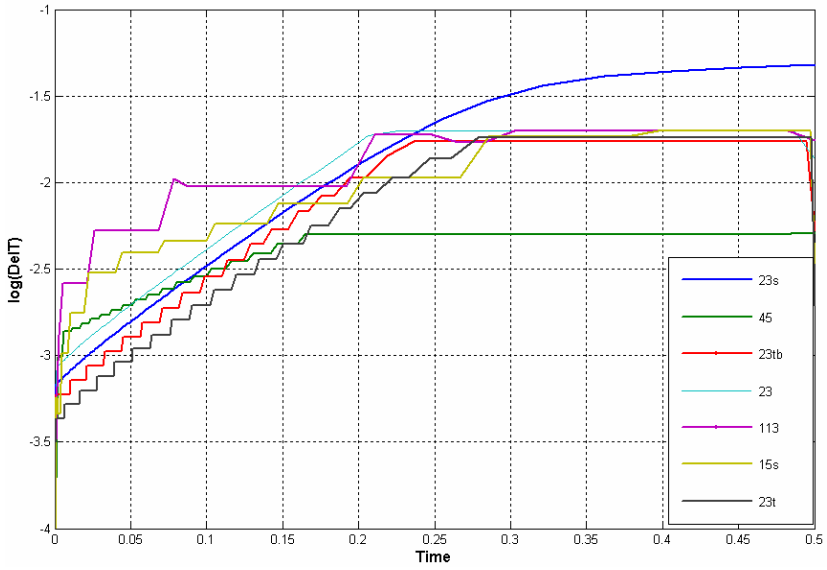

Figure 17: Plot of Δt for MATLAB solvers with Eps=1E-6

Before delving into the various other explicit and implicit schemes that can be used in adaptive time stepping, it is vital to determine a reference to compare all other calculated solution when the coupling between the multi-physics is included. This decision is based on several factors namely the number of steps necessary to reach specified accuracy and the function evaluation to achieve the same. The best candidate as concluded from Figs.

15, 16 and 17 is that ode15s outperforms the other stiff integrator and provides the solution as accurate as necessary without over-achieving unlike the ode45 solver.

Now let us look at the several other schemes implemented with step doubling and embedded time-step control strategies with *feedback coupling included* in the system.

V.1.2.2 Step doubling – with feedback

This particular adaptive stepping strategy as mentioned before in Chapter II is one of the easiest methods to implement. Since it requires that the integrator be called 3 times per step to find the local truncation error, schemes with many function evaluations/step will not perform very well. Moreover implicit schemes that require non-linear iterations are not particularly suited for this technique due to the fact that a large number of function evaluations are necessary to converge. Based on these facts, explicit and linearly implicit one step Rosenbrock 2^{nd} order scheme apart from the classic CN scheme with a 2^{nd} order prediction have been chosen to solve for the solution fields.

The results obtained using this strategy with tolerance = 1E-6, Δt_{max} = 0.01 s, Λ=1E-5, for a step reactivity change of 1.2 $ and an end-time of 5 secs are shown below.

Table V: Comparison of schemes for step-doubling strategy

Method	ERK3	ERK4	CN	ROS2
Steps	750	668	657	654
Trials	244	163	0	344
Fevals	6951	8300	5312	4640
Jevals	0	0	0	163
Average Δt	6.674E-3	7.496E-3	7.621E-3	7.657E-3
Order	3.2578	4.1455	2.0026	2.1116

All schemes tested provided a convergent solution for the parameters chosen and the unstable regions were avoided by controlling the maximum allowed step size. The results shown above in Table V indicate that the explicit schemes perform poorly for the stiff problems (S=70450) compared to the implicit CN scheme. The ERK3, ERK4 schemes undergo many oscillations in the selection of step size since they are limited by the

stability criteria during the smoother regions of the transient. This leads to a significant increase in the number of function evaluations than the CN scheme.

The ROS2 scheme has a comparable performance to the CN scheme and needs lesser number of function evaluations. Also, if the need to calculate the Jacobian matrix numerically can be eliminated using Jacobian-Free methods, the function evaluations can be reduced further.

One important outcome of this experiment is that an oscillatory pattern in the selection of time steps for ERK3, ERK4 and ROS2 schemes was observed. This explains the high number of step rejections in these schemes while the CN scheme was more stable in the prediction of the step and had no unexpected rejections. This suggests that the CN scheme is more stable among all the schemes tested in the step-doubling strategy when the stiffness is large in the system.

A similar run with decreased stiffness in the system using the parameters tolerance = 1E-6, Δt_{max} = 0.01 s, Λ=1E-4, for a step reactivity change of 1.2 $ and an end-time of 0.5 secs provide the following results.

Table VI: Alternate comparison of schemes for step-doubling strategy

Method	ERK3	ERK4	CN	ROS2
Steps	124	63	475	141
Trials	0	0	0	0
Fevals	861	620	1896	812
Jevals	0	0	0	28
Average Δt	4.065E-3	1.613E-2	1.052E-3	3.571E-3

Table VI suggests that the naïve higher orders ERK schemes do perform better than the implicit schemes for smaller transient durations. This occurs only when the step size is not limited by the stability but user specified tolerance alone since they exhibit poor performance when the stiffness in the system increases or during smoother transient periods. They might prove to be an alternate as long as the ‘Restrained region’ in Fig. 9 is completely avoided.

Now let us look at the results from the more popular embedded step-size control strategy.

V.1.2.3 Embedded strategy – With feedback

As mentioned before in Chapter II, the embedded strategy of adaptive discretization uses two different numerical schemes, one with a higher and another with a lower order of accuracy to calculate the local truncation error and to control the time-step accordingly. The theory behind the embedded scheme that is used to solve the reactor analysis problem has been tested for various stiff systems and hence gives us the basis to proceed further. Now, let us analyze the results from the different numerical schemes for a step transient with the all other parameters same as before.

The ERK3 and ERK4 methods were used in the embedded strategy to obtain the local 4^{th} order truncation error and to control the time step. Although this methodology is not truly embedded in the sense that the stages do not coincide, which actually leads to higher number of function evaluations, it is nonetheless interesting to find out the performance of this method for the given problem. Also, an embedded SDIRK *L*-stable scheme of order 3(4) by Hairer (Stiff problems: p.100) was used to solve the system of nonlinear equations. Based on some previous work on PRKE[20] the Generalized RK schemes namely the GRK4A, GRK4T proposed by Kaps and Rentrop along with the popular RADAU5 stiff integrator scheme have been used.

The results showing the number of steps, function and Jacobian evaluations for all the above schemes with tolerance = 1E-6, Δt_{max} = 0.01 s, Λ=1E-5, for a step reactivity change of 1.2 $ and an end-time of 5 secs are shown in Table VII.

Similar to the step-doubling strategy, the explicit ERK34 scheme do not produce good performance in terms of number of steps or the function evaluations needed. But the obvious winner among all the methods tried out, as seen from Table VII is the RADAU5 embedded scheme which requires the least number of function and Jacobian evaluations and produces solution fields with user desired accuracy. It is also interesting to note that the order of convergence of the RADAU scheme behaves as $O(h^6)$ as mentioned by Hairer, when large step sizes are used i.e., when the user specified tolerance is coarse enough to use very large Δt. But in the asymptotic limit, the RADAU scheme does yield

the theoretical order of accuracy proving that the nonlinearities present in the system are converged without losing the order (stiffly accurate).

Table VII: Comparison of schemes for embedded step control strategy

Method	ERK34	GRK4A	GRK4T	RADAU5	SDIRK34
Steps	1397	1916	1137	74	246
Trials	13	2231	1810	3	13
Fevals	8454	23841	15642	996	4000
Jevals	0	1897	1133	67	237
Average Δt	3.581E-3	2.638E-3	4.397E-3	6.75E-2	2.03E-2
Order	3.1272	4.0149	4.1351	4.9867	5.0789

On the other hand, SDIRK34 embedded *L*-stable method needs fewer steps than ERK, GRK schemes and proved to be a good candidate in converging the solution accurately. The implicit iterations require additional function evaluations which increase the total cost of the embedded scheme. But the performance of SDIRK for the stiff coupled problem needs to be compared with RADAU5 for several other tests to decide on the optimal scheme for usage. It was also determined that this scheme exceeds the performance of RADAU5 for crude tolerances, in terms of the number of function evaluations needed to obtain the solution.

Next, the Generalized RK schemes GRK4A, GRK4T need large number of steps and function evaluations. Also the cost of Jacobian evaluation proves to be an additional overhead to the total CPU time consumed by these methods. But the critical factor that renders these methods nonfeasible, as mentioned in Hairer [p.113], is the 'Hump phenomenon' observed in the GRK4x schemes where the step-size drops without any apparent exterior reason even though the actual solution does not have any discontinuities. To overcome this phenomenon, drastic step size reductions have to be used in the driver when the error asymptotes near the user specified tolerance. Both the Generalized RK schemes suffer from this deficiency and result in large number of step rejections which increases the overall cost of the method. This phenomenon prohibits the usage of this scheme for the stiff reactor problems.

Similar to the step doubling strategy, the stiffness in the system was decreased and the results from various schemes using the parameters tolerance = 1E-6, Δt_{max} = 0.01 s, Λ=1E-4, for a step reactivity change of 1.2 $ and an end-time of 0.5 secs are shown below in Table VIII.

Table VIII: Alternate comparison of schemes for embedded step control strategy

Method	ERK34	GRK4A	GRK4T	RADAU5	SDIRK34
Steps	129	87	75	25	78
Trials	0	2	1	2	4
Fevals	774	763	652	265	987
Jevals	0	82	70	17	18
Average Δt	3.906E-3	5.76E-3	6.667E-3	2.01E-2	6.41E-3

We can see that the ERK and GRK schemes compare well with RADAU5 performance for short time scales unlike the previous scenario where the stability restrictions force the time steps used by the ERK schemes to be limited.

V.1.2.4 Explicit vs Implicit RK schemes

To explain the difference in the performance of the ERK schemes, a simple test with increased stiffness and longer durations was conducted and the ERK4 step doubled scheme was compared to RADAU5 scheme. Figure 18 shows that once the solution values become smoother, step sizes used by ERK4 schemes tend to be restricted by stability criteria while the stiff integrator RADAU5 consistently increases the step sizes thereby performing much better for this stiff scenario.

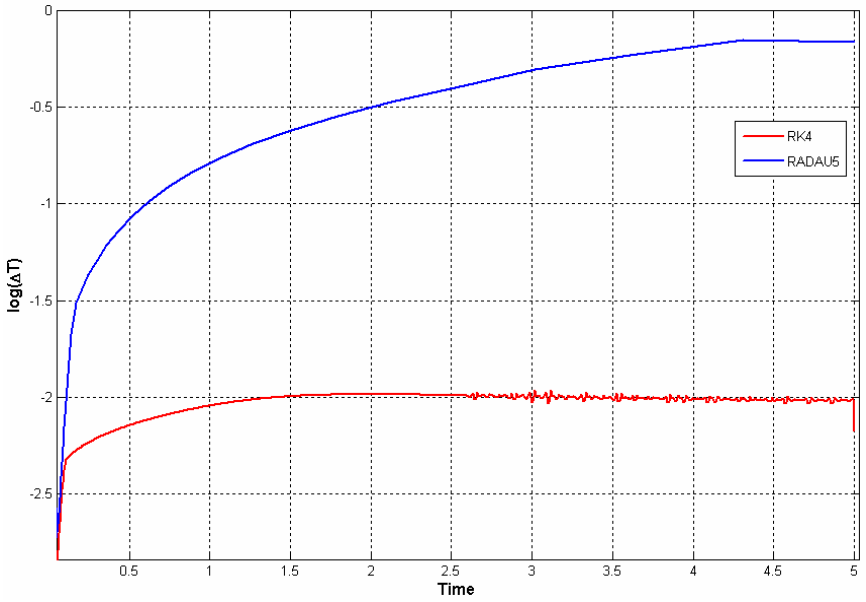

Figure 18: Plot of Δ*t* for RADAU5 & ERK4 solvers

Hence, ERK schemes are good alternatives when the transient solution needs high accuracy calculations with short durations so as to maintain the time steps in the 'Asymptotic region' rather than in the 'stability restricted' region.

V.1.2.5 Litmus test

To test and compare the performance of SDIRK34 and RADAU5 schemes, a transient problem with several SCRAM initiated events leading to local discontinuities in the solution was created. The parameters of the test are [Tolerance = 1E-6, Λ=1E-4, End-time=5 secs]. And the reactivity changes during the transient are given along with the power transient solution using ode23s as a reference in Fig. 19 and Fig. 20.

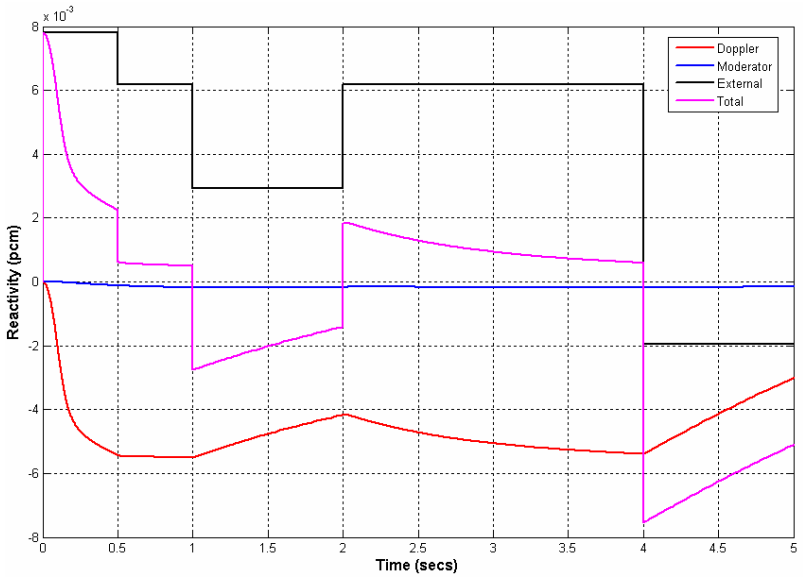

Figure 19: Reactivity vs time – SCRAM discontinuity in 0-D model

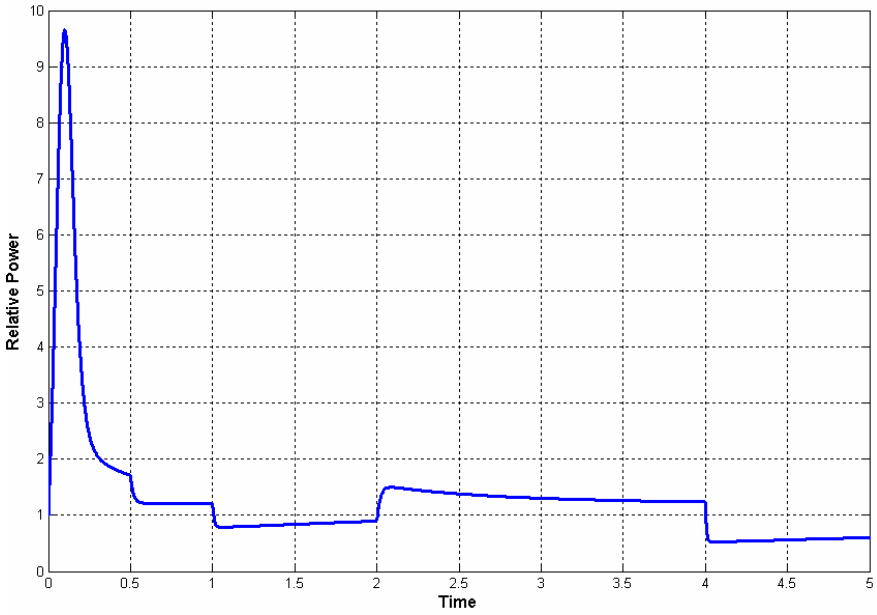

Figure 20: Power transient - SCRAM discontinuity in 0-D model

The errors in the solution calculated using RADAU5 and SDIRK34 solver are shown below in Figs. 21 and 22 respectively. From the figures, it is clear that both RADAU5 and SDIRK34 adaptive solvers resolve the discontinuities well once an increase in the estimated local error is observed. Although in comparison to the reference solution, the

local error obtained is slightly higher than the allowable tolerance at these discontinuities, the stiff solvers efficiently control time steps to obtain the maximum performance.

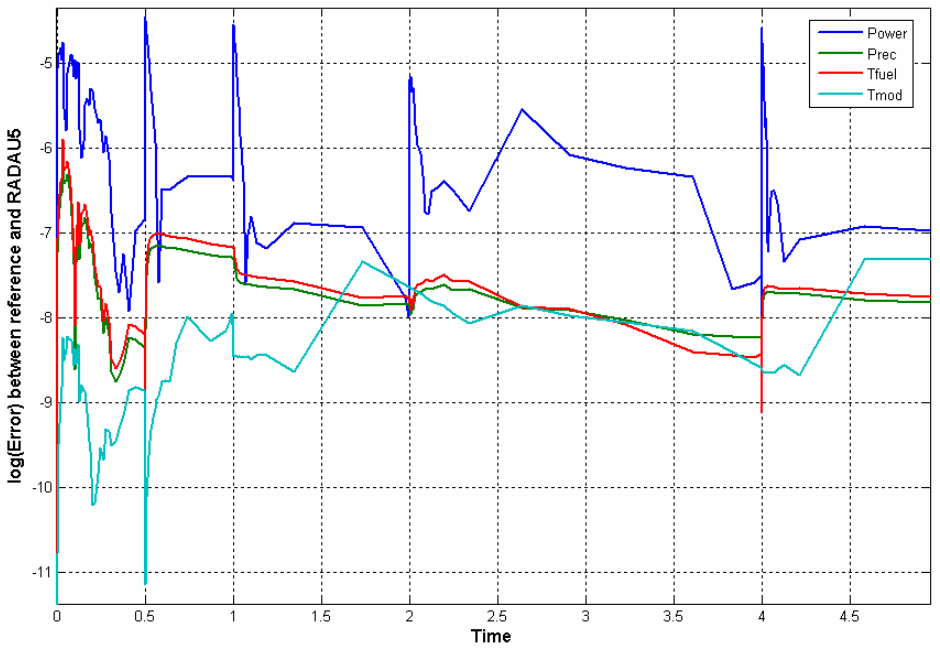

Figure 21: Error in solution fields from RADAU5 – Litmus test

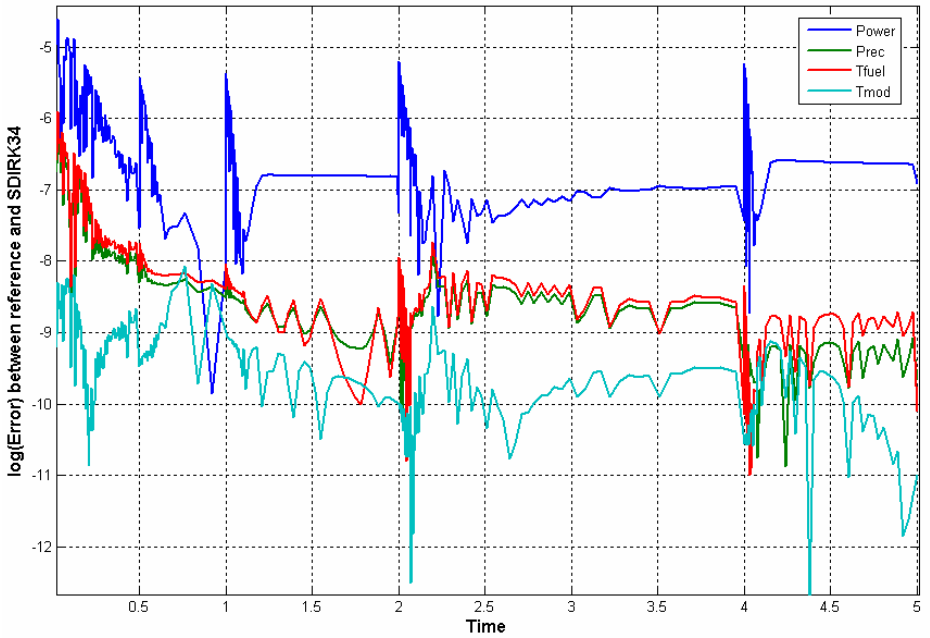

Figure 22: Error in solution fields from SDIRK34 – Litmus test

Table IX below presents the data on the number of steps and function evaluations needed by RADAU5 and SDIRK34 schemes to resolve the discontinuities.

Table IX: Comparison between RADAU5 and SDIRK34

Method	RADAU5	SDIRK34
Steps	173	362
Trials	84	11
Fevals	2302	7776
Jevals	92	353
Average Δt	2.89E-2	1.381E-2

A plot of the Δt values used by each of the solvers is shown in Fig. 23.

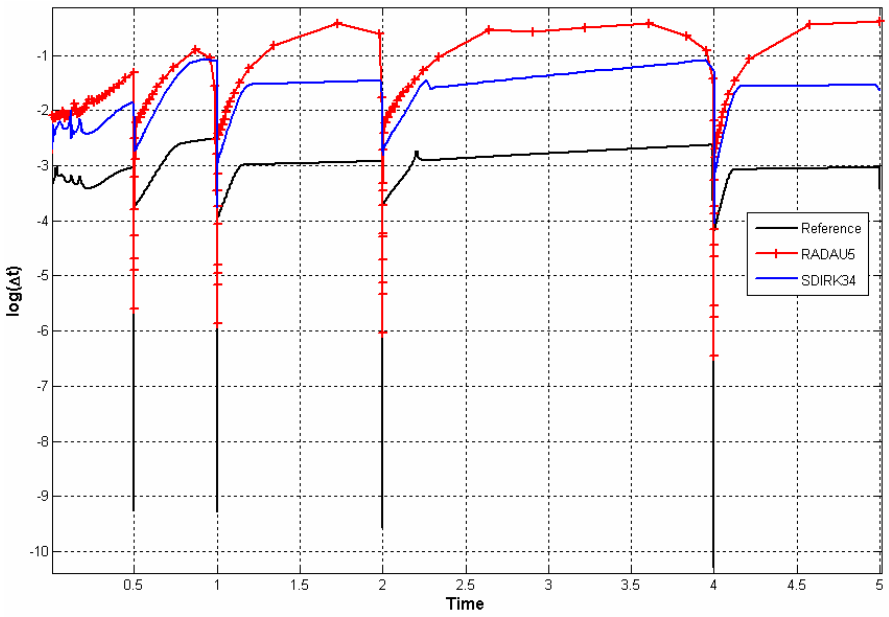

Figure 23: Plot of step sizes used by different solvers for Litmus test case

The RADAU5 scheme as given by Hairer is the best and optimal scheme among the methods tested. The number of steps and function evaluations are the least and are lower by a factor of 5 with respect to the SDIRK34 solver. Other alternate predictive step controllers like Gustafsson predictor which takes into account the number of Newton iterations taken to converge the solution for controlling the step might improve the performance for specific cases and scenarios.

V.1.2.6 Error analysis

V.1.2.6.1 True error vs estimated error

Based on the performance of the RADAU5 and SDIRK34 solvers, it is also essential to find out if the estimated error in the solver is equivalent to the true error in the solution. Such comparisons give us a complete picture on the efficiency of the adaptive solvers when finding solutions with discontinuities. Unfortunately, since an analytical solution is not available in the feedback scenario, we will resort to using the reference solution to calculate the true error.

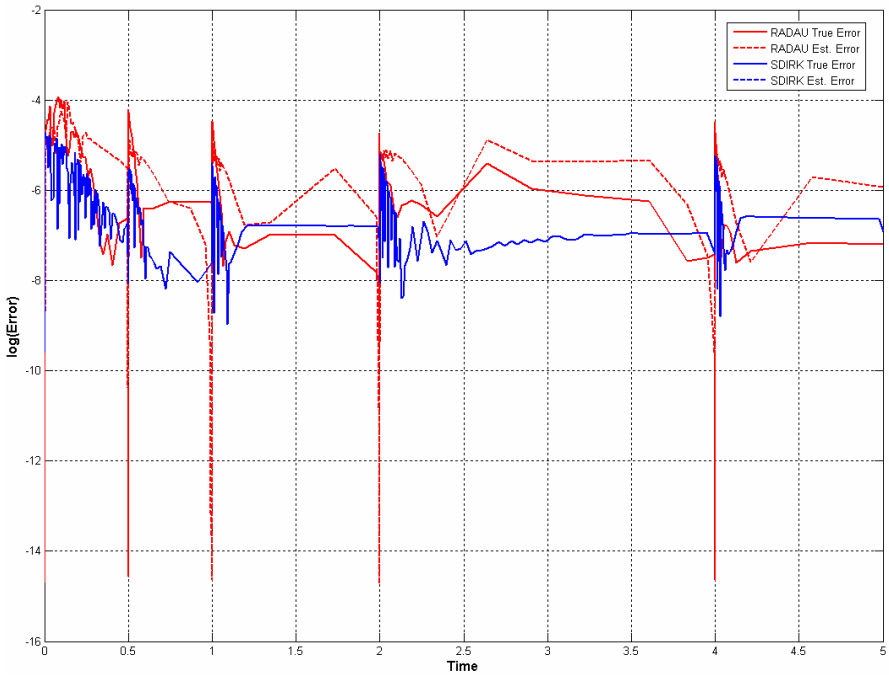

Figure 24: Actual error vs estimated error in the RADAU5 and SDIRK34 solvers

Fig. 24 shows the actual errors and estimated errors from RADAU5 and SDIRK solvers. It compares the efficiency of the error estimation for these solvers which significantly affects the time steps chosen by the scheme. The plot shows that the SDIRK that is based on the RADAU5 scheme step control code is actually more accurate in estimating the local error than RADAU5 scheme.

The effectivity is usually measured using the 'Effectivity Index' (*EI*) which is expressed as the ratio of estimated error to the true error. For a good error estimator, the

effectivity should be close to 1. From the above figure, the mean *EI* for both the schemes was found and is given below.

$$EI(\text{RADAU5}) = 11.5357 \qquad \text{and} \qquad EI(\text{SDIRK34}) = 2.69242$$

V.1.2.6.2 Component-wise errors

From the above figure, it is quite clear that both the solvers converge quite well even in the presence of discontinuities and this result validates that the error estimators in the solvers are efficient.

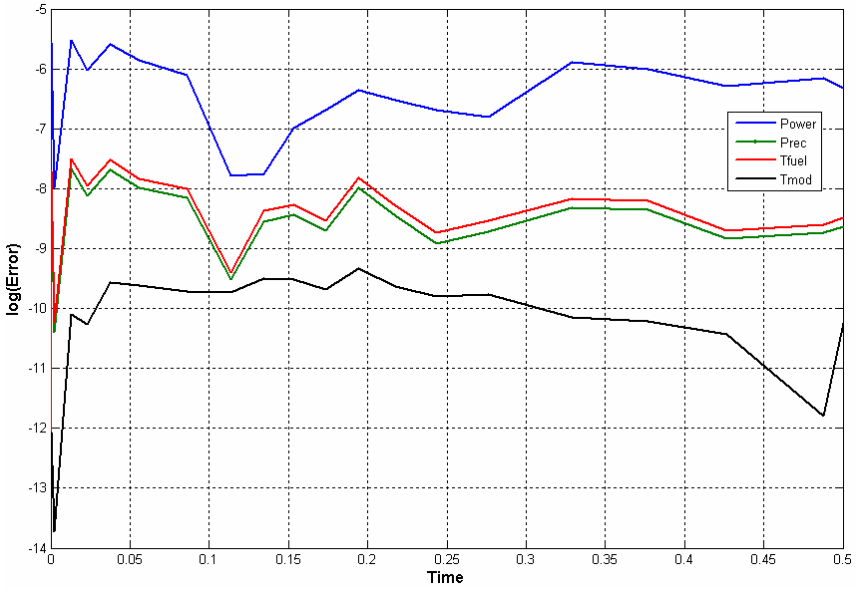

Figure 25: Error in RADAU5 solution fields – component-wise

Fig. 25 shows a plot similar of the relative error in the calculated power, 1 delayed group precursor concentration (*C*) and temperature solution fields. The important factor to note in the figure is that for the same step size, the local error in each of the solution fields is very different. The error with RADAU5 scheme for Power and Precursor, Fuel temperature differs by an order of 2.5 on average i.e., the Precursor concentration is 250 times more accurate than Power solution for the same step size and the moderator temperature is 10^4 times more accurate. The implication of this observation is that it would be wise to use different step sizes to find the solution of different components of the solution vector instead of using the same step size. This would reduce the total

number of overall calculations and function evaluations but definitely will involve complicated implementation techniques.

The results shown in the tables and the discussion above are for a transient due to a step reactivity addition but similar runs for a ramp reactivity addition were performed and similar results were obtained. Hence, the results from the implemented schemes are consistent for the system and can be applied to different other types of transients that need to be simulated like a sinusoidal or a pulse reactivity change.

V.1.3 Discussion

The observations from the results for the 0-D transient problem clearly prove that the explicit schemes do not perform well for the nuclear reactor transient problems. The SDIRK34 and RADAU5 embedded strategies are efficient in obtaining the ODE solution adaptively and surpass the performance of most MATLAB inbuilt solvers.

Among the two aforementioned methods tested for 0-D problem, RADAU5 scheme as implemented by Hairer needs the least number of steps, function and Jacobian evaluations. This scheme is very promising and the performance of the method needs to be analyzed further for complicated parabolic problems with a time dependent mass matrix, as is the case in the MGD 1-D transient problem.

Now let us analyze the results from the 1-D model by carrying forward what has been learnt from the 0-D model.

V.2 1-D model

Based on the results from constant stepping in 0-D model, a solution prediction methodology was implemented for the 1-D model but the Steffensen's acceleration was not included due to the reason that the gain in CPU time is very dependent on the way the solution driver is programmed and the platform used to run the computer code. As mentioned before in Chapter IV, a code was written in C# instead of MATLAB thereby increasing the overall speed of obtaining the result.

In the 1-D model, it is important to discretize the spatial variable thoroughly in order to eliminate the loss of order of convergence in the time integration due to a spatially un-converged or weak solution. Hence, a fine Finite Element (FE) discretization of the axial fuel for neutronics with 100 meshes over the 400cm length of the pin and a 10 region fuel

pin radial discretization for the fuel conduction were used to calculate the solution in all the following cases.

For the 1-D model, the homogenous reactivity addition is simulated by changing the thermal fission cross-section as a function of time, similar to the external reactivity function in the 0-D model. It is also imperative to note that the $1/v$ values for the chosen materials as a function of energy groups are in the order of .5456853E-07 sec/m for fast, and .2491910E-05 sec/m for thermal energy groups. This $1/v$ term, as seen in the time dependent MGD equation yields a small generation time (Λ) based on the relation in Equation (3.9), making the coupled system stiff.

Multi-parameterized cross-section data was obtained from NEA[2] for Rod ejection benchmark problems and were used for simulating all 1-D transients. The 2 group cross-section set provides the necessary data as a function of fuel temperature (T_f) and moderator density (ρ_{mod}). For feedback calculations, the cross-section was linearly interpolated between the table points to simulate the nonlinearly coupled behavior witnessed in reactor experiments as detailed in Chapter IV. The results obtained for the 1-D model for constant time-step strategy are given below.

V.2.1 Constant time-step strategy

Before we check the order of accuracy of the B.E and C.N schemes in the 1-D model for the coupled physics problem, the individual physics need to be tested if they produce the theoretical convergence orders when there is no feedback. We know that both the neutronics and heat conduction physics are inherently nonlinear in the presence of feedback but in the absence of feedback, only the heat conduction physics is nonlinear while the neutronics equations are linear. A step by step sanity check for the order of convergence in each of the individual physics is discussed in the following section.

V.2.1.1 No feedback – only neutronics

In this experiment, the fuel and moderator temperatures are calculated initially and used only as a parameter to find the cross-sections during the transient. The homogenous reactivity addition of about 0.5$ was simulated by changing the thermal absorption cross-section and the order of accuracy for the system is found and plotted in Fig. 26. It is

evident that the MGD model for neutronics without hydraulic feedback yields the theoretical order of convergence.

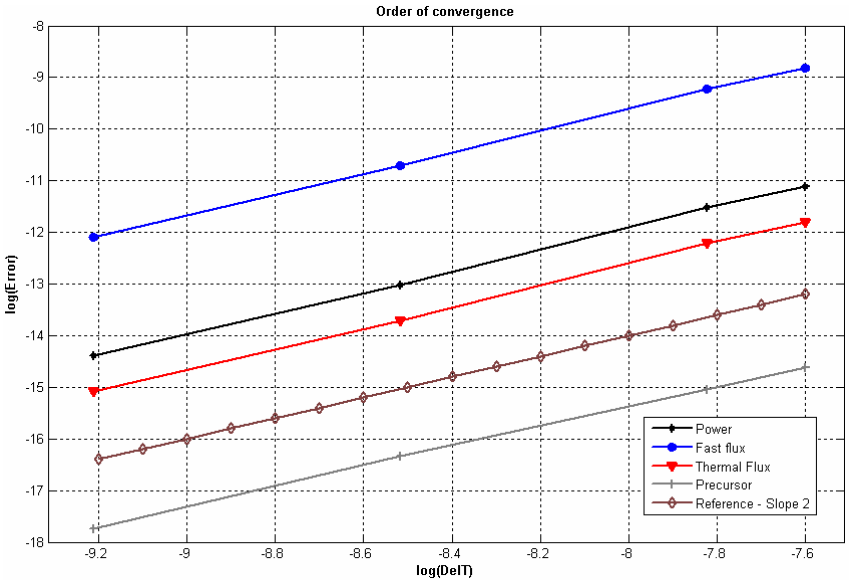

Figure 26: Order of convergence – only neutronics in 1-D model

V.2.1.2 No feedback – only hydraulics

Similar to the no feedback scenario in the previous section, the order of convergence for the nonlinear heat conduction physics for the B.E, CN scheme with a fixed power shape (cosine) was found. The transient was simulated with a homogenous change (amplitude 2.0) to the power profile and the effective fuel temperature profile was found out. Fig. 27 shows the order of convergence of the temperature field for a ramp transient in power. It can be observed that the fuel temperature is $O(h^2)$ for the simulated transient when the C.N scheme is used and is $O(h)$ for B.E.

Since the moderator temperature does not vary during the short durations for which the accident transients are simulated, the moderator transient solution field has been neglected in all calculations. This would be true even when the system is completely coupled to the neutronics which we shall discuss in the next section.

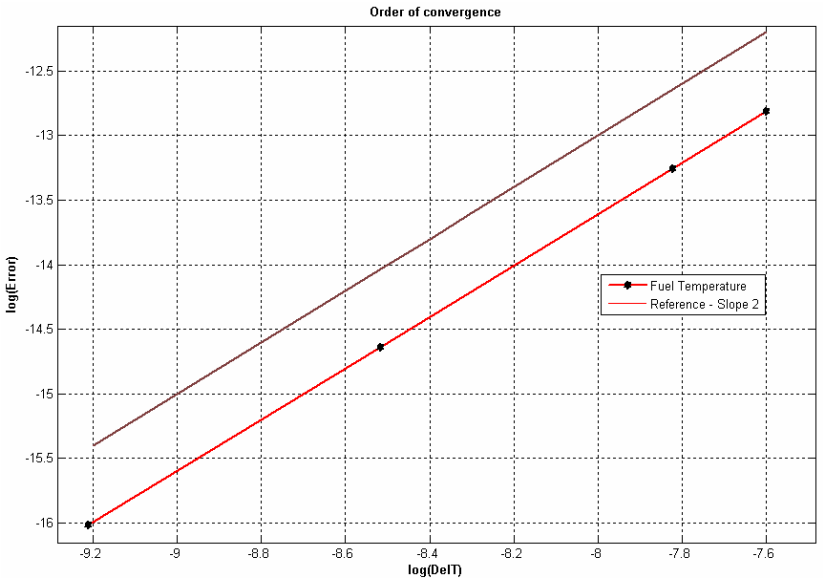

Figure 27: Order of convergence – only hydraulics in 1-D model

V.2.1.3 Fully coupled system

In the previous two sub-sections, we have checked the order of convergence of the individual physics for the B.E and CN constant time-stepping scheme in 1-D model. The results are consistent with the theoretical orders. Due to the fact that the physics components were decoupled, the solution prediction methodology was not necessary to converge the non-linearities between the physics. But when feedback is involved, the nonlinearities arising from the conventional operator-split coupling need to be resolved or else the order of accuracy $O(h^2)$ in the solutions from the C.N scheme will be degraded to a global accuracy of $O(h)$ as shown in Chapter II.

The results presented for this coupled 1-D model will involve various cases.

Case 1

A homogenous fuel pin with material number 3 from MSLB cross-section library (k_∞=0.9317748), with 2-energy groups and 1-delayed group will be used in calculation. A control rod is inserted initially for 50 cm when the reactor is critical and is withdrawn slowly with a ramp change at t^{0+} seconds. This is a ramp reactivity addition from t=[0^+, 0.1] secs to the fuel pin. The flux shapes for the fast and thermal group initially and during the transient simulated for a period of 1sec is shown in Figs. 28 and 29 respectively. It can be observed that the initial flux shape is peaked at the bottom with

vacuum boundary conditions changes to be peaked slightly closer to the middle as the transient progresses. This transient is simulated using the B.E discretization scheme with θ=1.

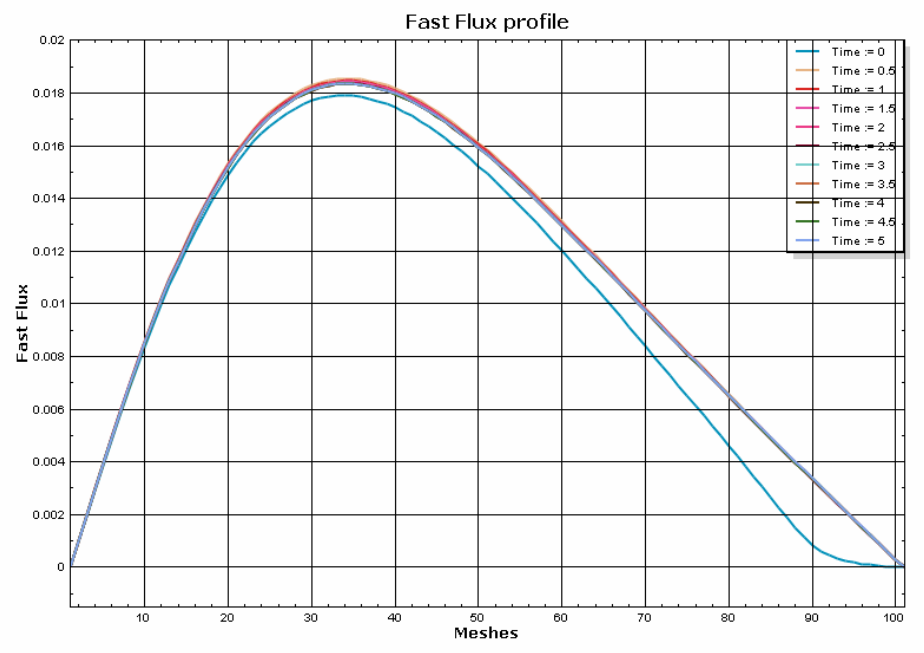

Figure 28: Fast flux transient profile – 1-D (Case 1)

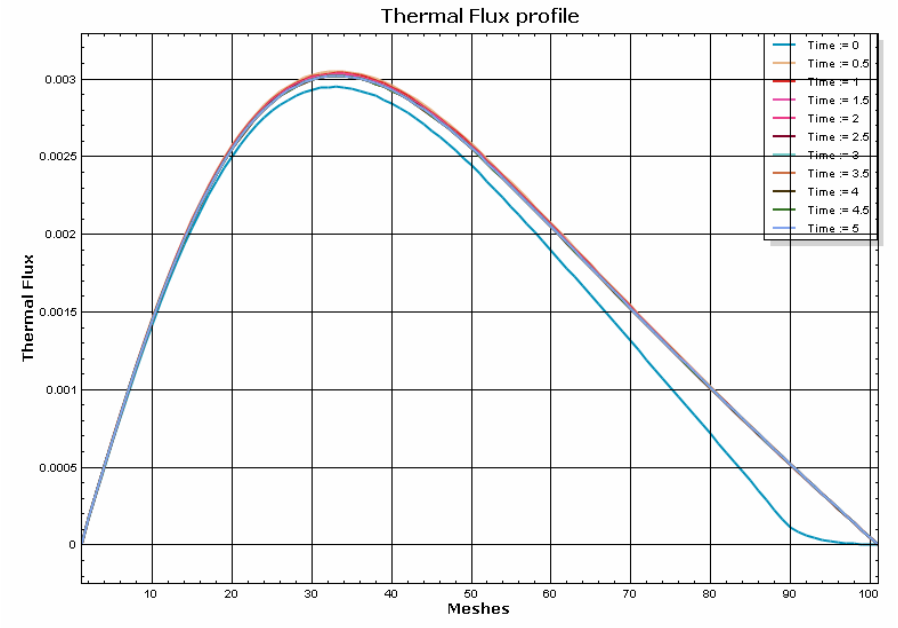

Figure 29: Thermal flux transient profile – 1-D (Case 1)

The flux initially can be noticed to be lower and when the ramp reactivity is added, the gain increases the flux slowly during the transient period and reaches the maximum amplitude at 0.1 secs. Due to the feedback mechanisms, the power and the flux then start to decrease slowly during the remaining transient until an equilibrium condition is reached $t>>0$. This is clearer from Fig. 30 showing the power transient profile exhibiting the exact same behavior of the flux.

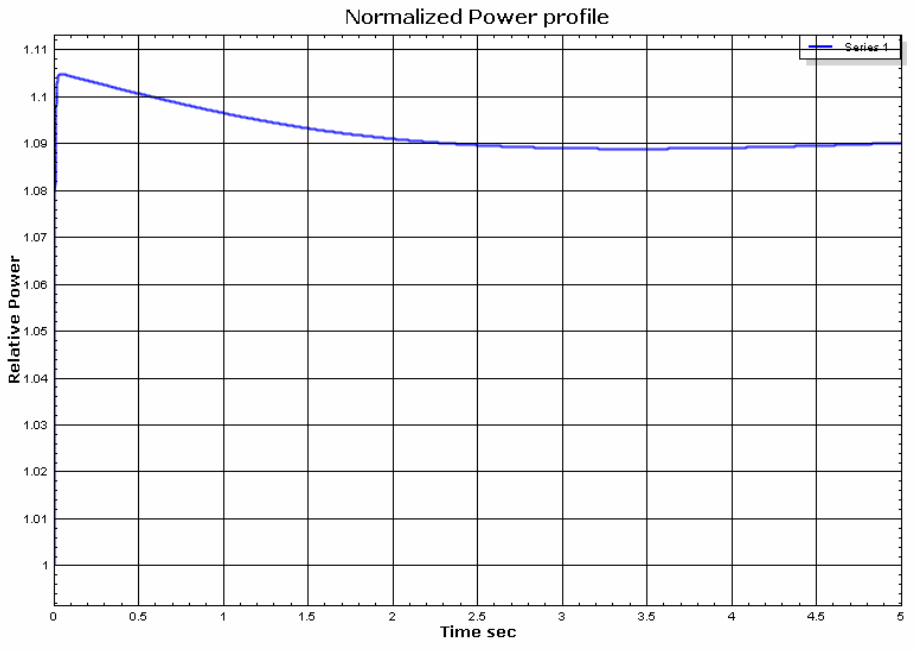

Figure 30: Power transient profile: 1-D (Case 1)

The above figures validate the theoretical behavior of the transient and so we now proceed to finding the order of convergence of different methods using the θ time discretization scheme. Then to obtain the reference solution for comparison, a fine Δt=1E-5 sec was used to obtain the solution at the end of the transient T = 1 sec. This will be used to measure the global error in the solution for different Δt ranging from [1E-4, 1E-3].

The conventional staggered scheme with both Neutronics first and Hydraulics first along with a non-staggered simultaneous procedure with no outer iteration over all the physics was tested to determine if the nonlinear inconsistency destroys $O(h^2)$ convergence. The results obtained for the global convergence order are plotted for the

C.N scheme in the conventional non-staggered update scheme and tabulated below for both B.E, C.N methods in all the above coupling strategies as shown in Table X.

Fig. 31 shows that all the solution fields yield the same order of convergence consistently. This is true for all the schemes mentioned above and the order of convergence found for the transient solution fields namely 2 group flux, 1 delayed group precursor concentration and the fuel, moderator temperatures are listed below.

Table X: Order of convergence for various method/coupling strategies

Method/Coupling strategy	Backward Euler	Crank Nicolson
Simultaneous update	1.0207	1.0268
Staggered update – Neutronics first	1.0220	0.9976
Staggered update – Hydraulics first	1.0220	1.0260

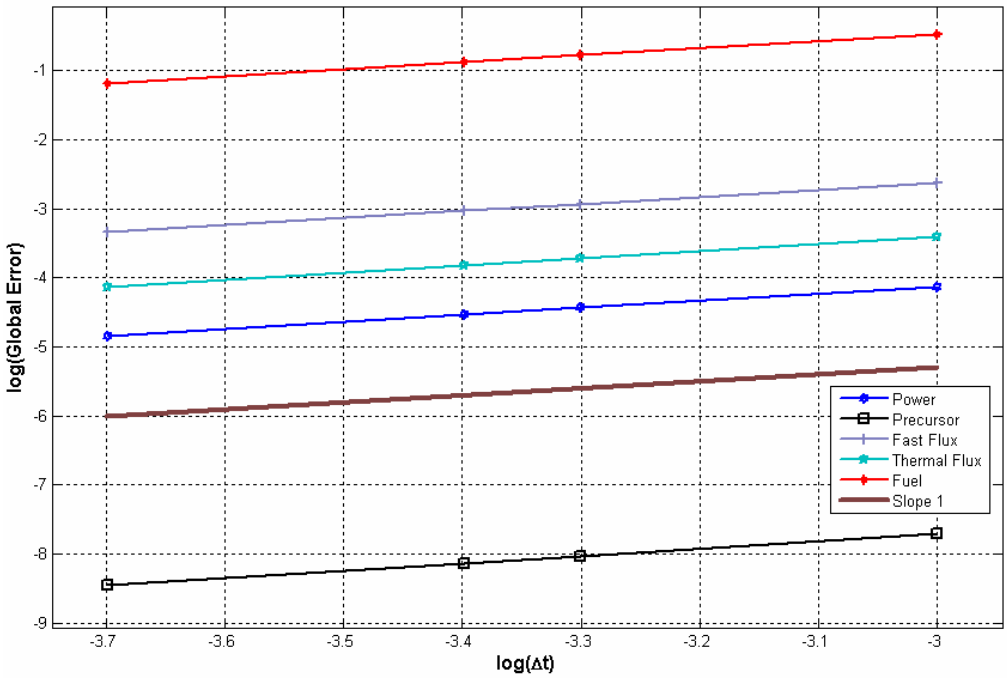

Figure 31: Order of convergence for simultaneous update coupling – 1-D model

The coupled system solution without FPI or prediction (conventional) yields a global $O(h)$ accurate solution for all the variables similar to the results shown in Fig. 6. If we perform FPI over the system, all the nonlinearities will be resolved completely and the

order of convergence will be restored. But this is very expensive for even a 1-D model since the number of matrix inversions and floating point operations limit the number of iterations that can be performed to avoid the influence of round-off errors. Instead, to save both CPU time and to get a more accurate solution, a prediction procedure is performed and tested for the 1-D model, with all the other parameters remaining the same.

The new results obtained for the convergence order using the C.N scheme are listed below.

Table XI: Order of convergence using solution prediction

Coupling strategy	Operation	Order of accuracy
Simultaneous update	FPI, No Pred	1.9875
Simultaneous update	No FPI, Pred	1.0605
Staggered update – Neutronics first	No FPI, Pred	2.1827
Staggered update – Hydraulics first	No FPI, Pred	2.1846

Results from Table XI show that the solution prediction method provides an improvement in the accuracy order for the conventional staggered coupling strategies, as we have seen in the 0-D model. But when a lagged simultaneous coupling is implemented, where each physics component is explicitly calculated using the value of other physics solution field at previous time step, the gain from the prediction methodology fails.

The results do prove that conventional staggered coupling can be improved with minor modifications to approximate the nonlinear term in the IVP to obtain the real theoretical orders of convergence but the simultaneous update coupling are $O(h)$ in accuracy, unless Picard iteration procedure is performed to converge the nonlinearities completely. But since the iterative nonlinear convergence procedure is expensive in terms of CPU time, alternates like the staggered coupling with prediction prove to be better candidates for improving existing multi-physics coupling strategies.

The scenario where the simultaneous update procedure does not provide $O(h^2)$ when prediction is applied is surprising since the staggered update procedure produces higher accuracy with the same prediction procedure. To confirm the behavior of this particular

operator-split coupling strategy, further work needs to be performed in the future and the reasons need to be analyzed.

Now, to test the consistency and improvement in the accuracy of the solution due to prediction in a staggered coupling strategy, a new scenario with alternate system properties were used.

Case 2

A heterogenous fuel pin assembly with different axial layers consisting of the material numbers 2, 3, 6 from MSLB cross-section library arranged as [2 2 6 3 3 3], with 2-energy groups and 1-delayed group will be used in calculation. Similar to the Litmus test case in 0-D model, a SCRAM is applied at t=0.5 secs.

Figs. 32 and 33 show the initial fast and thermal flux along with the transient variation in the profiles. As seen from the flux shape, the system undergoes severe changes in the course of the transient and the cross-sections as a function of the calculated fuel temperature and moderator density couple the neutronics and hydraulics physics intricately.

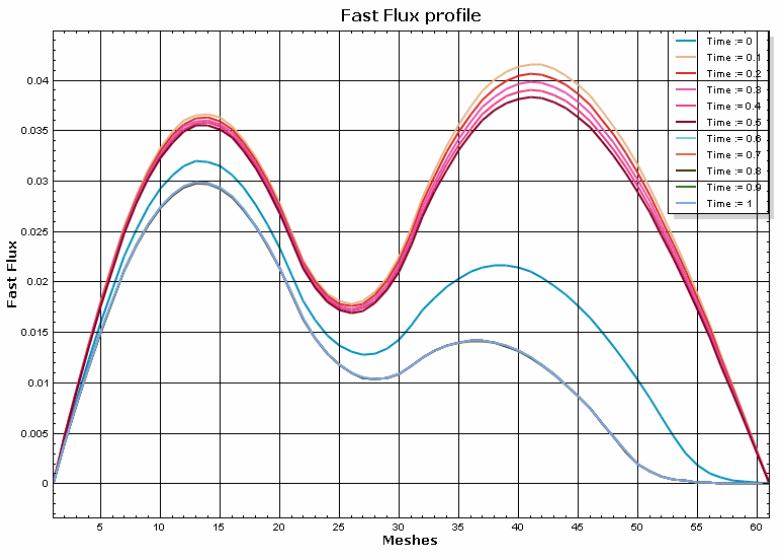

Figure 32: Fast flux transient profile – 1-D (Case 2)

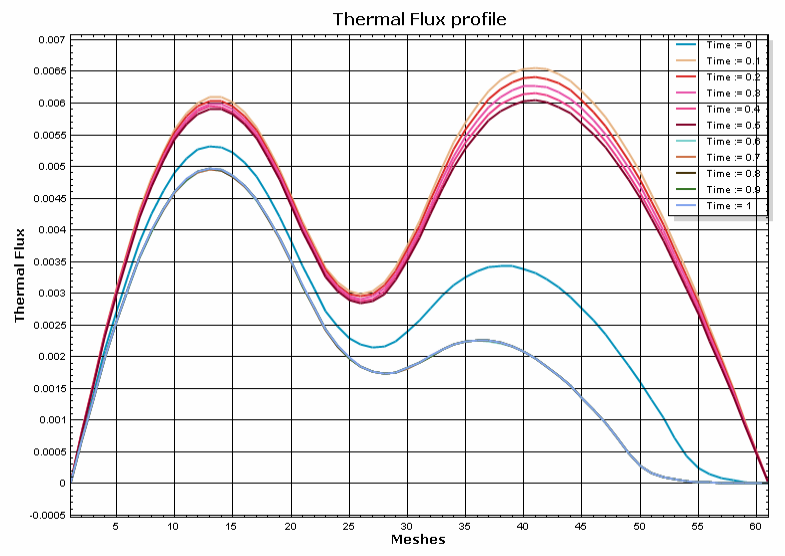

Figure 33: Thermal flux transient profile – 1-D (Case 2)

Fig. 34 shows the transient plot of the SCRAM accident scenario for the reference solution. Fig. 35 shows the enlarged picture right after the SCRAM event occurred during the transient, calculated using the conventional staggered coupling without prediction, with prediction and the reference solution.

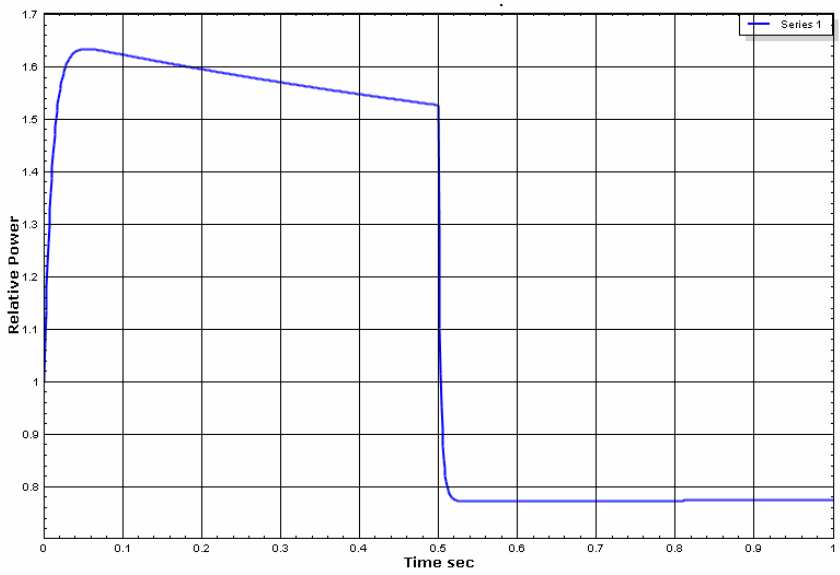

Figure 34: SCRAM Power transient plot (Case 2)

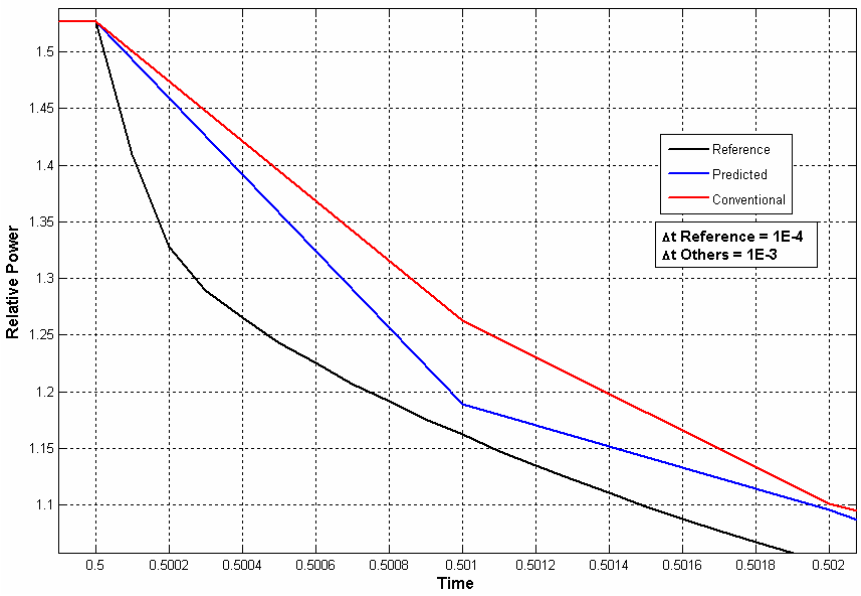

Figure 35: Comparison of conventional vs predicted (Enlarged at SCRAM)

The gain in accuracy and convergence using solution prediction is consistent with 0-D results. From Fig. 35, it is evident that the predicted solution is closer to the reference than the solution from the conventional staggered scheme, even though the prediction time step is same as the conventional solution time step. Also, the average error in the conventional scheme can be observed to be higher (8%) than the 2^{nd} order prediction solution (2.4%) by a factor of 3 which validates the efficient convergence and consistency of the scheme.

The overall gain in CPU time and accuracy by making the simple local extrapolation in the solution as compared to full Picard iterations is tremendous. The obvious advantage of this procedure is that it impacts the driver very minimally and requires the storage of only 2 preceding solution vectors in memory. Considering the memory improvement in recent years, this requirement is trivial to implement in the existing conventional black-box drivers. The only disadvantage of such constant time-stepping strategies are that it requires a lot more steps to obtain a certain accuracy and the error can be reduced only as $O(h^2)$ for the C.N scheme. Hence the alternatives to decrease the man hours spent on calculation is to use higher order implicit schemes or an adaptive time-stepping procedure as performed in the 0-D model to obtain confident solutions to user specified accuracy.

V.2.2 Adaptive time-step strategy

Based on the 0-D adaptive time-stepping results, the RADAU5, ERK34 embedded schemes were decided to be used to obtain the solution fields. The results obtained for the 1-dimensional analysis of the adaptive time-stepping schemes have not been included in this thesis but a detailed analysis will be presented in a related technical paper.

Nevertheless, the results from the constant stepping strategy do match the 0-D results for the same which implies that both the models exhibit similar behavior and that the NCC strategies devised to solve for the solution fields accurately are successful. Hence, the methods chosen are expected to work well in the 1-D adaptive time stepping case also and should provide considerable improvements in the CPU time and solution accuracy as observed before.

CHAPTER VI

CONCLUSION

The results obtained using the code written to implement the proposed strategies allowed us to test several alternatives to existing coupling strategies. Some of the key conclusions derived from the results are discussed below.

VI.1 Operator splitting vs monolithic system of equations

Conventional operator-split methods offer the advantage of being easy to implement with existing mono-disciplinary codes and allow the possibility of using solution prediction strategies or Picard iterations to regain the lost accuracy due to inconsistent coupling. The other parameters that need to be weighed between these two methods is that Picard iterations are costly in terms of CPU time since they require convergence at every time step. On the other hand, the explicit linearization using solution prediction provides only conditional stability to the scheme and hence cannot be used with larger time steps. Hence, a balance between these two approaches needs to be used while finding high fidelity solutions in reactor analysis problems.

Instead of the operator-split method, a monolithic block of the system of equations can be solved at once to obtain all the solution fields implicitly, thereby eliminating the loss of accuracy due to inconsistency. But, such calculations involve the determination of the Jacobian matrix which can be expensive to compute numerically. An alternative approach then would be to use Jacobian free schemes which could facilitate the usage of the method without actually forming the exact Jacobian matrix but instead use Matrix-Vector operations to approximate the matrix.

VI.2 Restoring accuracy in conventional schemes

Conventional operator-splitting methods provide a flexible way to treat the coupling of different physics in nuclear reactor analysis but due to the monodisciplinary nature of the existing codes and their solution procedure restrictions (CPU and hardware restrictions at the time of their development), the resulting coupling strategy was deficient and the accuracy in the solution was reduced to $O(h)$ globally. The modifications in this scheme by including local solution extrapolation does yield

improvements in the order of accuracy ($O(h^2)$ or $O(h^3)$ globally, depending on the type of the predictive scheme) with very minor changes to the driver interfacing different physics component codes. But the improvement in accuracy also comes with the price of conditional stability of the numerical scheme used which could restrict the time steps used.

Picard iterations with Steffensen's acceleration restore the global order of accuracy of the time-stepping scheme and but require several iterations/step which is computationally costly when transient solutions for large periods are needed.

Although both the above methods provide valuable improvement to the current conventional coupling methods, the optimal method for the problem in question needs to be chosen by weighing stability restrictions again increased computational time.

VI.3 Constant vs adaptive time-stepping

The results from the 0-D model are very conclusive and clearly show the advantageous of using adaptive time-stepping strategy over constant time-stepping. The reliability in the accuracy of the solution fields and the lower CPU time required to obtain the same are two crucial properties of this strategy.

Existing coupling codes in the nuclear field use constant stepping strategy and it would require some effort to change the implementation to include adaptive stepping strategy. Instead, the benefits of the solution prediction method can be realized much more immediately rather than changing the driver to include adaptivity.

VI.4 Method of choice for higher accuracy

The higher order implicit RK embedded scheme namely the 3-stage, RADAU5 scheme, exhibits stable behavior and is not restrained by the time steps chosen. Hence, the scheme offers the advantage of resolving the nonlinearities well in the reactor analysis problem during both the fast transient and the slowly varying period. This stiffly accurate scheme with a global convergence order of $O(h^5)$ provides good improvements in the results over the conventional θ-discretization scheme owing to a very efficient implementation.

There are few changes required to be made in the mono-physics codes to include this time integration scheme if a monoblock coupling strategy is used. The results from 0-D where such a coupling was used prove the feasibility of RADAU5 scheme to solve even

higher dimensional problems if appropriate modifications in existing codes can be made. This could significantly offer an improvement in the solution accuracy and reliability in the codes for solving reactor analysis transient problems.

VI.5 Future work

The efficient schemes for multi-physics coupling is a fast growing topic currently and offer insights into the behavior of nonlinear time integrators. Several time stepping strategies and schemes were tested for the 0-D and 1-D model for nuclear reactor problems, there are always modifications and improvements that can be made to existing schemes. Some of the future work that could be performed to improve our initial findings are given below:

1) Extension of the schemes to higher dimensional problems involving stronger coupling to the other physics. For instance, the improvement in solution for 3D reactor analysis problem using RADAU5 scheme will be more evident when there is a stronger coupling between neutronics and the hydraulics model. Based on the results obtained for the lower dimensional problems, the gain due to the scheme for such a complex problem will prove the coupling analysis performed here and will accelerate the move towards high fidelity modeling of reactor problems.
2) Most implicit RK schemes require the computation of the Jacobian at every time step which has a huge computational cost, especially when the spatially discretized system is large. Then, it becomes necessary to attack the problem using Jacobian-free Newton-Krylov and Rosenbrock methods. Such schemes should increase the computational speed by several orders for higher dimensional problems.
3) Existing mono-physics codes could be coupled with a custom written interface driver module to observe the real improvement in the accuracy of solution fields. For instance, a multi-dimensional transient code like <neutronics code > can be coupled and interfaced with RELAP3D and the solution fields for a rod ejection accident can be benchmarked. The analysis of the accuracy from such a calculation would validate the results obtained in the current research.

REFERENCES

1. A. F. Carlos, K.C. Park, F. Charbel, "Partitioned analysis of coupled mechanical systems", *Advances in Computational Methods for Fluid-Structure Interaction and Coupled Problems*, **190**, 24-25, 3247-3270 (2001).

2. H. Finnemann, A. Galati, "NEACRP 3-D LWR Core Transient Benchmarks", *NEA/NSC/DOC*, **25**, (1993).

3. K. Ivanov, T. Beam, A. Baratta, A. Irani, N. Trikouros, "PWR MSLB Benchmark", *NEA/NSC/DOC*, *US NRC, OECD NEA*, **8**, April (1999).

4. K. Eriksson, C. Johnson, A. Logg, "Explicit time-stepping for stiff ODEs", *SIAM Journal of Scientific Computing*, **25**, 4, 1142-1157 (2003).

5. R.B. Lowrie, "A comparison of implicit time integration methods for nonlinear relaxation and diffusion", *Journal of Computational Physics*, **196**, 2, 566-590, (2004).

6. D.A. Knoll, L. Chacon , L.G. Margolin, V.A. Mousseau, "On balanced approximations for time integration of multiple time scale systems", *Journal of Computational Physics*, **185**, 2, 583-611 (Mar. 2003).

7. E. Hairer, S.P. Norsett, G. Wanner, *Solving Ordinary Differential Equations I – Nonstiff Problems*, Springer, Berlin (1987).

8. E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II – Stiff and Differential Algebraic Problems*, Springer, Berlin (1996).

9. K. Dekker and J.D. Verwer, *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, North-Holland, New York (1984).

10. G. Dahlquist, "Stability and error bounds in the numerical integration of ordinary differential equations", *Trans. Roy. Inst. Technology*, **130**, 1-87 (1959).

11. K. Gustafsson, "Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods", *ACM TOMS*, **17**, 533-554 (1991).

12. K. Gustafsson, "Control theoretic techniques for stepsize selection in implicit Runge-Kutta methods", *ACM TOMS*, **20**, 496-517 (1994).

13. H. Watts, “A step size control in ordinary differential equation solvers”, *Society for Computer Simulation,* **1**, 15-25 (May 1984).

14. G. Soderlind, “Automatic control and adaptive time-stepping”, *Numerical Algorithms*, **31**, 281-310, (2002).

15. P. Kaps, P. Rentrop, “Generalized Runge-Kutta methods of order four with step size control for stiff ordinary differential equations”, *Numerische Mathematik*., **33**, 1, 55-68, (1979).

16. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd Edition, Cambridge University Press (1992).

17. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 2nd Edition, Springer-Verlag, New York, (1993)

18. J.J. Duderstadt and L.J. Hamilton, *Nuclear Reactor Analysis*, Wiley, New York, (1976).

19. MATLAB Function Reference, http://www.mathworks.com/ (accessed on Nov 16, 2006)

20. G.V.D Eynde, “The reactor point-kinetics equations: semi-analytical methods versus numerical methods”, Reactor Physics & MYRRHA http://sab.sscc.ru/imacs2005/papers/T2-I-72-0937.pdf (accessed on Nov 16, 2006)

APPENDIX A

Spatial and temporal discretization procedure for MGD equation in 1-Dimension

1. Equations

The general form of the equations is:

$$\frac{1}{v_g}\frac{\partial \Phi_g}{\partial t} = \vec{\nabla}\cdot\left(D_g \vec{\nabla}\Phi_g\right) - \Sigma_{r,g}\Phi_g + \sum_{\substack{g'=1 \\ g'\neq g}}^{G} \Sigma_{s,g'\to g}\Phi_{g'} + \chi_g \sum_{g'=1}^{G}\left(1-\beta_{g'}\right)\nu\Sigma_{f,g'}\Phi_{g'}$$

$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_\ell C_\ell, \qquad \text{for } g = 1..G \tag{1}$$

$$\frac{\partial C_\ell}{\partial t} = -\lambda_\ell C_\ell + \sum_{g'=1}^{G}\beta_{\ell,g'}\nu\Sigma_{f,g'}\Phi_{g'}, \qquad \text{for } \ell = 1..L \tag{2}$$

on the reactor domain D,

with initial data for the flux and precursors:

$$\Phi_g\left(\vec{r},0\right) = \Phi_g^0\left(\vec{r}\right) \tag{3}$$

$$C_\ell\left(\vec{r},0\right) = C_\ell^0\left(\vec{r}\right) \tag{4}$$

and boundary conditions for the flux:

$$\Phi_g\left(\vec{r},t\right) = 0 \quad \text{on } \partial D \tag{5}$$

The precursors' equations can be integrated in time:

$$C_\ell\left(\vec{r},t\right) = C_\ell^0\left(\vec{r}\right)\exp\left(-\lambda_\ell t\right) + \int_{s=0}^{s=t}\exp\left(-\lambda_\ell (t-s)\right)\times\sum_{g=1}^{G}\beta_{\ell,g}\nu\Sigma_{f,g}\Phi_g\left(\vec{r},s\right)ds \tag{6}$$

Substituting back into the neutron balance equation yields:

$$\frac{1}{v_g}\frac{\partial \Phi_g}{\partial t} = \vec{\nabla}\cdot\left(D_g \vec{\nabla}\Phi_g\right) - \Sigma_{r,g}\Phi_g + \sum_{\substack{g'=1 \\ g'\neq g}}^{G} \Sigma_{s,g'\to g}\Phi_{g'} + \chi_g \sum_{g'=1}^{G}\left(1-\beta_{g'}\right)\nu\Sigma_{f,g'}\Phi_{g'}$$

$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_\ell\left[C_\ell^0\left(\vec{r}\right)\exp\left(-\lambda_\ell t\right) + \int_{s=0}^{s=t}\exp\left(-\lambda_\ell (t-s)\right)\times\sum_{g'=1}^{G}\beta_{\ell,g'}\nu\Sigma_{f,g'}\Phi_{g'}\left(\vec{r},s\right)ds\right] \tag{7}$$

The following data can generally be considered independent of space and time: $\chi_{\ell,g}$, λ_ℓ, $\beta_{\ell,g}$, χ_g, v_g.

A note on the steady state:

$$0=\vec{\nabla}\cdot\left(D_g\vec{\nabla}\Phi_g\right)-\Sigma_{r,g}\Phi_g+\sum_{\substack{g'=1\\g'\neq g}}^{G}\Sigma_{s,g'\to g}\Phi_{g'}+\chi_g\sum_{g'=1}^{G}\left(1-\beta_{g'}\right)\nu\Sigma_{f,g'}\Phi_{g'}+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_\ell C_\ell \quad (8)$$

$$for\ g=1..G$$

and

$$0=-\lambda_\ell C_\ell+\sum_{g'=1}^{G}\beta_{\ell,g'}\nu\Sigma_{f,g'}\Phi_{g'},\qquad \text{for } \ell=1..L$$

(9)

Thus

$$\lambda_\ell C_\ell=\sum_{g'=1}^{G}\beta_{g'}\nu\Sigma_{f,g'}\Phi_{g'},\qquad \text{for } \ell=1..L \quad (10)$$

$$-\vec{\nabla}\cdot\left(D_g\vec{\nabla}\Phi_g\right)+\Sigma_{r,g}\Phi_g=\sum_{\substack{g'=1\\g'\neq g}}^{G}\Sigma_{s,g'\to g}\Phi_{g'}+\chi_g\sum_{g'=1}^{G}\left(1-\beta_{g'}\right)\nu\Sigma_{f,g'}\Phi_{g'} \quad (11)$$

$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\sum_{g'=1}^{G}\beta_{\ell,g'}\nu\Sigma_{f,g'}\Phi_{g'},\qquad \text{for } g=1..G$$

Letting

$$\chi_g\left(1-\beta_{g'}\right)\nu\Sigma_{f,g'}\Phi_{g'}+\nu\Sigma_{f,g'}\Phi_{g'}\sum_{\ell=1}^{L}\chi_{\ell,g}\beta_{\ell,g'}=\nu\Sigma_{f,g'}\Phi_{g'}\left[\chi_g\left(1-\beta_{g'}\right)+\sum_{\ell=1}^{L}\chi_{\ell,g}\beta_{\ell,g'}\right]$$

$$=\overline{\chi_g}\nu\Sigma_{f,g'}\Phi_{g'} \quad (12)$$

we obtain:

$$-\vec{\nabla}\cdot\left(D_g\vec{\nabla}\Phi_g\right)+\Sigma_{r,g}\Phi_g=+\sum_{\substack{g'=1\\g'\neq g}}^{G}\Sigma_{s,g'\to g}\Phi_{g'}+\overline{\chi_g}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}\Phi_{g'}\qquad \text{for } g=1..G \quad (13)$$

2. Finite element matrices

In FEM, the flux is expanded on N basis functions p as follows:

$$\Phi_g\left(\vec{r},t\right)=\sum_{i=1}^{N}\phi_{g,i}\left(t\right)\times p_i\left(\vec{r}\right) \quad (14)$$

Therefore,

$$\frac{1}{v_g}\frac{\partial\Phi_g}{\partial t}=\frac{1}{v_g}\sum_{i=1}^{N}\frac{d\phi_{g,i}}{dt}\left(t\right)\times p_i\left(\vec{r}\right)=\frac{1}{v_g}\sum_{i=1}^{N}p_i\left(\vec{r}\right)\frac{d\phi_{g,i}}{dt}\left(t\right) \quad (15)$$

$$\vec{\nabla}\cdot\left(D_g\vec{\nabla}\Phi_g\right)-\Sigma_{r,g}\Phi_g=\sum_{i=1}^{N}\left[\vec{\nabla}\cdot\left(D_g\vec{\nabla}\Phi_g\right)-\Sigma_{r,g}p_i\right]\phi_{g,i} \quad (16)$$

$$\sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g' \to g} \Phi_{g'} = \sum_{i=1}^{N} \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g' \to g} p_i \times \phi_{g',i} \tag{17}$$

$$\sum_{g'=1}^{G} \nu\Sigma_{f,g'} \Phi_{g'} = \sum_{i=1}^{N} \sum_{g'=1}^{G} \nu\Sigma_{f,g'} p_i \times \phi_{g',i} \tag{18}$$

Let us project the balance equation on the basis functions. We therefore need to define the following matrices:

- Mass matrix to be associated with the time differential:

$$M_{ij}^{gg'} = \int_D p_i p_j \delta_{gg'} \tag{19}$$

  (***M*** is block diagonal on the energy groups)

- Loss matrix (due to diffusion and removal):

$$L_{ij}^{gg'} = -\int_D \left[\vec{\nabla} \cdot \left(D_g \vec{\nabla} p_i \right) - \Sigma_{r,g} p_i \right] p_j \delta_{gg'} \tag{20}$$

  (***L*** is block diagonal on the energy groups)

- Scattering matrix:

$$T_{ij}^{gg'} = \begin{cases} \int_D \Sigma_{s,g' \to g} p_i p_j & \text{if } g' \neq g \\ 0 & \text{if } g' = g \end{cases} \tag{21}$$

  (***T*** is a full block matrix with blocks on the diagonal equal to zero because in the loss matrix we have the removal XS and not the total XS)

- Prompt fission production matrix:

$$P_{ij}^{gg'} = \chi_g \left(1 - \beta_{g'} \right) \int_D \nu\Sigma_{f,g'} p_i p_j \tag{22}$$

  (***P*** is a full block matrix, block lines are equal to zero where the spectrum χ_g is zero)

- delayed neutron fission matrix:

$$D_{ij}^{\ell,gg'} = \chi_{\ell,g} \beta_{\ell,g'} \int_D \nu\Sigma_{f,g'} p_i p_j \tag{23}$$

These matrices have a rank of $G \times N$ (or $L \times G \times N$). All these matrices, except the first one, are **time-dependent**. Using Green's formula on the diffusion matrix, we can also get:

$$L_{ij}^{gg'} = \int_D \left[D_g \vec{\nabla} p_i \cdot \vec{\nabla} p_j + \Sigma_{r,g} p_i p_j \right] \delta_{gg'} \tag{24}$$

Finally, the following notation will be used for the projected precursors:

$$c_{\ell,i}(t) = \int_D C_\ell \left(\vec{r}, t \right) p_i \left(\vec{r} \right) \tag{25}$$

($c_{\ell,i}$ is a vector)

Putting everything together, we get, for any group g :

$$\frac{1}{V}M^{gg}\frac{d\phi^{g}}{dt}=-L^{gg}(t)\phi^{g}(t)+\sum_{g'=1}^{G}T^{gg'}(t)\phi^{g'}(t)+\sum_{g'=1}^{G}P^{gg'}(t)\phi^{g'}(t)$$
$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_{\ell}c_{\ell}^{0}\left(\vec{r}\right)\exp\left(-\lambda_{\ell}t\right)$$
$$+\sum_{\ell=1}^{L}\lambda_{\ell}\exp\left(-\lambda_{\ell}t\right)\sum_{g'=1}^{G}\sum_{s=0}^{s=t}\exp\left(\lambda_{\ell}s\right)\times D^{\ell,gg'}\phi^{g'}\left(s\right)ds \qquad (26)$$

or

$$\frac{1}{V}M^{gg}\frac{d\phi^{g}}{dt}=-\sum_{g'=1}^{G}K^{gg'}(t)\phi^{g'}(t)$$
$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_{\ell}c_{\ell}^{0}\exp\left(-\lambda_{\ell}t\right)$$
$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_{\ell}\exp\left(-\lambda_{\ell}t\right)\sum_{g'=1}^{G}\sum_{s=0}^{s=t}\exp\left(\lambda_{\ell}s\right)\times\beta_{\ell,g'}F^{g'}\phi^{g'}\left(s\right)ds \qquad (27)$$

where

$$K^{gg'}(t)=L^{gg'}(t)-T^{gg'}(t)-P^{gg'}(t) \qquad (28)$$

and $F^{g'}$ is described below.

A note on vectors and matrices describing Equation (27).

1) ϕ_g is the column vector of nodal flux unknowns such that $\Phi_g = P^T\phi_g$ (P is the column vector composed of basis functions);
2) ϕ is simply the column vector composed of $(\phi_1^T,...,\phi_g^T,...,\phi_G^T)^T$ of length $N\times G$;
3) M is a time-independent **mass** matrix of rank $N\times G$; it is block-diagonal where each block represents an energy group. Each block contains the spatial integrals $\int_D p_i p_j$ and is not diagonal [except in the case of 1-D problems for which the spatial discretization used is either FD or linear Lagrange FEM with a Lobatto integration quadrature];
4) V is a diagonal matrix rank $N\times G$ containing the neutron velocities;
5) L is a time-dependent matrix of rank $N\times G$; it is block-diagonal where each block represents an energy group. Each block contains the spatial integrals $\int_D\left[D_g\vec{\nabla}p_i\cdot\vec{\nabla}p_j+\Sigma_{r,g}p_i p_j\right]$ (**stiffness** and **mass** matrices);
6) T is a time-dependent matrix of rank $N\times G$; it is not block-diagonal and each block represents the scattering between one energy group to another energy group. The way we defined it, all diagonal blocks are zero [the within group scattering has been removed to form the removal cross-section found in matrix L]. Each block contains the spatial integrals $\int_D\Sigma_{s,g'\to g}p_i p_j$ (**mass** matrix);

7) P is a time-dependent matrix of rank $N\times G$; it is not block-diagonal but can be full and each block represents the relation between the group of neutron inducing the fission and the group of the resulting fission neutrons. Each block contains the spatial integrals $\chi_g \int_D \left(1-\beta_{g'}\right)\nu\Sigma_{f,g'} p_i p_j$ (**mass** matrix). We can go further and decompose this matrix as:

(a) A block column matrix χ of size $\left[N\times G\right]\times N$ composed of the fission spectrum. Each block (size $N\times N$) is diagonal and contains the constant value χ_g. If it helps to see it as a matrix, then that's fine. To me, it may as well just be a **constant** scalar which is different for each energy group;

(b) A block line matrix $\tilde{\tilde{F}}$ of size $N\times\left[N\times G\right]$ composed of the total fission reaction rates weighted by the prompt neutrons fraction. Each block (size $N\times N$) is a **mass** matrix and contains $\tilde{\tilde{F}}^{g'} = \left(1-\beta_{g'}\right)\int_D \nu\Sigma_{f,g'} p_i p_j$. Since the total fission matrix is often used, we will denote it by: $F^{g'} = \int_D \nu\Sigma_{f,g'} p_i p_j$ (**mass** matrix);

(c) from these definitions, we obviously have : $\chi_{[N\times G]\times N} \times \tilde{\tilde{F}}_{N\times[N\times G]} = P_{[N\times G]\times[N\times G]}$

(d) Note that:

$$P^{gg'} = \chi_g \left(1-\beta_{g'}\right) F^{g'} \tag{29}$$

Note that:

$$D^{\ell,gg'} = \chi_{\ell,g}\beta_{\ell,g'} F^{g'} \tag{30}$$

8) C - The projected precursor concentrations are column **vectors**:

At steady state, we have

$$\lambda_\ell C_\ell = \sum_{g'=1}^{G} \beta_{\ell,g'}\nu\Sigma_{f,g'}\Phi^{g'} \tag{31}$$

hence,

$$\begin{aligned}\lambda_\ell c_{\ell,j}(t) &= \int_D \lambda_\ell C_\ell\left(\vec{r},t\right) p_j\left(\vec{r}\right) \\ &= \sum_{g'=1}^{G}\beta_{\ell,g'}\sum_i \int_D \nu\Sigma_{f,g'}\phi_i^{g'} p_i\left(\vec{r}\right) p_j\left(\vec{r}\right) \\ &= \sum_{g'=1}^{G}\beta_{\ell,g'} F^{g'}\phi^{g'}\end{aligned} \tag{32}$$

Note: This is how the initial value of precursors must be computed.

3. Time variation for the time-dependent matrices

Over the time interval $[t^n;t^{n+1}]$, we will assume a linear variation for $K(t)$:

$$K(t) = K^n u^n(t) + K^{n+1} u^{n+1}(t) \tag{33}$$

and for $F(t)$

$$F(t) = F^n u^n(t) + F^{n+1} u^{n+1}(t) \tag{34}$$

where

$$u^n(t) = \frac{t^{n+1} - t}{t^{n+1} - t^n} \tag{35}$$

$$u^{n+1}(t) = \frac{t - t^n}{t^{n+1} - t^n} \tag{36}$$

Integrating the balance equation over $[t^n;t^{n+1}]$ yields:

$$\begin{aligned} \frac{1}{V} M^{gg} \frac{d\phi^g}{dt} &= -\sum_{g'=1}^{G} K^{gg'}(t)\phi^{g'}(t) \\ &+ \sum_{\ell=1}^{L} \chi_{\ell,g} \lambda_\ell c_\ell^0 \exp(-\lambda_\ell t) \\ &+ \sum_{\ell=1}^{L} \chi_{\ell,g} \lambda_\ell \exp(-\lambda_\ell t) \sum_{g'=1}^{G} \sum_{s=0}^{s=t} \exp(\lambda_\ell s) \times \beta_{\ell,g'} F^{g'} \phi^{g'}(s)\, ds \end{aligned} \tag{37}$$

This can be rearranged as

$$\begin{aligned} \frac{1}{V} M^{gg} \left[\phi^{n+1}_g - \phi^n_g \right] &= -\sum_{g'=1}^{G} \int_{t^n}^{t^{n+1}} dt K^{gg'}(t)\phi^{g'}(t) \\ &+ \sum_{\ell=1}^{L} \chi_{\ell,g} \lambda_\ell \int_{t^n}^{t^{n+1}} dt c_\ell^0 \exp(-\lambda_\ell t) \\ &+ \sum_{\ell=1}^{L} \chi_{\ell,g} \lambda_\ell \int_{t^n}^{t^{n+1}} dt \left[\exp(-\lambda_\ell t) \sum_{g'=1}^{G} \sum_{s=0}^{s=t} ds \exp(\lambda_\ell s) \times \beta_{\ell,g'} F^{g'} \phi^{g'}(s) \right] \end{aligned} \tag{38}$$

4. Time variation for the flux

$$\phi(t) = w^n(t)\phi^n + w^{n+1}(t)\phi^{n+1} \tag{39}$$

The functions $w^n(t)$ and $w^{n+1}(t)$ over the time interval $[t^n;t^{n+1}]$ will be determined later.

Recalling that

$$K(t) = K^n u^n(t) + K^{n+1} u^{n+1}(t) \tag{40}$$

and using the following definitions,

$$a_{n,q} = \int_{t^n}^{t^{n+1}} u^n(t) w^q(t) dt \tag{41}$$

$$b_{n,q}^{\ell} = \int_{t^n}^{t^{n+1}} \exp\left(-\lambda_\ell \left(t^n - t\right)\right) \times u^n(t) w^q(t) dt \tag{42}$$

$$c_{n,q}^{\ell} = \int_{t^n}^{t^{n+1}} dt \int_{t^n}^{t} ds \exp\left(-\lambda_\ell \left(t - s\right)\right) \times u^n(s) w^q(s) \tag{43}$$

we obtain:

$$\int_{t^n}^{t^{n+1}} K(t)\phi(t) dt = \left[a_{n,n} K^n + a_{n+1,n} K^{n+1}\right]\phi^n + \left[a_{n,n+1} K^n + a_{n+1,n+1} K^{n+1}\right]\phi^{n+1} \tag{44}$$

and

$$\begin{aligned} &\chi_{\ell,g} \lambda_\ell \sum_{g'=1}^{G} \left\{ \beta_{\ell,g'} \int_{t^n}^{t^{n+1}} \left[\exp\left(-\lambda_\ell t\right)_{s=t^n}^{s=t} \exp\left(\lambda_\ell s\right) \times F^{g'}(s) \phi^{g'}(s) ds \right] dt \right\} \\ &= \chi_{\ell,g} \lambda_\ell \sum_{g'=1}^{G} \left\{ \beta_{\ell,g'} \times \begin{bmatrix} \left[c_{n,n}^{\ell} F^{g',n} + c_{n+1,n}^{\ell} F^{g',n+1}\right] \phi^{n}_{g'} \\ + \left[c_{n,n+1}^{\ell} F^{g',n} + c_{n+1,n+1}^{\ell} F^{g',n+1}\right] \phi^{n+1}_{g'} \end{bmatrix} \right\} \end{aligned} \tag{45}$$

In the above expression, the lower bound for the integration on s was t^n. We need to finish the integration starting from $s = 0$.

$$\begin{aligned} &\chi_{\ell,g} \lambda_\ell \int_{t^n}^{t^{n+1}} \exp\left(-\lambda_\ell t\right) \left[c_\ell^0 + \sum_{g'=1}^{G} \left\{ \beta_{\ell,g'} \int_0^{t^n} \exp\left(\lambda_\ell s\right) \times F^{g'}(s) \phi^{g'}(s) ds \right\} \right] dt \\ &= \chi_{\ell,g} \left[-\exp\left(-\lambda_\ell t^{n+1}\right) + \exp\left(-\lambda_\ell t^n\right) \right] \times \left[c_\ell^0 + \sum_{g'=1}^{G} \left\{ \beta_{\ell,g'} \int_0^{t^n} \exp\left(\lambda_\ell s\right) \times F^{g'}(s) \phi^{g'}(s) ds \right\} \right] \\ &= \chi_{\ell,g} \left[1 - \exp\left(-\lambda_\ell \Delta t\right)\right] c_\ell^n \end{aligned} \tag{46}$$

where:

$$c_\ell^n = \exp\left(-\lambda_\ell t^n\right) \left[c_\ell^0 + \sum_{g'=1}^{G} \left\{ \beta_{\ell,g'} \int_0^{t^n} \exp\left(\lambda_\ell s\right) \times F^{g'}(s) \phi^{g'}(s) ds \right\} \right] \tag{47}$$

The latter term can be computed by induction:

$$c_\ell^{n+1} = \exp\left(-\lambda_\ell t^{n+1}\right)\left[c_\ell^0 + \sum_{g'=1}^{G}\left\{\beta_{\ell,g'}\int_0^{t^{n+1}}\exp\left(\lambda_\ell s\right)\times F^{g'}(s)\phi^{g'}\left(s\right)ds\right\}\right]$$

$$= \exp\left(-\lambda_\ell \Delta t\right)\exp\left(-\lambda_\ell t^n\right)\left[\begin{array}{l} c_\ell^0 + \sum_{g'=1}^{G}\left\{\beta_{\ell,g'}\int_0^{t^n}\exp\left(\lambda_\ell s\right)\times F^{g'}(s)\phi^{g'}\left(s\right)ds\right\} \\ +\sum_{g'=1}^{G}\left\{\beta_{\ell,g'}\int_{t^n}^{t^{n+1}}\exp\left(\lambda_\ell s\right)\times F^{g'}(s)\phi^{g'}\left(s\right)ds\right\}\end{array}\right]$$

$$= \exp\left(-\lambda_\ell \Delta t\right)\left[c_\ell^n + \sum_{g'=1}^{G}\left\{\beta_{\ell,g'}\int_{t^n}^{t^{n+1}}\exp\left(\lambda_\ell\left(s-t^n\right)\right)\times F^{g'}(s)\phi^{g'}\left(s\right)ds\right\}\right] \quad (48)$$

Finally,

$$c_\ell^{n+1} = \exp\left(-\lambda_\ell \Delta t\right)\left[c_\ell^n + \sum_{g'=1}^{G}\left\{\beta_{\ell,g'}\int_{t^n}^{t^{n+1}}\exp\left(-\lambda_\ell\left(t^n - s\right)\right)\times F^{g'}(s)\phi^{g'}\left(s\right)ds\right\}\right]$$

$$= \exp\left(-\lambda_\ell \Delta t\right)\left[c_\ell^n + \sum_{g'=1}^{G}\beta_{\ell,g'}\left\{\begin{array}{l}\left[b_{n,n}^\ell F^{g',n} + b_{n+1,n}^\ell F^{g',n+1}\right]\phi_{g'}^{n} \\ +\left[b_{n,n+1}^\ell F^{g',n} + b_{n+1,n+1}^\ell F^{g',n+1}\right]\phi_{g'}^{n+1}\end{array}\right\}\right] \quad (49)$$

Putting everything together, the time integration equations

$$\frac{1}{V}M^{gg}\left[\phi^{n+1}_g - \phi^n_g\right] = -\sum_{g'=1}^{G}\int_{t^n}^{t^{n+1}} dtK^{gg'}(t)\phi^{g'}(t)$$

$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_\ell\int_{t^n}^{t^{n+1}} dtc_\ell^0\exp\left(-\lambda_\ell t\right) \quad (50)$$

$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_\ell\int_{t^n}^{t^{n+1}} dt\left[\exp\left(-\lambda_\ell t\right)\sum_{g'=1}^{G}\int_0^t ds\exp\left(\lambda_\ell s\right)\times\beta_{\ell,g'}F^{g'}\phi^{g'}\left(s\right)\right]$$

become

$$\frac{1}{V}M^{gg}\left[\phi^{n+1}_g - \phi^n_g\right] = -\sum_{g'=1}^{G}\left\{\begin{array}{l}\left[a_{n,n}K^{gg',n} + a_{n+1,n}K^{gg',n+1}\right]\phi^{g',n} \\ +\left[a_{n,n+1}K^{gg',n} + a_{n+1,n+1}K^{gg',n+1}\right]\phi^{g',n+1}\end{array}\right\}$$

$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_\ell\sum_{g'=1}^{G}\left\{\beta_{\ell,g'}\times\left[\begin{array}{l}\left[c_{n,n}^\ell F^{g',n} + c_{n+1,n}^\ell F^{g',n+1}\right]\phi_{g'}^{n} \\ +\left[c_{n,n+1}^\ell F^{g',n} + c_{n+1,n+1}^\ell F^{g',n+1}\right]\phi_{g'}^{n+1}\end{array}\right]\right\}$$

$$+\sum_{\ell=1}^{L}\chi_{\ell,g}\left[1-\exp\left(-\lambda_\ell\Delta t\right)\right]c_\ell^n \quad (51)$$

with c_ℓ^n the precursors concentration at the beginning of the time step. Once the new flux ϕ^{n+1}_g is solved for, we can get the precursors at the end of the time step by using

$$c_\ell^{n+1} = \exp\left(-\lambda_\ell \Delta t\right)\left[c_\ell^n + \sum_{g'=1}^{G} \beta_{\ell,g'} \left\{ \begin{array}{l} \left[b_{n,n}^\ell F^{g',n} + b_{n+1,n}^\ell F^{g',n+1}\right]\phi_{g'}^{n} \\ + \left[b_{n,n+1}^\ell F^{g',n} + b_{n+1,n+1}^\ell F^{g',n+1}\right]\phi_{g'}^{n+1} \end{array} \right\}\right]$$

In other words, it is of the form:

$$A^{n+1}\phi^{n+1} = A^n\phi^n + \sum_{\ell=1}^{L} \chi_{\ell,g}\left[1-\exp\left(-\lambda_\ell \Delta t\right)\right]c_\ell^n \tag{52}$$

where

$$\begin{aligned} A^{gg',n}\phi^{g',n} = {} & \frac{1}{V^g} M \phi^{g',n}\delta_{gg'} - a_{n,n}\sum_{g'=1}^{G}\left[L^{gg',n}\delta_{gg'} - T^{gg',n} - P^{gg',n}\right]\phi^{g',n} \\ & - a_{n+1,n}\sum_{g'=1}^{G}\left[L^{gg',n+1}\delta_{gg'} - T^{gg',n+1} - P^{gg',n+1}\right]\phi^{g',n} \\ & + \sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_\ell \sum_{g'=1}^{G}\beta_{\ell,g'}\left[c_{n,n}^\ell F^{g',n} + c_{n+1,n}^\ell F^{g',n+1}\right]\phi^{g',n} \end{aligned} \tag{53}$$

$$\begin{aligned} A^{gg',n+1}\phi^{g',n+1} = {} & \frac{1}{V^g} M \phi^{g',n+1}\delta_{gg'} + a_{n,n+1}\sum_{g'=1}^{G}\left[L^{gg',n}\delta_{gg'} - T^{gg',n} - P^{gg',n}\right]\phi^{g',n+1} \\ & + a_{n+1,n+1}\sum_{g'=1}^{G}\left[L^{gg',n+1}\delta_{gg'} - T^{gg',n+1} - P^{gg',n+1}\right]\phi^{g',n+1} \\ & - \sum_{\ell=1}^{L}\chi_{\ell,g}\lambda_\ell \sum_{g'=1}^{G}\beta_{\ell,g'}\left[c_{n,n+1}^\ell F^{g',n} + c_{n+1,n+1}^\ell F^{g',n+1}\right]\phi^{g',n+1} \end{aligned} \tag{54}$$

If we let

$$S^{g,n} = \sum_{\ell=1}^{L}\chi_{\ell,g}\left[1-\exp\left(-\lambda_\ell \Delta t\right)\right]c_\ell^n \tag{55}$$

we have to solve the multi-group fixed source problem:

$$A^{n+1}\phi^{n+1} = A^n\phi^n + S^n \tag{56}$$

Once ϕ^{n+1} is determined, we can then compute c_ℓ^{n+1} by induction.

5. Choice of time functions

The time-weight functions must satisfy:

$$\int_{t^n}^{t^{n+1}} w^n(t)dt = (1-\theta)\Delta t \tag{57}$$

$$\int_{t^n}^{t^{n+1}} w^{n+1}(t)dt = \theta\Delta t \tag{58}$$

In order to get the Crank-Nicholson scheme, it is obvious to let

$$w^n(t) = u^n(t) \tag{59}$$

$$w^{n+1}(t) = u^{n+1}(t) \tag{60}$$

i.e., $$\phi(t) = \frac{t^{n+1}-t}{t^{n+1}-t^n}\phi^n + \frac{t-t^n}{t^{n+1}-t^n}\phi^{n+1} \tag{61}$$

(A linear approximation between the beginning and end times)

But, for any other value of θ, such a choice would lead to a discontinuous approximation in time. We have to take a more general approach to define the weight function in time. This is explained below.

The conditions to be satisfied by $w^n(t)$ and $w^{n+1}(t)$ are:

$$\begin{cases} w^n(t_n) = 1 \\ w^n(t_{n+1}) = 0 \\ \int_{t^n}^{t^{n+1}} w^n(t)dt = (1-\theta)\Delta t \end{cases} \quad \text{and} \quad \begin{cases} w^{n+1}(t_n) = 0 \\ w^{n+1}(t_{n+1}) = 1 \\ \int_{t^n}^{t^{n+1}} w^{n+1}(t)dt = \theta\Delta t \end{cases} \tag{62}$$

leading to the general form for the time functions:

$$w^n(t) = \left(1-u^{n+1}(t)\right)\times\left[1-3u^{n+1}(t)(2\theta-1)\right] = u^n(t)\times\left[1-3u^{n+1}(t)(2\theta-1)\right] \tag{63}$$

$$w^{n+1}(t) = u^{n+1}(t)\times\left[2(3\theta-1)+3u^{n+1}(t)(1-2\theta)\right] \tag{64}$$

We find

- for $\theta = 1/2$

$$w^n(t) = u^n(t) \tag{65}$$

$$w^{n+1}(t) = u^{n+1}(t) \tag{66}$$

- and for $\theta = 1$

$$w^n(t) = \left(1-u^{n+1}(t)\right)\times\left[1-3u^{n+1}(t)\right] \tag{67}$$

$$w^{n+1}(t) = u^{n+1}(t)\times\left[4-3u^{n+1}(t)\right] \tag{68}$$

6. Computation of the various coefficients

$$a_{n,q} = \int_{t^n}^{t^{n+1}} u^n(t) w^q(t) dt \tag{69}$$

$$b_{n,q}^{\ell} = \int_{t^n}^{t^{n+1}} \exp\left(-\lambda_{\ell}\left(t^n - t\right)\right) \times u^n(t) w^q(t) dt \tag{70}$$

$$c_{n,q}^{\ell} = \int_{t^n}^{t^{n+1}} dt \int_{t^n}^{t} ds \exp\left(-\lambda_{\ell}\left(t - s\right)\right) \times u^n(s) w^q(s) \tag{71}$$

where the linear variation over a time step of the properties is obtained using the functions

$$u^n(t) = \frac{t^{n+1} - t}{t^{n+1} - t^n} \tag{72}$$

$$u^{n+1}(t) = \frac{t - t^n}{t^{n+1} - t^n} \tag{73}$$

The flux variation over the time step has been approximated by

$$\phi(t) = w^n(t)\phi^n + w^{n+1}(t)\phi^{n+1} \tag{74}$$

with

$$w^n(t) = \left(1 - u^{n+1}(t)\right) \times \left[1 - 3u^{n+1}(t)\left(2\theta - 1\right)\right] \tag{75}$$

$$w^{n+1}(t) = u^{n+1}(t) \times \left[2\left(3\theta - 1\right) + 3u^{n+1}(t)\left(1 - 2\theta\right)\right] \tag{76}$$

6.1. $a_{n,q}$

$$a_{n,n} = \Delta t \frac{1 + 6\left(1 - \theta\right)}{12} = \Delta t \frac{7 - 6\theta}{12} \tag{77}$$

$$a_{n,n+1} = \Delta t \frac{6\theta - 1}{12} \tag{78}$$

$$a_{n+1,n+1} = \Delta t \frac{1 + 6\theta}{12} \tag{79}$$

$$a_{n+1,n} = \Delta t \frac{6(1 - \theta) - 1}{12} = \Delta t \frac{5 - 6\theta}{12} \tag{80}$$

Note that:

- $a_{n,n} + a_{n,n+1} = \Delta t/2$ and $a_{n+1,n} + a_{n+1,n+1} = \Delta t/2$
- $a_{n,n} + a_{n+1,n} = \Delta t(1 - \theta)$ and $a_{n,n+1} + a_{n+1,n+1} = \Delta t\theta$

6.2. $b_{n,q}$

By definition,

$$b_{n,q}^{\ell} = \int_{t^n}^{t^{n+1}} \exp\left(-\lambda_{\ell}\left(t^n - t\right)\right) \times u^n(t) w^q(t) dt \tag{81}$$

The various forms of $u^n(t)w^q(t)$ are 3-rd order polynomials in time.

$$b_{n,q}^{\ell} = \int_{t^n}^{t^{n+1}} \exp\left(-\lambda_{\ell}\left(t^n - t\right)\right) \times P_3(t) dt \tag{82}$$

Letting

$$x = \frac{t - t_n}{\Delta t} \tag{83}$$

$$b_{n,q}^{\ell} = \Delta t \int_0^1 \exp\left(\lambda_{\ell} \Delta t x\right) \times P_3(\Delta t x + t_n) dx \tag{84}$$

We now need to find the integral of

$$I_m = \int_0^1 x^m \exp\left(\gamma x\right) dx \text{ with } \gamma = \lambda \Delta t \tag{85}$$

By parts, we get:

$$I_m = \frac{1}{\gamma}\left[\exp\gamma - m I_{m-1}\right] \text{ for } m \geq 1 \tag{86}$$

Then, the coefficients we are looking for are

$$I_0 = \frac{-1 + \exp(\gamma)}{\gamma} \tag{87}$$

$$I_1 = \frac{1 + \exp(\gamma)\left[-1 + \gamma\right]}{\gamma^2} \tag{88}$$

$$I_2 = \frac{-2 + \exp(\gamma)\left[2 - 2\gamma + \gamma^2\right]}{\gamma^3} \tag{89}$$

$$I_3 = \frac{6 + \exp(\gamma)\left[-6 + 6\gamma - 3\gamma^2 + \gamma^3\right]}{\gamma^4} \tag{90}$$

We now need to write the polynomials $P_3(t) = P_3(\Delta t x + t_n)$. Their coefficients in power of x^m will be put in front of the previous integrals I_m to form $b_{n,q}^{\ell}$.

$$u^n(t)w^n(t) = 1 + (1 - 6\theta)x + (12\theta - 5)x^2 + (3 - 6\theta)x^3 \tag{91}$$

$$u^n(t)w^{n+1}(t) = (6\theta - 2)x + (5 - 12\theta)x^2 + (6\theta - 3)x^3 \tag{92}$$

$$u^{n+1}(t)w^n(t) = x + (2 - 6\theta)x^2 + (6\theta - 3)x^3 \tag{93}$$

$$u^{n+1}(t)w^{n+1}(t) = (-2 + 6\theta)x^2 + (-6\theta + 3)x^3 \tag{94}$$

Which then yields

$$b_{n,n} = \Delta t\left[I_0 + \left[1-6\theta\right] I_1 + (12\theta - 5) I_2 + \left(3-6\theta\right) I_3 \right] \tag{95}$$

$$b_{n,n+1} = \Delta t\left[(6\theta - 2) I_1 + (5-12\theta) I_2 + (6\theta - 3) I_3 \right] \tag{96}$$

$$b_{n+1,n} = \Delta t\left[I_1 + \left(2-6\theta\right) I_2 + (6\theta - 3) I_3 \right] \tag{97}$$

$$b_{n+1,n+1} = \Delta t\left[\left(6\theta - 2\right) I_2 + \left(3-6\theta\right) I_3 \right] \tag{98}$$

6.3. $c_{n,q}$

By definition,

$$c_{n,q}^{\ell} = \int_{t^n}^{t^{n+1}} dt \int_{t^n}^{t} ds \exp\left(-\lambda_{\ell}\left(t-s\right)\right) \times u^n(s) w^q(s) \tag{99}$$

The various forms of $u^n(t)w^q(t)$ are 3-rd order polynomials in time.

$$c_{n,q}^{\ell} = \int_{t^n}^{t^{n+1}} dt \int_{t^n}^{t} ds \exp\left(-\lambda_{\ell}\left(t-s\right)\right) \times P_3(s) \tag{100}$$

Letting

$$x = \frac{s - t_n}{\Delta t} \tag{101}$$

$$\begin{aligned} c_{n,q}^{\ell} &= \int_{t^n}^{t^{n+1}} dt \int_{t^n}^{t} ds \exp\left(-\lambda_{\ell}\left(t-s\right)\right) \times P_3(s) \\ &= \Delta t \int_{t^n}^{t^{n+1}} dt \int_{0}^{\frac{t-t_n}{\Delta t}} dx \exp\left(\lambda_{\ell} \Delta t \left(x + \frac{t_n - t}{\Delta t} \right) \right) \times P_3(x\Delta t + t_n) \end{aligned} \tag{102}$$

Letting

$$y = \frac{t - t_n}{\Delta t} \tag{103}$$

$$\begin{aligned} c_{n,q}^{\ell} &= \Delta t \int_{t^n}^{t^{n+1}} dt \int_{0}^{\frac{t-t_n}{\Delta t}} dx \exp\left(\lambda_{\ell} \Delta t \left(x + \frac{t_n - t}{\Delta t} \right) \right) \times P_3(x\Delta t + t_n) \\ &= \Delta t^2 \int_0^1 dy \int_0^y dx \exp\left(\lambda_{\ell} \Delta t \left(x - y\right)\right) \times P_3(x\Delta t + t_n) \end{aligned} \tag{104}$$

We now need to find the integral of

$$J_m = \int_0^1 dy \int_0^y x^m \exp\left(-\gamma(y-x)\right) dx \text{ with } \gamma = \lambda \Delta t \tag{105}$$

By induction, we get:

$$J_m = \frac{1}{\gamma}\left[\frac{1}{m+1} - mJ_{m-1}\right] \text{ for } m \geq 1 \tag{106}$$

or,

$$J_0 = \frac{-1+\gamma+\exp(-\gamma)}{\gamma^2} \tag{107}$$

$$J_1 = \frac{2-2\exp(-\gamma)-2\gamma+\gamma^2}{2\gamma^3} \tag{108}$$

$$J_2 = -\frac{6-6\exp(-\gamma)-6\gamma+3\gamma^2-\gamma^3}{3\gamma^4} \tag{109}$$

$$J_3 = \frac{24-24\exp(-\gamma)+\gamma\left(-24+\gamma\left(12+\left(\gamma-4\right)\gamma\right)\right)}{4\gamma^5} \tag{110}$$

Which then yields

$$c_{n,n} = \Delta t^2\left[J_0 + \left[1-6\theta\right]J_1 + (12\theta-5)J_2 + \left(3-6\theta\right)J_3\right] \tag{111}$$

$$c_{n,n+1} = \Delta t^2\left[(6\theta-2)J_1 + (5-12\theta)J_2 + (6\theta-3)J_3\right] \tag{112}$$

$$c_{n+1,n} = \Delta t^2\left[J_1 + \left(2-6\theta\right)J_2 + (6\theta-3)J_3\right] \tag{113}$$

$$c_{n+1,n+1} = \Delta t^2\left[\left(6\theta-2\right)J_2 + \left(3-6\theta\right)J_3\right] \tag{114}$$

APPENDIX B

Sample table for 2 group material cross-section data

```
*
*  NEM-Cross Section Table Input
*
*    T Fuel        Rho Mod.      Boron ppm.    T Mod.
         5                6                0           0
*******           X-Section set #        1
  1
*
*      Group No.  1
*
***************            Diffusion Coefficient Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .1508746E+01  .1511112E+01  .1512206E+01  .1512700E+01
  .1518247E+01  .1442037E+01  .1444304E+01  .1445194E+01  .1445690E+01
  .1450740E+01  .1392654E+01  .1394824E+01  .1395713E+01  .1396108E+01
  .1400958E+01  .1391094E+01  .1393265E+01  .1394155E+01  .1394650E+01
  .1399403E+01  .1384241E+01  .1386413E+01  .1387202E+01  .1387698E+01
  .1392449E+01  .1360958E+01  .1363033E+01  .1363924E+01  .1364319E+01
  .1368548E+01
*
***************       Total Absorption X-Section Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .1058338E-01  .1073651E-01  .1079159E-01  .1081860E-01
  .1106677E-01  .1077265E-01  .1092982E-01  .1098688E-01  .1101492E-01
  .1127017E-01  .1090575E-01  .1106592E-01  .1112498E-01  .1115302E-01
  .1141525E-01  .1091032E-01  .1107151E-01  .1112957E-01  .1115860E-01
  .1142085E-01  .1092912E-01  .1109030E-01  .1114936E-01  .1117740E-01
  .1144162E-01  .1099822E-01  .1116148E-01  .1122057E-01  .1124962E-01
  .1151338E-01
*
***************            Fission X-Section Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .1983180E-02  .1975335E-02  .1972097E-02  .1970381E-02
  .1953093E-02  .2013847E-02  .2005950E-02  .2002737E-02  .2001020E-02
  .1983736E-02  .2036499E-02  .2028554E-02  .2025408E-02  .2023688E-02
  .2006335E-02  .2037614E-02  .2029706E-02  .2026487E-02  .2024764E-02
  .2007453E-02  .2040996E-02  .2033049E-02  .2029828E-02  .2028183E-02
  .2010788E-02  .2053112E-02  .2045272E-02  .2042010E-02  .2040287E-02
  .2022316E-02
*
***************            Nu-Fission X-Section Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .5356767E-02  .5336368E-02  .5328014E-02  .5323575E-02
  .5278625E-02  .5433965E-02  .5413659E-02  .5405386E-02  .5400952E-02
  .5356086E-02  .5489179E-02  .5468777E-02  .5460499E-02  .5456064E-02
  .5411287E-02  .5491165E-02  .5470871E-02  .5462597E-02  .5458158E-02
  .5413298E-02  .5499056E-02  .5478661E-02  .5470388E-02  .5465954E-02
  .5421083E-02  .5527800E-02  .5507508E-02  .5499132E-02  .5494696E-02
  .5448120E-02
```

```
*
***************           Scattering X-Section Table
*
****   From group 1 to 2
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .1339091E-01  .1330232E-01  .1326915E-01  .1325304E-01
  .1310006E-01  .1515151E-01  .1505683E-01  .1502159E-01  .1500448E-01
  .1484336E-01  .1655319E-01  .1645547E-01  .1641921E-01  .1640109E-01
  .1623286E-01  .1659264E-01  .1649494E-01  .1645869E-01  .1644056E-01
  .1627138E-01  .1679482E-01  .1669712E-01  .1665986E-01  .1664175E-01
  .1647253E-01  .1752311E-01  .1742239E-01  .1738511E-01  .1736698E-01
  .1718833E-01
*
*      Group No.  2
*
***************           Diffusion Coefficient Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .3934076E+00  .3940181E+00  .3942747E+00  .3944050E+00
  .3959027E+00  .3597593E+00  .3603187E+00  .3605462E+00  .3606666E+00
  .3619079E+00  .3355342E+00  .3360643E+00  .3362854E+00  .3363927E+00
  .3375699E+00  .3344979E+00  .3350412E+00  .3352591E+00  .3353698E+00
  .3365476E+00  .3310488E+00  .3315824E+00  .3318037E+00  .3319111E+00
  .3330892E+00  .3195803E+00  .3201075E+00  .3203192E+00  .3204300E+00
  .3214827E+00
*
***************       Total Absorption X-Section Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .9577228E-01  .9558064E-01  .9550184E-01  .9546290E-01
  .9508551E-01  .9695843E-01  .9675865E-01  .9667581E-01  .9663488E-01
  .9619936E-01  .9805718E-01  .9784644E-01  .9776057E-01  .9771180E-01
  .9725226E-01  .9822100E-01  .9801108E-01  .9792421E-01  .9787637E-01
  .9741169E-01  .9844869E-01  .9822893E-01  .9814298E-01  .9809416E-01
  .9762637E-01  .9923134E-01  .9900244E-01  .9891636E-01  .9886842E-01
  .9834462E-01
*
***************           Fission X-Section Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .5002976E-01  .4993335E-01  .4989055E-01  .4987119E-01
  .4967748E-01  .5036214E-01  .5025733E-01  .5021079E-01  .5019138E-01
  .4996737E-01  .5067940E-01  .5056706E-01  .5052276E-01  .5049917E-01
  .5026231E-01  .5075058E-01  .5063864E-01  .5059193E-01  .5056883E-01
  .5032640E-01  .5082717E-01  .5071153E-01  .5066529E-01  .5064354E-01
  .5039907E-01  .5109075E-01  .5096993E-01  .5092356E-01  .5089675E-01
  .5062727E-01
*
***************           Nu-Fission X-Section Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .1351354E+00  .1348949E+00  .1347893E+00  .1347420E+00
```

```
  .1342633E+00  .1358922E+00  .1356345E+00  .1355189E+00  .1354716E+00
  .1349119E+00  .1366012E+00  .1363237E+00  .1362094E+00  .1361508E+00
  .1355625E+00  .1367677E+00  .1364914E+00  .1363756E+00  .1363184E+00
  .1357102E+00  .1369436E+00  .1366574E+00  .1365429E+00  .1364844E+00
  .1358759E+00  .1375567E+00  .1372518E+00  .1371372E+00  .1370700E+00
  .1363899E+00
*
***************           Xe X-Section Table
*
  .5000000E+03  .7602200E+03  .8672700E+03  .9218800E+03  .1500000E+04
  .6413994E+03  .7114275E+03  .7694675E+03  .7724436E+03  .7813064E+03
  .8100986E+03  .3239827E-02  .3230555E-02  .3226600E-02  .3224590E-02
  .3204245E-02  .3240648E-02  .3231327E-02  .3227365E-02  .3225351E-02
  .3203756E-02  .3252643E-02  .3243114E-02  .3239178E-02  .3237033E-02
  .3215146E-02  .3259883E-02  .3250283E-02  .3246216E-02  .3244198E-02
  .3222104E-02  .3264798E-02  .3255194E-02  .3251157E-02  .3249009E-02
  .3226905E-02  .3280296E-02  .3270513E-02  .3266372E-02  .3264252E-02
  .3240836E-02
*
***************           Inv. Neutron Velocities
*
  .5456853E-07  .2491910E-05
*
```

APPENDIX C

A sample Input.xml file used for 1-D calculations

```
<?xml version="1.0" encoding="utf-8"?>
<InputDeck>
    <BaseParams ReactorType="PWR">
        <Dimensions>
            <Height>4</Height>
            <Radius_Fuel>4.6955E-3</Radius_Fuel>
            <Radius_Gap>4.791E-3</Radius_Gap>
            <Radius_Clad>5.464E-3</Radius_Clad>
            <Radius_WH>6.325E-3</Radius_WH>
            <Assembly_Pitch>0.21504</Assembly_Pitch>
            <NumHoles>225</NumHoles>
            <NumWHoles>17</NumWHoles>
            <NumAssemblies>177</NumAssemblies>
        </Dimensions>
        <Operation>
            <Pressure>155.5E5</Pressure>
            <Coolant_Inlet_Temp>291</Coolant_Inlet_Temp>
            <Coolant_Velocity>5.03</Coolant_Velocity>
            <InitialNPower>1.0</InitialNPower>
            <BaseNPower>2772E6</BaseNPower>
        </Operation>
    </BaseParams>
    <Physics>
        <HeatConduction Tf_Init="650" TFuel_Max="2500">
            <HGap>11356</HGap>
            <!-TfAvg = Tf_Param*Tf,center + (1-Tf_Param)*Tf,surface-->
                  <Teff_Param>0.3</Teff_Param>
                  <RhoFuel>10412.0</RhoFuel>
                  <RhoClad>6600.0</RhoClad>
                  <Discretization>
                        <NFuel>7</NFuel>
                        <NClad>0</NClad>
                  </Discretization>
                  <RunParams>
                        <OuterIter>500</OuterIter>
                        <Eps>1E-3</Eps>
                  </RunParams>
        </HeatConduction>
        <Neutronics>
            <!-- For 0-D Calculations -->
            <PRKE>
                  <GenerationTime>1E-4</GenerationTime>
                  <Feedback>
                        <AlphaD>-150E-5</AlphaD>
                        <AlphaM>-7E-5</AlphaM>
                  </Feedback>
            </PRKE>
            <OneDimension FluxInit="flat">
                  <!-- Lattice arrangement -->
                  <LatticeGeometry>1 2 1</LatticeGeometry>
                  <!-- Energy Groups -->
                  <NGroups>2</NGroups>
                  <!-- Last Fast Group -->
                  <LFG>0</LFG>
                  <!-- Cross-section tables -->
                  <XSData External="true">
                        <XSUnroddedFile>MSLB_XS </XSUnroddedFile>
```

```
                        <XSRoddedFile>MSLB_XSR </XSRoddedFile>
                    </XSData>
                    <Delayed Groups="1">
                        <DGroup dgid="1" beta="0.00471" lambda="0.1" />
                    </Delayed>
                    <Discretization>
                        <!-- Physical refinement -->
                        <Refinement>50 40 50</Refinement>
                        <Order>1</Order>
                    </Discretization>
                    <RunParams>
                        <OuterIter>1000000</OuterIter>
                        <ThermalIter>500</ThermalIter>
                        <Eps_K>1E-12</Eps_K>
                        <Eps_S>1E-10</Eps_S>
                        <Eps_Thermal>1E-8</Eps_Thermal>
                        <SORParam_Thermal>1.0</SORParam_Thermal>
                        <Thermal_Rebalance>false</Thermal_Rebalance>
                    </RunParams>
                </OneDimension>
            </Neutronics>
            <ThermalHydraulics TmodInit="310"></ThermalHydraulics>
        </Physics>
        <RunParams>
            <IsAdaptive>false</IsAdaptive>
            <IncludeFeedback>true</IncludeFeedback>
            <IsCriticalStart>true</IsCriticalStart>
            <IsPredictive>true</IsPredictive>
            <OnlyTH>false</OnlyTH>
            <IncludeModFB>true</IncludeModFB>
            <Staggered>true</Staggered>
            <NeutronicsFirst>true</NeutronicsFirst>
            <Theta>0.5</Theta>
            <DelT>1E-3</DelT>
            <Total_Duration>1</Total_Duration>
            <OuterIter>1</OuterIter>
            <Eps>1E-3</Eps>
            <Output_Frequency>2</Output_Frequency>
            <!-- Format for printing output -->
            <OutputFileFormat>2Group.out</OutputFileFormat>
            <Transient>
                <RodHeightInitial>50</RodHeightInitial>
                <!--Format : Type, Start time, End time, Amplitude (pcm)-->
                <Reactivities>
                    <Reactivity type="Step" start="0" amp="-50" />
                    <Reactivity type="Ramp" start="0" end="0" amp="10" />
                </Reactivities>
            </Transient>
        </RunParams>
    </InputDeck>
```

VITA

Vijay Subramaniam Mahadevan graduated from the Regional Engineering College, affiliated with Bharathidasan University, Trichy, India in May 2002 with a Bachelor of Technology degree in chemical engineering. He entered the nuclear engineering graduate program at Texas A&M University in August 2004. Future contact may be made by e-mail at vijaysm@tamu.edu or by mail forwarded through the department of Nuclear Engineering, c/o Dr. Jean C. Ragusa, Texas A&M University, College Station, TX 77843-3133