Development and Analysis of High Order Neutron Transport–Depletion Coupling Algorithms

by

Colin Josey

B.S., University of New Mexico (2013)
S.M., Massachusetts Institute of Technology (2015)

Submitted to the Department of Nuclear Science and Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Nuclear Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Author .....
Department of Nuclear Science and Engineering
September 1, 2017

Certified by .....
Benoit Forget
Associate Professor of Nuclear Science and Engineering
Thesis Supervisor

Certified by .....
Kord Smith
KEPCO Professor of the Practice of Nuclear Science and Engineering
Thesis Reader

Accepted by .....
Ju Li
Battelle Energy Alliance Professor of Nuclear Science and Engineering
Chairman, Department Committee on Graduate Theses

Development and Analysis of High Order Neutron Transport–Depletion Coupling Algorithms

by

Colin Josey

Submitted to the Department of Nuclear Science and Engineering
on September 1, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Nuclear Science and Engineering

Abstract

The coupling of depletion and neutron transport together creates a particularly challenging mathematical problem. Due to the stiffness of the ODE, special algorithms needed to be developed to minimize the number of transport simulations required. In addition, for stochastic transport, both the time step and the number of particles per time step need to be considered. In recent years, many new coupling algorithms have been developed. However, relatively little analysis of the numeric and stochastic convergence of these techniques has been performed.

In this document, several new algorithms are introduced. Some are improvements of current techniques, some are taken from similar problems in other fields, and some are derived from scratch for this specific problem. These were then tested on several test problems to investigate their convergence. With regard to numerical error, the CF4 algorithm (a commutator-free Lie integrator) outperformed all tested algorithms. In number of transport simulations to achieve a 0.1% gadolinium relative error, CF4 requires half the simulations. With regard to stochastic error, it was found that once a time step is sufficiently reduced, errors are mostly a function of the number of particles used during the simulation. The remaining variability is due to how stochastic noise propagates through each numerical integrator. Using this information, a technique is developed to minimize the cost of running a depletion simulation.

Thesis Supervisor: Benoit Forget
Title: Associate Professor of Nuclear Science and Engineering

Acknowledgments

I would like to thank all of those who have taught me so much over the course of this project. Specifically, I would like to thank my advisers, Ben Forget and Kord Smith. Without them to bounce ideas off of, many aspects of this project would not exist. In addition, I would like to thank my family for helping me through this project.

This research was funded in part by the Consortium for Advanced Simulation of Light Water Reactors (CASL) under U.S. Department of Energy Contract No. DE-AC05-00OR22725; DOE's Center for Exascale Simulation of Advanced Reactors (CESAR) under U.S. Department of Energy Contract No. DE-AC02-06CH11357; and by a Department of Energy Nuclear Energy University Programs Graduate Fellowship. This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy under Contract No. DE-AC07-05ID14517.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Depletion, the transmutation and decay of nuclides within materials under irradiation in a nuclear reactor, is one of the most challenging problems in the field of nuclear engineering. The ODEs that govern the process present a number of challenges. Primarily, the ODEs are extraordinarily stiff. Secondly, the coefficients of the ODE are computed from a neutron transport simulation. The combination of these two issues mean that very special algorithms are required to perform integration efficiently. Further complications arise if the transport is performed via Monte Carlo, as now the solution to the ODEs have a statistical component.

Unfortunately, relatively little analysis on how these special algorithms work has been performed, especially with regard to uncertain coefficients. The result is that many simulations are performed inefficiently. This takes an already computationally intense problem and increases its cost further. The main goal of this document is to reduce the computational cost of depletion, which is done via a thorough analysis of the algorithms. In addition, several new algorithms are proposed for special use cases. An overview of these goals and the entire project is presented in Section 1.3.

This chapter introduces the governing equations of the depletion process, as well as a brief introduction to why these equations are so challenging to solve. In addition, several implementation challenges are briefly introduced, which will be used to guide the design of new algorithms. In Chapter 2, the current state-of-the-art depletion methods, as well as some advanced techniques from other fields, are reviewed. This is followed by Chapter 3, in which several improvements are introduced as well as some new techniques such as an adaptive time step scheme. Chapter 4 formulates a simple test problem to rapidly test all of these algorithms. From the results of that chapter, a smaller selection of algorithms are then tested on two real depletion problems in Chapter 5. Finally, Chapter 6 summarizes the results, makes algorithm recommendations, and estimates the costs required for full simulations.

1.1 Fundamental Equation

The equation that governs the transmutation and decay of nuclides inside of an irradiated environment is the Bateman equation [48]. Nuclides can be created and destroyed via either decay or transmutation. When these processes are combined, the result is Equation (1.1).

$$\begin{aligned}\frac{dN_i(\mathbf{r},t)}{dt} = &\sum_j \left[\int d\mathbf{\Omega} \int_0^\infty dE \sigma_{j\to i}(\mathbf{r},\mathbf{\Omega},E,t)\phi(\mathbf{r},\mathbf{\Omega},E,t) + \lambda_{j\to i}\right] N_j(\mathbf{r},t) \\ &- \sum_j \left[\int d\mathbf{\Omega} \int_0^\infty dE \sigma_{i\to j}(\mathbf{r},\mathbf{\Omega},E,t)\phi(\mathbf{r},\mathbf{\Omega},E,t) + \lambda_{i\to j}\right] N_i(\mathbf{r},t)\end{aligned} \tag{1.1}$$

Where:

$$\begin{aligned} N_i(\mathbf{r},t) &= \text{nuclide } i \text{ density at position } \mathbf{r} \text{ and time } t \\ \sigma_{i\to j}(\mathbf{r},\mathbf{\Omega},E,t) &= \text{microscopic cross section of a reaction where nuclide } i \\ &\quad \text{generates nuclide } j \text{ at position } \mathbf{r}, \text{ angle } \mathbf{\Omega}, \text{ energy } E \text{ and time } t \\ \phi(\mathbf{r},\mathbf{\Omega},E,t) &= \text{neutron flux at position } \mathbf{r}, \text{ angle } \mathbf{\Omega}, \text{ energy } E \text{ and time } t \\ \lambda_{i\to j} &= \text{decay constant of nuclide } i \text{ to nuclide } j \end{aligned}$$

As a matter of convenience, the rest of this document assumes that the reactor is discretized into regions in which the cross sections and nuclide densities are constant. This is not strictly necessary; instead one could approximate spatially varying terms with polynomials and integrate the coefficients [18], but the former is the most common approach. This modification yields Equation (1.2), in which an integral over volume has been performed. $N_i^{(c)}(t)$ stands for the atom density of nuclide i in cell c.

$$\begin{aligned}\frac{dN_i^{(c)}(t)}{dt} = &\sum_j \left[\frac{1}{V_c}\int_{\text{cell}} d\mathbf{r} \int d\mathbf{\Omega} \int_0^\infty dE \sigma_{j\to i}(\mathbf{r},\mathbf{\Omega},E,t)\phi(\mathbf{r},\mathbf{\Omega},E,t) + \lambda_{j\to i}\right] N_j^{(c)}(t) \\ &- \sum_j \left[\frac{1}{V_c}\int_{\text{cell}} d\mathbf{r} \int d\mathbf{\Omega} \int_0^\infty dE \sigma_{i\to j}(\mathbf{r},\mathbf{\Omega},E,t)\phi(\mathbf{r},\mathbf{\Omega},E,t) + \lambda_{i\to j}\right] N_i^{(c)}(t)\end{aligned} \tag{1.2}$$

The next consideration is where to get components of the above equation to solve it. The decay constants are the easiest, as they are available from databases such as ENDF/B-VII.1 [14]. The integrals however, require a space-angle-energy neutron transport solution. When a neutron transport solver is given N_i at time t, a typical reaction rate tally will yield the quantity $\hat{R}$ given

in Equation (1.3) [42].

$$\begin{aligned}\hat{R}^{(c)}_{i\to j}(t) &= \int_{\text{cell}} d\mathbf{r} \int d\mathbf{\Omega} \int_0^\infty dE \sigma_{i\to j}(\mathbf{r},\mathbf{\Omega},E,t) N_i(\mathbf{r},t) \hat{\phi}(\mathbf{r},\mathbf{\Omega},E,t) \\ \hat{R}^{(c)}_{i\to j}(t) &= N^{(c)}_i(t) \int_{\text{cell}} d\mathbf{r} \int d\mathbf{\Omega} \int_0^\infty dE \sigma_{i\to j}(\mathbf{r},\mathbf{\Omega},E,t) \hat{\phi}(\mathbf{r},\mathbf{\Omega},E,t) \end{aligned} \tag{1.3}$$

This is almost but not quite the integral shown in Equation (1.2), the difference being the $\hat{\phi}$, which represents the unnormalized flux and not the true flux. To get the ratio between the true flux and the unnormalized flux, power normalization must be performed. The power given an unnormalized flux is given by Equation (1.4). P_{other} includes energy deposition from scattering and other similar events, as computed using the unit flux. $Q^{(r)}$ and $Q^{(d)}$ are the *recoverable* quantities of energy emitted given a reaction or decay respectively.

$$\begin{aligned} P_{\text{unit}}(t) = &\sum_{i,j} \int d\mathbf{r} \int d\mathbf{\Omega} \int_0^\infty dE \sigma_{i\to j}(\mathbf{r},\mathbf{\Omega},E,t) N_i(\mathbf{r},t) \hat{\phi}(\mathbf{r},\mathbf{\Omega},E,t_s) Q^{(r)}_{i\to j}(\mathbf{r},\mathbf{\Omega},E) \\ &+ \sum_{c,i,j} V_c N^{(c)}_i(t) \lambda_{i\to j} Q^{(d)}_{i\to j} + P_{\text{other}}(t) \end{aligned} \tag{1.4}$$

The average of $Q^{(r)}$ is often used to further simplify this calculation, as then the first integral just becomes the sum of $\hat{R}^{(c)}_{i\to j} Q^{(r)}_{i\to j}$ for all c, i, and j. A more thorough discussion of power computation is presented in Section 1.2.1. Once P_{unit} is known, then R is given by Equation (1.5).

$$R^{(c)}_{i\to j}(t) = \frac{P(t)}{P_{\text{unit}}(t)} \hat{R}^{(c)}_{i\to j}(t) \tag{1.5}$$

This simplifies Equation (1.2) further to the form given in Equation (1.6).

$$\begin{aligned} \frac{dN^{(c)}_i(t)}{dt} = &\sum_j \left[\frac{R^{(c)}_{j\to i}(t)}{V_c N^{(c)}_j(t)} + \lambda_{j\to i} \right] N^{(c)}_j(t) \\ &- \sum_j \left[\frac{R^{(c)}_{i\to j}(t)}{V_c N^{(c)}_i(t)} + \lambda_{i\to j} \right] N^{(c)}_i(t) \end{aligned} \tag{1.6}$$

One might note that $N^{(c)}(t)$ is in both the denominator and numerator in parts of the equation. While it is tempting to cancel it out, one notes by comparison to Equation (1.3) that $N^{(c)}(t)$ is already cancelled out in the fraction $R^{(c)}(t)/N^{(c)}(t)$. Care must be taken to ensure that this value is correctly computed even when $N_i = 0$, by directly computing the integral in (1.3) or by adding near-infinitely dilute quantities of nuclides. This form can be trans-

formed one final time into the form of Equation (1.7).

$$y'^{(c)}(t) = A^{(c)}(y,t)y^{(c)}(t) \tag{1.7}$$

$$y_i^{(c)}(t) = V_c N_i^{(c)}(t)$$

$$A_{i,j}^{(c)}(y,t) = \frac{R_{i\to j}^{(c)}(t)}{y_i^{(c)}(t)} + \lambda_{i\to j} \tag{1.8}$$

The formation of A is then a simple process, given by Algorithm 1.

Algorithm 1 Forming A for a cell at time t

Run Neutronics simulation using densities $N(t)$
$\hat{R}_{i\to j}^{(c)} \leftarrow$ the tally of reaction $i \to j$
$P_{\text{unit}}(t) \leftarrow$ the power computed from the neutronics simulation
$R_{i\to j}^{(c)} \leftarrow$ Equation (1.5)
$A_{i,j}^{(c)}(t) \leftarrow$ Equation (1.8)

The general form of this equation, Equation (1.9), is a common one in the field of mathematics. As a consequence, there are dozens of methods developed over the years to solve it. The next several chapters detail some of the many techniques available.

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

A brief introduction into the impact a transport solver has on this ODE is presented in Section 1.1.1. In addition, this particular ODE has a number of stability issues that must be considered during integration. The most fundamental stability issue posed is that of A and L-stability. These are both introduced in Section 1.1.2. The other stability issue is that of xenon oscillations, which is introduced in Section 1.1.3.

1.1.1 Computing Coefficients

There are a wide number of different ways to compute the reaction rates for neutron transport. The broadest categorization of these methods splits the techniques into the stochastic and the deterministic sets. The choice between these two will have a significant impact on depletion and must be considered first.

Deterministic algorithms are the most straightforward. These algorithms solve the neutron transport equation using a fixed process. An example of this kind of method is the Method of Characteristics (MOC) [3]. The result is that for a fixed set of inputs, the output is also fixed. This is convenient, as most

numerical analysis techniques rely on this property. Depending on the level of approximation used, deterministic calculations can be performed in a matter of seconds.

Stochastic methods (for example Monte Carlo [11]) are the opposite. These algorithms use random number sequences to compute the solution to the neutron transport equations by following sampled neutrons. The result is that if one takes the exact same geometry with the exact same initial condition but a different random number sequence, the solutions will be different. One of the main issues with Monte Carlo is its convergence rate. The convergence is often proportional to $1/\sqrt{N}$, where N is the number of particle samples. This convergence rate can make the simulation of a nuclear reactor core with Monte Carlo take several hundreds or thousands of CPU hours to compute.

As such, a deterministic solve can get a fixed, approximate A with no stochastic noise in a relatively short amount of time. Stochastic methods can randomly sample A slowly. These differences will become relevant when choosing a depletion algorithm.

1.1.2 A- and L-Stability

The concepts of A- and L-stability are important in the design and discussion of depletion algorithms. Both of these stability criterion are based on the solutions to a simplified initial value problem (IVP) of the form shown in Equation (1.10) [9].

$$y'(t) = \lambda y(t), y(0) = 1, \lambda \in \mathbb{C}, \Re\lambda < 0 \tag{1.10}$$

This IVP has an analytic solution of the form $y(t) = e^{\lambda t}$. Now, obviously, if $\Re\lambda < 0$, the analytic solution converges to zero at $t \to \infty$. It would thus be valuable if a numerical method also converged to zero on this same problem.

Let us assume one integrates this IVP and generates two successive estimates for y at time step n and $n+1$, which are separated by a step size of h. A function R can be defined as shown in Equation (1.11).

$$\begin{aligned} R(z) &= \frac{y_{n+1}}{y_n} \\ z &= \lambda h \end{aligned} \tag{1.11}$$

Option one for ensuring a method converges to zero is to ensure that any given choice in time step yields an estimate to y closer to zero than the previous estimate. Then, if one took infinite time steps, one is sure to recover the exact answer at $t = \infty$. This leads to the A-stability criterion, which is given by:

$$|R(z)| < 1, \forall z, \Re z < 0.$$

Option two additionally states that if one takes a single infinitely large time step, the exact solution should also appear. This leads to the L-stability criterion:

$$\lim_{|z|\to\infty} R(z) = 0, \forall z, \Re z < 0.$$

As an example, R for explicit Euler and implicit Euler are shown in Equation (1.12) and Equation (1.13) respectively. One can note that explicit Euler will only converge for $|1+z| < 1$ and that implicit Euler is both A- and L-stable.

$$R_{\text{explicit}}(z) = 1 + z \tag{1.12}$$

$$R_{\text{implicit}}(z) = \frac{1}{1-z} \tag{1.13}$$

One can then extend these criteria for a system of ODEs defined by $y' = Ay$ by eigendecomposing the matrix A and redefining y. The result is that for a system of ODEs, there are many λs which are given by the eigenvalues of A, and the requirements must hold for all of them.

So, why is this important for depletion? The problem with depletion is that the span of eigenvalues can be as large as $\lambda \in [-10^{21}, 0]s^{-1}$ [39]. If one used the Euler method, then the method will diverge unless:

$$|1 + \lambda h| < 1, \forall \lambda.$$

The result is that the time step h must be on the order of 10^{-21} seconds or the solution will diverge. This can be mitigated through the use of an A-stable method. However, there are no solvers that operate purely with linear combinations of function evaluations that are both explicit and A-stable [37]. This then leads to a choice between implicit methods (which require multiple function evaluations to converge), or nonlinear methods. The added function cost of implicit methods is usually impractical, so all methods presented in Chapter 2 and Chapter 3 use a nonlinear approach. Some methods are also implicit to solve the xenon instability problem.

1.1.3 Xenon Instabilities

The next form of instability is that of xenon instabilities. Xenon oscillations occur when a slight perturbation in the neutron flux increases the power locally in a region of a reactor. This increases the xenon concentration over several hours. As the xenon builds up, the thermal flux goes down, lowering the power. Without feedback of any form, either through thermal, mechanical, or operator-controlled mechanisms, these oscillations can grow. This effect is most prevalent in large geometries.

This effect is demonstrated in Figure 1-1. Here, a 3D fuel rod is axially dis-

cretized into 8 regions and all boundary conditions are set as reflective. As such, the solution should be uniform. The full model is described in Section 5.2. It is integrated with two different algorithms, the explicit predictor (Section 2.1.1) and the implicit SIE RR (Section 2.1.2.3). SIE is specifically designed to avoid oscillations. As shown, the ^{135}Xe concentration oscillates with a 6-hour time step regardless of algorithm. This process would also likely occur in a real problem if feedback were removed.

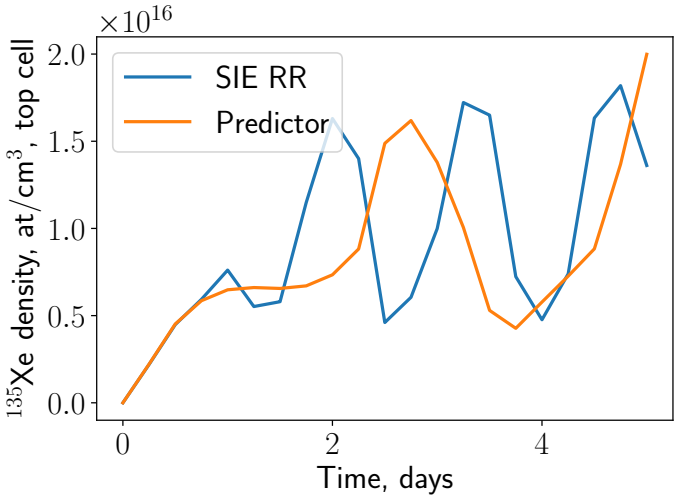


Figure 1-1: Oscillations in depletion with a short time step

	Predictor	SIE RR
Batches Total	5000	5000
Batches Active	2500	2500
Neutrons / Batch	128000	12800
Neutrons Per Time Step	640 million	640 million

Table 1.1: Oscillation demonstration simulation parameters

The problem occurs when one wants to take longer steps than the characteristic time of the xenon oscillation. When this is done, the oscillation will still attempt to occur as shown in Figure 1-2. With some unstable algorithms, such as predictor, this oscillation will grow until the solution is unusable. With stabilized algorithms, such as SIE RR, this does not.

There is yet no clear mathematical understanding for this particular oscillation. The few algorithms that solve it do so either by forcing nuclides to take a non-oscillatory solution or by performing an implicit solve. However, the

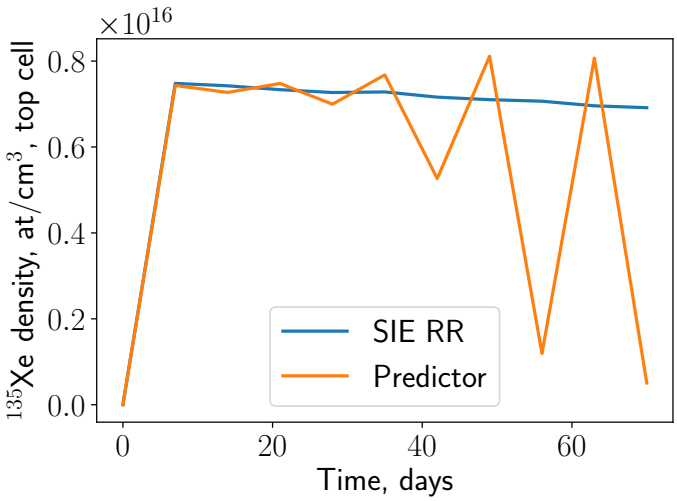


Figure 1-2: Oscillations in depletion with a long time step

particular geometry that generated Figure 1-2 is an extreme case, and many easier geometries can be depleted using normal methods as will be shown in Section 5.1 and Section 5.2.2.

1.2 Implementation Issues

There are three major implementation concerns. The first two, concerning data, do not significantly impact the choice of algorithm, but are mentioned to explain some decisions made in testing later. The two data issues involve the proper computation of power inside of a nuclear reactor (Section 1.2.1) and isomer capture branching ratios (Section 1.2.2). The third issue is that of memory usage. Depletion as a process can take a tremendous amount of memory. Algorithms that require more memory than others can make some geometries impossible to solve with contemporary clusters. The memory issue is briefly discussed in Section 1.2.3.

1.2.1 Calculating Power

The energy released by a single fission is a transient process. There is the prompt energy deposition at the moment of fission and the decay of the daughter nuclei at some time in the future. Ideally, a depletion solver would handle this correctly. The issue is that the total energy deposition is known much more accurately than the prompt energy deposition. For example, the ^{235}U `MT = 458` data is given in Table 1.2. Here, the "Total energy less the energy of neutrinos" is

known quite accurately, but the sum of the prompt values is not.

Quantity	Value (eV)	Uncertainty (eV)
Kinetic energy of fission products	1.6913×10^8	4.9000×10^5
Kinetic energy of prompt fission neutrons	4.8380×10^6	7.0000×10^4
Kinetic energy of delayed fission neutrons	7.4000×10^3	1.1100×10^3
Total energy released by prompt γ	6.6000×10^6	5.0000×10^5
Total energy released by delayed γ	6.3300×10^6	5.0000×10^4
Total energy released by delayed β	6.5000×10^6	5.0000×10^4
Energy carried away by neutrinos	8.7500×10^6	7.0000×10^4
Total energy less the energy of neutrinos	1.9341×10^8	1.5000×10^5

Table 1.2: ^{235}U `MT = 458` data from ENDF-B/VII.1

This presents a choice on where one wants to approximate, and as such there are a number of options. The first is to simply assume a prompt energy deposition for all recoverable fission energy. Then, to compute the energy from neutron capture one can either track the fission products and capture products separately or compute a time integral approximation for total energy released per capture. The first option requires a more complicated solver and runs into memory issues (see Section 1.2.3). The second option is the one used in CASMO-5 [41], and has the issue that the capture energy deposition is spectrum-dependent.

A second option is to consider prompt and delayed energy separately. This restores temporal accuracy. However, since the total less neutrinos fission value is known quite accurately but the prompt value is not, the uncertainty will be higher. A possible improvement is to compute the total recoverable energy released from the decay of the fission yield spectrum via the decay chain and subtract it from the total less neutrino value to get an effective "prompt" value. Thus, the high accuracy total less neutrino `MT = 458` value is conserved and one still gets transient energy deposition.

A third option is to fold an approximate neutron capture value into the prompt fission (for example, by computing approximate energy per capture and captures per fission). The result, usually around 200 MeV for ^{235}U, is extremely easy to implement, but is both spectrum-dependent and temporally inaccurate.

The final option is to take the third option and track decay heat using a decay heat correlation ODE. This technique, presented in [40], restores temporal information to method three. However, these correlations are also spectrum-dependent.

For this document, option three with 200 MeV for all fissionable isotopes will be used unless otherwise noted, as it is simple, easy to implement, and

has little to no bearing on the comparison of the numerical methods. It is recommended to not use this technique in a real depletion solver.

1.2.2 Isomeric Branching Ratios

For certain reactions, the most well-known being ^{241}Am (n, γ), several results can occur. For this example, either ^{242}Am or ^{242m}Am can be produced. In general, two approaches have been taken for the computation of the rate of production of each isomer. Option one is to use pre-computed branching ratios in which the branching ratio has been integrated over an approximate energy spectrum of the neutron flux. Option two is to recompute the branching ratio using the actual flux for each function evaluation. The second method typically yields results closer to experiment, but it is still very sensitive to the quality of the underlying library [21].

For the depletion solver used in this paper, constant branching ratios from the Serpent wiki [1] will be used. While these are less accurate than proper energy dependent branching ratios, current versions of OpenMC do not support directly tallying production of isomers.

1.2.3 Memory Concerns

The last major implementation concern is the minimization of memory utilization. In the ENDF-B/VII.1 library, there are roughly 3800 nuclides with 2000 reaction channels in the set of fission, (n, γ), (n,p), (n,α), and (n, xn). There are a number of ways to simplify this decay chain [20], but this analysis will be performed using these full decay chains.

There are several ways to tally reaction rates. Option one is to simply directly tally each channel, for a total of approximately 2000 values. Option two is to use a coarse group flux, such as 63 groups, and use spectrum-dependent group collapsed cross sections to compute reaction rates [23]. While efficient, this limits the ability to analyze reactors with spectra significantly different from that used to generate the cross section library. Option three is to tally an ultrafine group flux, which can require up to 43000 values [22]. Depending on implementation, the flux-based tallies are much faster than reaction rate tallies. For this research, the first option will be used.

Two reactor models will be discussed. Both of them are based on the specifications of the Seabrook Station reactor listed in Appendix K of [47]. This reactor has 50952 fuel pins, each with an active height of 365.8 cm. The "medium accuracy" model discretizes each pin into 24 axial regions, each with a height of 15.2 cm. The "high accuracy" model discretizes the axial dimension into 366 regions. In addition, each pin is radially discretized into 10 equal volume rings. As such, the medium accuracy model has 1.22 million regions,

	Model (GB)	
	Medium	High
Nuclides	37.2	5700
Reaction Rate Tally	19.6	3000
Coarse Flux	0.6	93
Fine Flux	420.6	64200

Table 1.3: Memory utilization for nuclides and reaction rates for two different reactor models

and the high accuracy model has 186 million regions.

Assuming double precision to store reaction rates and nuclides, each region will require 30.4 kB to store nuclides and 16 kB to store reaction rates. The resulting memory usage for both reactor examples is listed in Table 1.3. Regardless of what one chooses for the high accuracy model, the memory utilization will be well in excess of contemporary single node computer memory limits. As such, domain decomposition would be essential. Similarly, the fine flux tally may make even the medium model impractically expensive.

Bringing this back to the choice of algorithm, some high order methods require the storage of multiple reaction rates or nuclide quantities. This increase in cost, coupled with the fact that the memory requirements are already very large may make some problems impractical to solve. This increase in memory utilization must be balanced against accuracy and stability requirements.

1.3 Goals

The primary goal of this project is to accelerate depletion simulations to enable full core reactor simulations. However, this is a fairly broad statement, so it is worthwhile to consider how one could go about this. Every algorithm has a number of properties: arithmetic cost, accuracy, stability, and memory requirements. Each of these can be a viable avenue to improve depletion.

The arithmetic cost of an algorithm is the number of raw mathematical operations necessary to complete a time step. For many algorithms, this cost is dominated by the number of matrix exponentials used. For some algorithms, specifically the polynomial algorithms CE/LI (see Section 2.1.2.1) and LE/QI (see Section 2.1.2.2) it is possible to reduce the number of matrix exponentials without compromising accuracy by performing the integration in a different way. This improved algorithm is introduced in Section 3.1.

The accuracy of an algorithm is arguably its most important property, and any improvements would be welcome. Current depletion methods mostly all come from the same family of algorithms, the predictor-corrector family, which

are discussed in Section 2.1.2. Several other families of algorithms were investigated. One such family, the commutator-free Lie integrators (see Section 2.2.3), were developed in other fields to solve mathematically similar problems. In addition, two new families of algorithms (extended predictor-corrector, Section 3.3, and exponential-linear, Section 3.4) were derived from scratch in the search for improved accuracy.

As mentioned in Section 2.1.3, the stability of methods is also very important to a depletion algorithm. One tested improvement was to recast the CE/LI and LE/QI methods into an implicit form, which was done in Section 3.2.

Finally, with regard to memory, the memory requirements for all new algorithms were computed and compared against current methods. The result is a balancing act between the capability of an algorithm and the memory requirements of said algorithm. This balancing act is sensitive to a user's needs.

Chapter 2

Current Numerical Methods

In this chapter, current methods for the solution of $y' = A(y,t)y$ will be discussed. These are all L-stable methods. Except for stochastic implicit Euler, all methods are also explicit. Section 2.1 presents an overview of the algorithms that are currently used in depletion. This includes the predictor and predictor-corrector families. Section 2.2 then presents some of the techniques commonly used in other fields such as the Runge–Kutta–Munthe-Kaas methods and other Lie algebra based integrators. Finally, since all of these algorithms make heavy use of the matrix exponential, important considerations for matrix exponential algorithms are shown in Section 2.3.

2.1 Current Depletion Algorithms

The majority of current depletion methods reside in the matrix exponential predictor and predictor-corrector families of algorithms. These methods make various approximations to A and then leverage the matrix exponential (discussed more thoroughly in Section 2.3) to integrate forwards in an L-stable manner. While these algorithms are easy to implement and easy to understand, they are more limited in construction when compared to other methods.

2.1.1 The Predictor Method

The easiest way to make an L-stable method for depletion is to approximate A as the constant matrix A_n over an interval $t \in [t_n, t_n + h]$. Under these circumstances, the ODE now has, in principle, an analytic solution, given in Equation (2.1).

$$\begin{aligned} A_n &= A(y_n, t_n) \\ y_{n+1} &= \operatorname{expm}(hA_n)\, y_n \end{aligned} \tag{2.1}$$

The function "expm" is the matrix exponential. This method, while extremely simple, is not without drawbacks. Structurally, the algorithm error converges $\mathcal{O}(h)$. As such, asymptotically, the error is proportional to the step size. Changes in A during a time step are completely ignored, resulting in errors resolving rapid changes in reaction rates (for example, irradiation of a burnable poison). As such, predictor is often too inefficient to use.

2.1.2 Predictor-Corrector

The predictor-corrector family is a set of multi-stage improvements over the predictor method. Using values of y already known, several values of A are computed. Then, y is integrated using these values to some point in the future (the prediction stage). A is evaluated at this predicted y, and the new value of A is then used to re-integrate to y_{n+1} (the correction stage). Many such methods are available. Several are listed in [30].

The simplest example would be the CE/CM algorithm used in MCNP6 [19]. The algorithm follows Equation (2.2).

$$\begin{aligned} y_{n+1/2} &= \text{expm}\left(\frac{h}{2}A(y_n, t_n)\right) y_n \\ y_{n+1} &= \text{expm}\left(hA(y_{n+1/2}, t_{n+1/2})\right) y_n \end{aligned} \tag{2.2}$$

Here, the value of y at the midpoint is estimated using A at the beginning of time. Then, A is evaluated using this midpoint estimate and integrated to the end of time. This algorithm has a number of advantages, the main one being that the solution converges $\mathcal{O}(h^2)$. However, it is not the only option available.

2.1.2.1 CE/LI

The CE/LI algorithm, used by default in Serpent [34], takes a slightly different approach. In this algorithm, the initial guess of A is integrated to the end of the interval, and an approximate A that is strictly dependent on time is formed by linear interpolation. The process is shown in Equation (2.3).

$$\begin{aligned} y_{n+1}^{(p)} &= \text{expm}\left(hA(y_n, t_n)\right) y_n \\ A_l(t) &= A(y_n, t_n)\left(1 - \frac{t - t_n}{h}\right) + A(y_{n+1}^{(p)}, t_{n+1})\frac{t - t_n}{h} \\ \hat{y}'(t) &= A_l(t)\hat{y}(t), \quad \hat{y}(t_n) = y_n \\ y_{n+1} &= \hat{y}(t_n + h) \end{aligned} \tag{2.3}$$

There are a number of ways to solve the final, simplified ODE. The CMCDT code uses the VODE algorithm to integrate it directly [10]. Serpent uses a

matrix exponential substepping algorithm. For each substep, s, A_l is integrated. Then, the end result is computed via Equation (2.4). $m = 5$ is often used.

$$\begin{aligned} A_s &= \int_{t_n+\frac{s-1}{m}h}^{t_n+\frac{s}{m}h} A(s)ds \\ y(t_n+h) &= \operatorname{expm}(A_m)\operatorname{expm}(A_{m-1})\ldots\operatorname{expm}(A_1)\,y(t_n) \end{aligned} \tag{2.4}$$

In Section 2.2.1, this will be demonstrated to be a very simple multistep Magnus integrator [5]. There are many other choices of Magnus integrators, and some will be tested as replacements in later chapters. However, whatever choice one makes with the secondary integrator, the overall order of the method will be shown to be limited to $\mathcal{O}(h^2)$.

2.1.2.2 LE/QI

Extending the CE/LI idea from the previous method, one could contemplate forming polynomials using information from previous time steps. This is the approach used in the LE/QI method [27], in which a linear polynomial is used to predict, and a quadratic one to correct. The predictor stage is shown in Equation (2.5) and the corrector stage is shown in Equation (2.6). Here, A_{last} and A_0 are the value of A at $t_n - h_1$ and t_n, respectively, and integration is performed to $t_n + h_2$.

$$\begin{aligned} A_l(t) &= -\frac{t-t_n}{h_1}A_{\text{last}} + \left(1+\frac{t-t_n}{h_1}\right)A_0 \\ \hat{y}_l'(t) &= A_l(t)\hat{y}_l(t), \quad y_l(t_n) = y_n \\ y_l &= \hat{y}_l(t_n+h_2) \end{aligned} \tag{2.5}$$

$$\begin{aligned} A_1 &= A(y_l, t_n+h_2) \\ A_q(t) &= \left(-\frac{h_2(t-t_n)}{h_1(h_1+h_2)} + \frac{(t-t_n)^2}{h_1(h_1+h_2)}\right)A_{\text{last}} \\ &\quad + \left(1 - \frac{(h_1-h_2)(t-t_n)}{h_1h_2} - \frac{(t-t_n)^2}{h_1h_2}\right)A_0 \\ &\quad + \left(\frac{h_1(t-t_n)}{h_2(h_1+h_2)} + \frac{(t-t_n)^2}{h_2(h_1+h_2)}\right)A_1 \\ \hat{y}_q(t) &= A_q(t)\hat{y}_q(t), \quad y_q(t_n) = y_n \\ y_{n+1} &= \hat{y}_q(t_n+h_2) \end{aligned} \tag{2.6}$$

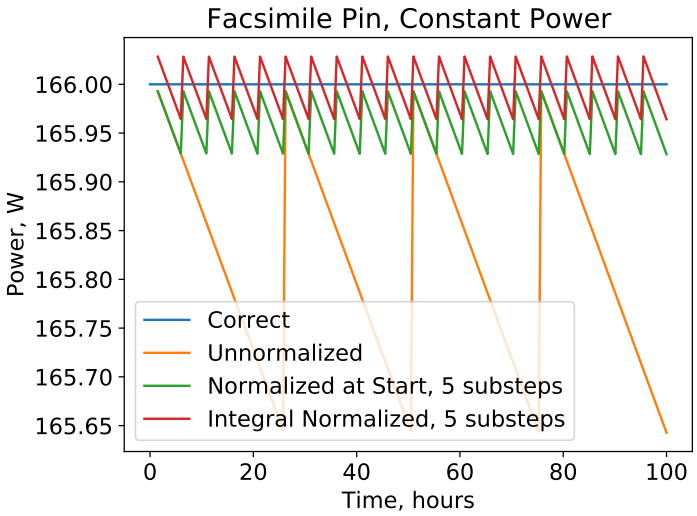


Figure 2-1: Power as a function of time on the facsimile pin for varying normalization, LE/QI, 4 time steps with one initial CE/CM step of 1 hour

The asymptotic order of the method can be sensitive to the integrator used to perform the simplified ODEs. In Section 4.2.3, the convergence with several integrators will be analyzed.

For both of the polynomial-based algorithms, it is additionally possible to slightly tweak how integration is performed to improve accuracy [28]. The idea is that while the power is normalized for each function evaluation, the power is not in fact constant over the time step. Additionally, the integral energy deposition over the time step is usually not correct. This discrepancy is shown in Figure 2-1, in which the LE/QI method is used to integrate the facsimile pin problem from Chapter 4. The unnormalized power is off by a maximum of 0.2%. One avenue of improvement is to renormalize the power at each substep using either the beginning of time or midpoint power. Another avenue is to add the following ODE to the matrix exponential:

$$\frac{dq}{dt} = \sum_{i,j} Q^{(r)}_{i\to j} \Sigma_{i\to j} y_i(t) + \sum_{i,j} Q^{(d)}_{i\to j} \lambda_{i\to j} y_i(t), \quad q(t_0) = 0$$

where q is the energy deposited during a substep. By renormalizing the reaction rates with the result for this ODE, one can achieve more accurate results. The beginning of substep and integral renormalization results are also shown in Figure 2-1, in which both improve results greatly.

The main drawback to these improvements is that the renormalization makes numerical analysis far more difficult. As such, it is difficult to make any guarantees about the performance of these algorithms after renormalization.

Secondarily, the integral method requires multiple matrix exponentials during iteration and the substep renormalization requires substeps. This added cost may make these algorithms impractical for simpler deterministic simulations.

2.1.2.3 Stochastic Implicit Methods

The stochastic implicit Euler (SIE) method was developed in order to solve the xenon oscillation problem [15, 16]. The predictor method is transformed into the implicit form and stochastic gradient descent is used to solve for the nuclide quantity and reaction rate that forms a consistent system. This results in an algorithm that takes the form of Equation (2.7). This specific form is the "relaxation of the neutron flux" version, which, since the computation of reaction rates from the flux is a linear operation, can be recast as a "relaxation of the reaction rate". An alternative form is to compute the mean of the nuclides instead.

$$\begin{aligned} y_{n+1}^{(0)} &= \text{expm}(hA(y_n, t_n))y_n \\ A_{n+1}^{(i)} &= A(y_{n+1}^{(i)}, t_{n+1}) \\ \bar{A}_{n+1}^{(i)} &= \frac{1}{i+1}\sum_{j=0}^{i} A_{n+1}^{(j)} \\ y_{n+1}^{(i)} &= \text{expm}(h\bar{A}_{n+1}^{(i-1)})y_n \\ y_{n+1} &= y_{n+1}^{(m)} \end{aligned} \tag{2.7}$$

Here, m is the number of substeps, and i is the current substep. In the original paper, a value of $m = 10$ was used.

Typically, the flux will tend to counter a xenon oscillation, which will eventually bring it back to equilibrium. For every substep, the flux is allowed to counter the current xenon oscillation. The averaged reaction rate will then result in a smaller xenon oscillation for the next substep. For a sufficiently large m, the numerical oscillations will dampen out completely.

While this method can help substantially with xenon oscillations, it is also asymptotically first order. So, while the method is very stable, it is not very accurate. This limits its practical use.

2.1.3 Stabilizing ^{135}Xe by Asserting a Solution

Another way to eliminate xenon oscillations is to simply assert that the solution takes a particular form. There are two methods derived for Monte Carlo to improve ^{135}Xe stability. The first, tested in [29], is to assume that both ^{135}Xe and ^{135}I are in equilibrium. To enforce this, reaction rates are tallied for every batch of a Monte Carlo simulation. Then, using these values, the equilibrium

^{135}Xe and ^{135}I values are solved for and used in the next batch. Depletion is performed normally, but the final ^{135}Xe and ^{135}I values are overwritten by the mean during the active cycles of the simulation. This method acts similar to SIE in allowing the flux to counter the oscillation, but it also eliminates any time evolution of either nuclide from the problem.

The second method, presented in [50] operates much the same, except it is no longer assumed that ^{135}Xe and ^{135}I are in equilibrium. The assumption is instead that the initial concentrations are zero at $t = 0$. Each batch is updated by computing the solution to Equation (2.8) with this initial condition. In effect, these two nuclides are handled by a single step of stochastic implicit Euler, while all other nuclides are handled normally.

$$\begin{aligned} N'_I(t) &= \sum \gamma_I \Sigma_f \phi - \lambda_I N_I(t) \\ N'_{Xe}(t) &= \sum \gamma_{Xe} \Sigma_f \phi + \lambda_I N_I(t) - (\sigma_a^{Xe} \phi + \lambda_{Xe}) N_{Xe}(t) \end{aligned} \tag{2.8}$$

The result is that the nuclide concentration of ^{135}Xe and ^{135}I can evolve in time using a single step of a first order algorithm. For small t, the xenon concentration can vary. As t increases, the result approaches the equilibrium solution above.

Both algorithms have been shown to successfully stabilize the depletion process. However, since the iodine and xenon concentrations are updated within the transport operator, these approaches are incompatible with a depletion wrapper approach. In addition, overwriting the concentrations at the end of the depletion has an unknown impact on the numerical properties of both algorithms.

2.2 Current Methods from Other Fields

In order to truly improve the state-of-the-art in depletion, it is worthwhile to consider algorithms from other fields. This is the primary motivation behind deriving depletion in the form $y' = A(y, t)y$. Integration of stiff equations of this form are a relatively popular topic in the numerical method field, so a huge number of options are available.

First, a brief detour in Section 2.2.1 is made to discuss solutions for the nonautonomous linear ODE $y' = A(t)y$. Then, the Runge–Kutta–Munthe-Kaas family are introduced in Section 2.2.2, followed by the commutator-free Lie integrator family in Section 2.2.3.

2.2.1 Integrators for $y' = A(t)y$

In the earlier predictor-corrector discussion, it was noted that the nonautonomous linear ODE appears a number of times as a subproblem that needs to be solved. The Magnus expansion is an infinite series solution to this ODE [5]. The Magnus expansion takes the form of Equation (2.9).

$$\begin{aligned} y(t) &= \exp \Omega(t) y(0) \\ \Omega(0) &= \mathbf{O} \\ \Omega(t) &= \sum_{k=1}^{\infty} \Omega_k(t) \end{aligned} \tag{2.9}$$

Essentially, the solution to $y(t)$ is given by a single matrix exponential of the matrix $\Omega(t)$. This matrix is formed via an infinite sum of terms, of which the first three are given below. Here, $[\cdot, \cdot]$ is the matrix commutator, where $[A, B] = AB - BA$.

$$\Omega_1(t) = \int_0^t A(t_1)\, dt_1$$

$$\Omega_2(t) = \frac{1}{2} \int_0^t \int_0^{t_1} [A(t_1), A(t_2)]\ dt_2\, dt_1$$

$$\Omega_3(t) = \frac{1}{6} \int_0^t \int_0^{t_1} \int_0^{t_2} ([A(t_1), [A(t_2), A(t_3)]] + [A(t_3), [A(t_2), A(t_1)]])\ dt_3\, dt_2\, dt_1$$

Further terms become increasingly complex. One way to transform this into an effective integrator is to truncate the sum in Equation (2.9). If accuracy is still too low, one could perform substepping in which, instead of integrating in one shot to $t + h$, m integrations of h/m are performed. If one performs this substepping with Ω truncated to Ω_1, the result is Equation (2.10). This happens to be identical to the substepping method used for the CE/LI and LE/QI predictor-corrector methods.

$$\begin{aligned} A_s &= \int_{t_n + \frac{s-1}{m} h}^{t_n + \frac{s}{m} h} A(s) ds \\ y(t_n + h) &= \operatorname{expm}(A_m) \operatorname{expm}(A_{m-1}) \ldots \operatorname{expm}(A_1)\, y(t_n) \end{aligned} \tag{2.10}$$

A much cheaper approach than directly evaluating the expansion is to form a quadrature. There are a wide variety of quadrature options. One option, Equation (2.11), only requires one matrix exponential, but requires matrix

commutators in order to do so [5].

$$\begin{aligned} c &= \frac{1}{2} \mp \frac{\sqrt{3}}{6} \\ A_i &= A(t + c_i h) \\ \Omega^{[4]}(h) &= \frac{h}{2}\left(A_1 + A_2\right) - \frac{h^2\sqrt{3}}{12}\left[A_1, A_2\right] \\ y(t+h) &= \exp\left(\Omega^{[4]}(h)\right) y(t) \end{aligned} \tag{2.11}$$

Unfortunately, this particular method proved to be unstable during testing. An alternative form, and one that did not have such an issue, is the commutator-free integrator shown in Equation (2.12) [46]. This form removes the need to compute commutators in exchange for requiring multiple matrix exponentials. This algorithm will be abbreviated as "CFQ4" in the rest of the paper.

$$\begin{aligned} c &= \frac{1}{2} \mp \frac{\sqrt{3}}{6} \\ a &= \frac{1}{4} \pm \frac{\sqrt{3}}{6} \\ A_i &= hA(t + c_i h) \\ y(t+h) &= \exp\left(a_2 A_1 + a_1 A_2\right) \exp\left(a_1 A_1 + a_2 A_2\right) y(t) \end{aligned} \tag{2.12}$$

It is unclear why one works and the other does not, but there are at least a few cases where the use of commutators reduces the stability of numerical integration [45]. Additionally, methods based on the Magnus expansion directly may fail if the expansion does not converge. This can happen if

$$\int_0^t \|A(s)\|_2 \, ds \geq \pi.$$

The other problem is that the eigenvalues of Ω are not known very rigorously. This can cause issues with the Chebyshev rational approximation matrix exponential recommended for use with depletion (see Section 2.3).

2.2.2 Runge–Kutta–Munthe-Kaas Integrators

The Runge–Kutta–Munthe-Kaas (RK-MK) integrators are a relatively recent innovation in the field of numerical integrators. By leveraging different exponential maps, RK-MK acts as a coordinate-free form of Runge-Kutta [26]. This flexibility allows one to solve a problem using an exponential map of their choice. As such, "exp" will be used instead of "expm," as this is a more general

solution. In the depletion case, the exponential map of interest is the matrix exponential. The method is as follows. Given Equation (1.9), the solution is asserted to be

$$y(t) = \exp(\Theta(t))y_0.$$

One can then derive a new ODE to solve for the matrix Θ, which is given by Equation (2.13).

$$\begin{aligned} \Theta'(t) &= \operatorname{dexp}^{-1}_{\Theta(t)}(A(y,t)), \quad \Theta(0) = \mathbf{O} \\ \operatorname{dexp}^{-1}_{A}(C) &= \sum_{j=0}^{\infty} \frac{B_j}{j!} \operatorname{ad}^j_A(C) \\ \operatorname{ad}^0_A(C) &= C \\ \operatorname{ad}^j_A(C) &= \left[A, \operatorname{ad}^{j-1}_A(C)\right] \end{aligned} \tag{2.13}$$

Here, B_j are the Bernoulli numbers, ad_A is the adjoint action, and $[\cdot,\cdot]$ is the Jacobi-Lie bracket (in the matrix exponential case, this is also the matrix commutator). Equation (2.13) can then be solved with Runge-Kutta.

Given an explicit Butcher's tableau [9] of the form shown in Equation (2.14), the resulting algorithm takes the form of Algorithm 2. Here, $\operatorname{dexp}^{-1}(\Theta_k, A_k, p)$ is the $p-1$ order truncation of $\operatorname{dexp}^{-1}_{\Theta_k}(A_k)$.

Algorithm 2 Runge–Kutta–Munthe-Kaas Explicit Form

$p \leftarrow$ the order of the underlying Runge–Kutta scheme
$v \leftarrow$ the number of stages
$i \leftarrow 0$
$y_n \leftarrow$ the current value of y
for $i \leq s$ **do**
  $\Theta_i \leftarrow \sum_{j=1}^{i-1} a_{i,j} F_j$
  $A_i \leftarrow hA(\exp(\Theta_i) y_n, t_n + c_i h)$
  $F_i \leftarrow \operatorname{dexp}^{-1}(\Theta_i, A_i, p)$
  $i \leftarrow i + 1$
end for
$\Theta \leftarrow \sum_{j=1}^{v} b_j F_j$
$y_{n+1} \leftarrow \exp(\Theta) y_n$

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & \dots & \dots & a_{s,s-1} & \\ \hline & b_1 & \dots & \dots & b_{s,s-1} & b_s \end{array} . \tag{2.14}$$

One might notice some level of similarity to prior methods. Applying Algorithm 2 to that of the Butcher's tableau in Equation (2.15) (which is the explicit Euler tableau) results in the method in Equation (2.16). This exactly matches that of predictor.

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array} \tag{2.15}$$

$$\begin{aligned} \Theta_1 &= \mathbf{O} \\ A_1 &= hA(t_n, y_n) \\ F_1 &= \operatorname{dexp}^{-1}(\Theta_1, A_1, 1) = A_1 \\ \Theta &= F_1 \\ y_{n+1} &= \exp(\Theta) y_n = \exp\left(hA(t_n, y_n)\right) y_n \end{aligned} \tag{2.16}$$

Similarly, let us consider the tableau given by Equation (2.17). The resulting method is Equation (2.18). Aside from the commutator in F_2, it is otherwise identical to the CE/CM predictor-corrector method.

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array} . \tag{2.17}$$

$$
\begin{aligned}
\Theta_1 &= \mathbf{O} \\
A_1 &= hA(t_n, y_n) \\
F_1 &= \text{dexp}^{-1}(\Theta_1, A_1, 2) = A_1 \\
\Theta_2 &= \frac{F_1}{2} \\
A_2 &= hA\left(t_n + \frac{h}{2}, \exp(\Theta_2) y_n\right) \\
F_2 &= \text{dexp}^{-1}(\Theta_2, A_2, 2) = A_2 - \frac{1}{2}\left[\Theta_2, A_2\right] \\
\Theta &= F_2 \\
y_{n+1} &= \exp(\Theta) y_n
\end{aligned} \tag{2.18}
$$

However, it does get a bit more interesting. The use of $\text{dexp}^{-1}(\Theta_i, A_i, p)$ is enough to guarantee order p, but it is not the minimum-cost case. F can be reduced further using Lie-Butcher series [36]. As it turns out, the commutator is not required for second order methods. It is only order three and higher in which commutators are required.

There are a few issues with RK-MK schemes when applied to depletion. At least a few are shared with the Magnus integrators in the previous section: commutators may reduce the stability of the method and the eigenvalues of Θ are not known rigorously. Additionally, as the order increases, the number of commutators increases, which takes time to compute and consumes more memory.

2.2.3 Commutator-Free Methods

The commutator-free methods attempt to alleviate a few of the issues with RK-MK methods. By using multiple exponentials per function evaluation, it becomes possible to create a high order method without the use of commutators [13]. Similar to RK-MK, any exponential map can be used with this algorithm, but for depletion a matrix exponential will be used. The algorithm takes the form of Equation (2.19).

$$
\begin{aligned}
Y_r &= \exp\left(\sum_k \alpha_{rJ}^k F_k\right) \ldots \exp\left(\sum_k \alpha_{r1}^k F_k\right) y_0 \\
F_r &= hA(Y_r) \\
y_1 &= \exp\left(\sum_k \beta_J^k F_k\right) \ldots \exp\left(\sum_k \beta_1^k F_k\right) y_0
\end{aligned} \tag{2.19}
$$

The coefficients are then listed off in a tableau similar to that of Runge-Kutta Butcher tableaux. One trick is to reuse intermediate stages to eliminate matrix exponents. For example, in the CF3 algorithm given by Equation (2.20), one can reuse the value Y_2 in the y_1 calculation to eliminate a matrix exponential.

$$\begin{aligned} F_1 &= hA(y_0) \\ Y_2 &= \exp\left(\frac{1}{3}F_1\right) y_0 \\ F_2 &= hA(Y_2) \\ Y_3 &= \exp\left(\frac{2}{3}F_2\right) y_0 \\ F_3 &= hA(Y_3) \\ y_1 &= \exp\left(-\frac{1}{12}F_1 + \frac{3}{4}F_3\right) \exp\left(\frac{1}{3}F_1\right) y_0 \end{aligned} \tag{2.20}$$

There is also a fourth order method named CF4. New commutator-free methods are relatively easy to derive using the technique described in [38].

2.3 Matrix Exponents

In the algorithms listed prior, a common theme is the use of the matrix exponential. There are many matrix exponential algorithms, owing to both the difficulty of the matrix exponential and its utility in other fields [35]. Each method provides some level of tradeoff, and so careful consideration is important.

There are two main considerations for a good depletion matrix exponential. First, a single A evaluation typically has, as mentioned prior, eigenvalues in the span $\lambda \in [-10^{21}, 0]s^{-1}$ [39]. As such, high accuracy with extreme negative eigenvalues is important, but accuracy for positive eigenvalues is less important. Second, the matrix A is typically quite sparse, on the order of 0.2%. Some algorithms can leverage this sparsity to improve computational performance.

In this section, three algorithms will be introduced: Chebyshev rational approximation, scaling and squaring, and Krylov subspace methods. Their relative capabilities will be discussed. Then, each method will be tested on a simple benchmark problem to demonstrate their capabilities.

2.3.1 Chebyshev Rational Approximation

The Chebyshev rational approximation method (CRAM) is a relatively straightforward algorithm. A rational function $\hat{r}_{k,k}(x)$ is found that minimizes the maximum error with regard to the scalar exponent along the negative real axis [39].

The defining equation is Equation (2.21), where $\pi_{k,k}$ is the set of all rational functions with numerators and denominators of order k. As k increases, the accuracy of the approximation also increases.

$$\sup_{x\in\mathbb{R}_-} |\hat{r}_{k,k}(x) - e^x| = \inf_{r_{k,k}\in\pi_{k,k}} \left\{ \sup_{x\in\mathbb{R}_-} |r_{k,k}(x) - e^x| \right\} \tag{2.21}$$

Once the function $\hat{r}_{k,k}(x)$ is known, it can be rearranged to reduce costs further or to improve numerical stability. The incomplete partial fraction (IPF) form, shown in Equation (2.22), is a good combination of numerical stability and efficiency. The values α_l and θ_l are tabulated and are available for a variety of values of k up to 48. In the IPF form, only sparse matrix solves are necessary to compute the action on a vector.

$$\hat{r}_{k,k}(x) = \alpha_0 \prod_{l=1}^{k/2} \left(1 + 2\Re \left\{ \frac{\tilde{\alpha}_l}{x - \theta_l} \right\} \right) \tag{2.22}$$

CRAM is both efficient and highly accurate over the domain in which it is derived. However, eigenvalues with extremely large imaginary components or positive real components will reduce the accuracy. As such, CRAM is not recommended for use in highly oscillatory problems or those with possible exponential growth such as reactor dynamics.

2.3.2 Scaling and Squaring

Scaling and squaring relies on the transformation in Equation (2.23), which holds for all non-negative n [35]. n is chosen such that that new matrix $\hat{A} = A/2^n$ can be easily exponentiated, typically with a low order Padé approximant.

$$e^A = \left(e^{\frac{A}{2^n}} \right)^{2^n} \tag{2.23}$$

Several nearly-optimal schemes have been developed over the years [25, 2]. The first one is built-in to the Julia programming language [4], and so will be the one tested.

The advantage of this method is that it is valid for any set of eigenvalues. However, the repeated squaring necessary to efficiently compute the 2^n exponential for large eigenvalues has drawbacks. First, as n becomes large, rounding errors will accumulate. Second, the matrix must be treated as dense during this operation for efficiency.

2.3.3 Krylov Subspace

The Krylov subspace approach is another algorithm that reduces the matrix exponent into a simpler problem [43]. However, unlike scaling and squaring, the Krylov subspace method applies the action of a matrix on a vector instead of computing the matrix exponential itself. This is beneficial, as such a solution is all that is needed. Assuming one wants $e^A v$, an Arnoldi factorization of A is performed:

$$\begin{aligned} V_1 &= v/\|v\|_2 \\ AV_m &= V_m H_m + h_{m+1,m} v_{m+1} e_m^T \end{aligned}$$

Once this is done, the exponential is then given by Equation (2.24).

$$\begin{aligned} e^A v &\approx \|v\|_2 V_{m+1} e^{\overline{H}_m} e_1 \\ \overline{H}_m &= \begin{pmatrix} H_m & 0 \\ h_{m+1,m} e_m^T & 0 \end{pmatrix} \end{aligned} \tag{2.24}$$

This inner matrix exponential can be performed using a method described in the prior sections. Since $\overline{H}_m$ is an $m+1 \times m+1$ matrix, when m is small the exponential can be quite efficient. One can then use successive iterates of the Arnoldi factorization to determine when the method has converged.

The key advantage of this method is that it is simultaneously general and operates with sparse matrices efficiently. However, the method only converges to the accuracy of the inner matrix exponent, so accuracy issues with said algorithm are still important. In addition, it deletes "low importance" information, so for tightly coupled reaction/decay chains the Krylov subspace method poses no benefit.

2.3.4 Exponential Testing

In this section, three algorithms will be tested on an example A matrix and y_0 vector. The test matrix is the initial condition for the facsimile pin problem described in Chapter 4, which is an 11×11 matrix with eigenvalues spanning $\lambda \in [-2.772, 0]$. While more difficult matrices are easy to generate, creating a reference solution for a realistic problem becomes more and more difficult. y_0 is the initial nuclide quantity vector from the same problem. The reference solution was computed in 100 digit arithmetic using the matrix exponential in Mathematica [49] for various times from 1 second to 10^9 seconds. After 10^9 seconds, the exponential for this matrix becomes essentially steady state and uninteresting.

For CRAM, there are various orders of approximation available, from order 4 to order 48. The error is compared in Figure 2-2. Here, it is notable that

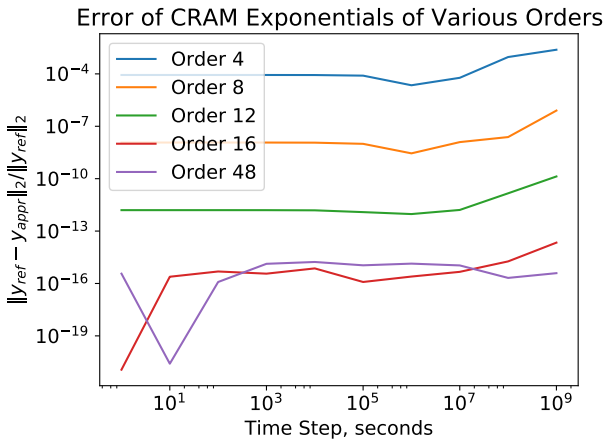


Figure 2-2: Error between the approximate and reference solution for various orders of CRAM for time steps from 1 second to 10^9 seconds

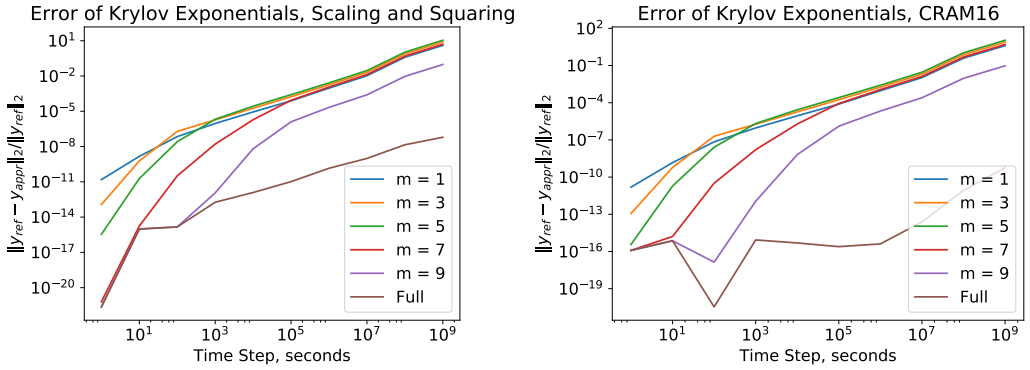


Figure 2-3: Error between the approximate and reference solution for the Krylov method with various Arnoldi factorization steps for both inner exponentials

the error is roughly proportional to 10^{-p}, where p is the order of the method. In addition, around $t = 10^7$, all exponents except the 48th order method see the beginning of the growth of error. Overall, the high order CRAM methods perform extremely well.

The next set of comparisons are with regard to the Krylov method. As the Krylov method requires an inner exponential, both CRAM and scaling and squaring were tested, as shown in Figure 2-3. For both, as the number of steps in the Arnoldi factorization increased, both converged towards the inner integrator. However, even with short time steps, the accuracy of the non-full exponential is particularly poor. On this test problem, the Krylov method is not recommended.

Finally, a direct comparison between CRAM and scaling and squaring is

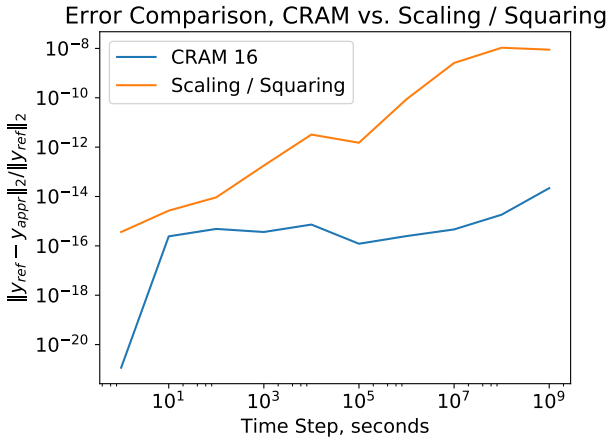


Figure 2-4: Error between the approximate and reference solution for CRAM and scaling and squaring from 1 second to 10^9 seconds

done in Figure 2-4. Notably, as the time step increases (and thus the eigenvalue of the matrix fed into the matrix exponential increases), the scaling and squaring method begins to become less and less accurate. For CRAM, however, this does not happen. In a real depletion problem, the eigenvalues will be much larger. As such, the scaling and squaring algorithm is not recommended for depletion.

In summary, of the three algorithms, only CRAM has sufficient accuracy as to be suitable for depletion. Of the CRAM methods, order 16 is the first one to approach machine precision for even small eigenvalues and is the lowest order recommended for use. In Chapter 5, when testing on real depletion problems is performed, the $k = 48$ CRAM method will be used to minimize the possible impacts of CRAM itself.

Chapter 3

New Methods

Although the previous chapter introduced a number of algorithms from other fields that could be used for depletion, this is not the limit of what can be done. Other possibilities include improving current depletion methods or deriving custom algorithms specifically optimized for depletion. Both routes will be taken in this chapter.

With regard to improving current methods, there are three options. The first is to replace the substepping algorithm used in polynomial based predictor-corrector methods. This can reduce the cost and complexity of these algorithms. The process is discussed in Section 3.1. A second possibility is to implement those same algorithms in stochastic implicit mode, as is done in Section 3.2. A third possibility is to add more intermediate stages to predictor-corrector to improve the final estimate of A over the interval. This "extended predictor-corrector" is derived in Section 3.3.

As for the custom algorithms, an algorithm structurally similar to the commutator-free Lie integrators from the previous chapter is derived from scratch in Section 3.4. These exponential-linear integrators are then optimized to minimize statistical noise and to stay as stable as possible in a depletion-type problem using techniques developed in Section 3.5. Finally, while deterministic adaptive time stepping is a well-understood topic, nearly nothing has been developed with regard to when A is randomly sampled from a distribution (via, in this case, Monte Carlo transport). Section 3.6 introduces a stochastic-aware adaptive time stepping algorithm.

3.1 Replacing the Subintegrator in CE/LI and LE/QI

In both CE/LI and LE/QI (Section 2.1.2), $y' = A(y,t)y$ is transformed into the much simpler $y' = A(t)y$, in which $A(t)$ is a polynomial in time with

matrix coefficients. The current way to perform this secondary integration is via exponentiation with the average of the polynomial over short intervals.

There are two problems with this approach. The first is that such an algorithm is $\mathcal{O}(h^2/m^2)$ and would prevent any otherwise third order algorithm (such as LE/QI) from being third order overall. The second issue is that m substeps requires m matrix exponentials. If A is cheap to evaluate, such as with deterministic transport, then the modest improvement may not be worth the cost.

As discussed in Section 2.2.1, it is possible to use a higher order quadrature-based method. Specifically, the commutator-free fourth order integrator is a promising candidate, as it only requires two matrix exponentials. By using Equation (2.12) to integrate the nonautonomous ODEs in Equation (2.3), CE/LI then takes the form of Equation (3.1). Similarly, solving the nonautonomous ODEs in Equation (2.5) and Equation (2.6) with Equation (2.12) yields Equation (3.2). These forms do not have power normalization built-in, but can be modified to do so. These combined algorithms will be tested against the substepping versions throughout the next few chapters.

$$\begin{aligned}
A_0 &= A(y_n, t_n) \\
y_{n+1}^{(p)} &= \operatorname{expm}(hA_0)y_n \\
A_1 &= A(y_{n+1}^{(p)}, t_n + h) \\
y_{n+1} &= \operatorname{expm}\left(\frac{h}{12}A_0 + \frac{5h}{12}A_1\right)\operatorname{expm}\left(\frac{5h}{12}A_0 + \frac{h}{12}A_1\right)y_n
\end{aligned} \tag{3.1}$$

$$\begin{aligned}
A_{\text{last}} &= A(y_{n-1}, t_n - h_1) \\
A_0 &= A(y_n, t_n) \\
y_{n+1}^{(p)} &= \operatorname{expm}\left(-\frac{5h_2^2}{12h_1}A_{\text{last}} + \frac{h_2\,(6h_1 + 5h_2)}{12h_1}A_0\right) \\
&\quad \operatorname{expm}\left(-\frac{h_2^2}{12h_1}A_{\text{last}} + \frac{h_2\,(6h_1 + h_2)}{12h_1}A_0\right)y_n \\
A_1 &= A(y_{n+1}^{(p)}, t_n + h_2) \\
y_{n+1} &= \operatorname{expm}\left(-\frac{h_2^3}{12h_1\,(h_1 + h_2)}A_{\text{last}} + \frac{(h_1^2 + 2h_2h_1 + h_2^2)\,h_2}{12h_1\,(h_1 + h_2)}A_0 + \frac{(5h_1^2 + 4h_2h_1)\,h_2}{12h_1\,(h_1 + h_2)}A_1\right) \\
&\quad \operatorname{expm}\left(-\frac{h_2^3}{12h_1\,(h_1 + h_2)}A_{\text{last}} + \frac{(5h_1^2 + 6h_2h_1 + h_2^2)\,h_2}{12h_1\,(h_1 + h_2)}A_0 + \frac{h_1h_2}{12\,(h_1 + h_2)}A_1\right)y_n
\end{aligned} \tag{3.2}$$

3.2 Stochastic Implicit CE/LI and LE/QI

In the stochastic implicit Euler (SIE) algorithm in Section 2.1.2.3, each time step is performed several times. For each attempt, the reaction rates of prior attempts are averaged. The advantage of this technique is that it allows for oscillations to dampen out. One possible modification to the CE/LI and LE/QI algorithms is to perform the same process with the end point reaction rates. A similar process was performed with a logarithmic interpolation in [33]. Take the CE/LI algorithm (Equation (2.3)) for example. The modified algorithm takes the form of Algorithm 3.

Algorithm 3 Stochastic Implicit CE/LI

$y_0 \leftarrow$ initial condition
$t \leftarrow$ vector of times
$A_{\text{BOS}} \leftarrow A(y_0, t_0)$
$s \leftarrow$ the number of stages
for $n = 1 \rightarrow$ number of steps **do**
  $y_{n+1}^{(p)} \leftarrow \text{expm}(hA_{\text{BOS}})y_n$
  $A_{\text{EOS}}^{(p)} \leftarrow A(y_{n+1}^{(p)}, t_{n+1})$
  $A_l^{(p)}(t) \leftarrow A_{\text{BOS}}\left(1 - \frac{t-t_n}{h}\right) + A_{\text{EOS}}^{(p)}\frac{t-t_n}{h}$
  Solve ODE: $\hat{y}_p'(t) = A_l^{(p)}(t)\hat{y}_p(t), \quad \hat{y}_p(t_n) = y_n$
  $y_{n+1}^{(c,0)} \leftarrow \hat{y}_p(t_{n+1})$
  for $m = 1 \rightarrow s$ **do**
    $A_{\text{EOS}}^{(c,m)} \leftarrow A(y_{n+1}^{(c,m-1)}, t_{n+1})$
    $\overline{A}_{\text{EOS}}^{(c,m)} \leftarrow \frac{1}{m}\sum_{j=1}^{m} A_{\text{EOS}}^{(c,j)}$
    $A_l^{(c,m)}(t) \leftarrow A_{\text{BOS}}\left(1 - \frac{t-t_n}{h}\right) + \overline{A}_{\text{EOS}}^{(c,m)}\frac{t-t_n}{h}$
    Solve ODE: $\hat{y}_{c,m}'(t) = A_l^{(c,m)}(t)\hat{y}_{c,m}(t), \quad \hat{y}_{c,m}(t_n) = y_n$
    $y_{n+1}^{(c,m)} \leftarrow \hat{y}_{c,m}(t_{n+1})$
  end for
  $A_{\text{BOS}} \leftarrow \overline{A}_{\text{EOS}}^{(c,s)}$
  $y_{n+1} \leftarrow y_{n+1}^{(c,s)}$
end for

There are a few things to note about this algorithm. First, the corrector is only computed using reaction rates computed via previous corrector stages. This may not be necessary, but it makes the algorithm consistent. Second, due to the fact that $A_{\text{EOS}}^{(c)}$ converges to the end point reaction rate via rate relaxation, it can be reused for the next step if there are no discontinuities. If there are, it should be recomputed. Finally, the first A_{BOS} only consists of one single function evaluation. Unlike SIE, this value is used directly. For statistics

reasons, it should be computed with s times as many neutrons as later function evaluations.

This exact same structure can be performed with LE/QI (using stochastic implicit CE/LI to compute the first time step), resulting in a stabilized implicit third order method. For testing, $s = 10$ was used (for a total of 11 function evaluations per time step) and the fourth order commutator free integrator from Section 2.2.1 was used to solve the linear and quadratic matrix polynomial ODEs.

3.3 Extended Predictor-Corrector

One option for the general improvement of predictor-corrector is to add more intermediate stages and combine them in such a way as to reduce the error of the problem [31]. If the extended predictor-corrector algorithm is defined as in Equation (3.3), then the coefficients a, b, and c take a rather familiar form. If arranged in a Butcher's tableau like those of Runge-Kutta, many predictor-corrector methods then take the form of valid Runge-Kutta tableaux. One then could consider using a high-order Runge-Kutta method instead.

$$\begin{aligned} x_1 &= y_n \\ x_i &= \operatorname{expm}\left(h \sum_{j=1}^{i-1} a_{ij} A\left(x_j, t_n + c_j h\right)\right) y_n \\ y_{n+1} &= \operatorname{expm}\left(h \sum_{j=1}^{s} b_j A\left(x_j, t_n + c_j h\right)\right) y_n \end{aligned} \tag{3.3}$$

These methods can also be derived from another direction. If one takes the Runge–Kutta–Munthe-Kaas methods from Section 2.2.2 and set p to 1, the result is a valid algorithm with a maximum general order of 2 in which all commutators are eliminated. Interestingly, in a commuting algebra (such as scalar arithmetic), $\operatorname{dexp}^{-1}(a, b, p) = \operatorname{dexp}^{-1}(a, b, 1)$. In such an algebra, an extended predictor-corrector method will have the same order as the underlying Runge-Kutta method for single ODEs.

Two such methods will be tested. The first uses the tableau of the classic RK4 method, given below. This combined algorithm is labeled as "EPC-RK4"

in the rest of this paper.

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1/2	
	1/6	1/3	1/3	1/6

The second is the Cash-Karp 6 stage 5th order method [12]. This tableau also has embedded in it 4 other estimates of the final value for orders 1-4. Due to the truncation of dexp^{-1}, these added terms are not usable in the EPC form and as such are not listed. The resulting tableau is given below. This combined algorithm will be referred to elsewhere as "EPC-RK45".

0						
1/5	1/5					
3/10	3/40	9/40				
3/5	3/10	−9/10	6/5			
1	−11/54	5/2	−70/27	35/27		
7/8	1631/55296	175/512	575/13824	44275/110592	253/4096	
	37/378	0	250/621	125/594	0	512/1771

3.4 The Exponential-Linear Method

The exponential-linear (EL) family was derived by noting that a Richardson extrapolation of CE/CM successfully created a third order algorithm for a non-commuting problem [31]. The resulting algorithm is shown in Equation (3.4). This method requires 5 evaluations of A. There are two key differences between this algorithm and the extended predictor-corrector discussed prior: the

exponentials no longer operate on just y_n, and vectors can be summed.

$$\begin{aligned}
x_1 &= \text{expm}\left(\frac{h}{2}A\left(y_n, t\right)\right) y_n \\
x_2 &= \text{expm}\left(\frac{h}{4}A\left(y_n, t\right)\right) y_n \\
x_3 &= \text{expm}\left(\frac{h}{2}A\left(x_2, t+\frac{h}{4}\right)\right) y_n \\
x_4 &= \text{expm}\left(\frac{h}{4}A\left(x_3, t+\frac{h}{2}\right)\right) x_3 \\
y_{n+1} &= \frac{4}{3}\text{expm}\left(\frac{h}{2}A\left(x_4, t+\frac{3h}{4}\right)\right) x_3 - \frac{1}{3}\text{expm}\left(hA\left(x_1, t+\frac{h}{2}\right)\right) y_n
\end{aligned} \tag{3.4}$$

One then can imagine a purely generic form of this in which any combination of exponentials operating on prior vectors can be summed. The explicit s stage form is given as Equation (3.5).

$$\begin{aligned}
x_1 &= y_n \\
x_{i+1} &= \sum_{j=1}^{i} d_{ij}\text{expm}\left(h\sum_{k=1}^{i} a_{ijk}A(x_k, t_n + c_k h)\right) x_j \\
y_{n+1} &= x_{s+1}
\end{aligned} \tag{3.5}$$

A careful observer would note that this algorithm is very closely related to the commutator-free Lie integrators discussed in Section 2.2.3. For efficiency, the commutator-free methods tend to reuse previous vector fields. In the EL family, this reuse is implicit in the algorithm. The difference is in the summation of vectors.

The derivation of the coefficients is also different. While the CF algorithms are derived using a rather elegant ordered tree approach using Lie algebra such that the exponential map is arbitrary, the EL methods are derived assuming the exponential map is the matrix exponential. It is unclear if the EL methods can be rederived similarly, as the vector addition lies outside of the matrix Lie algebra.

Section 3.4.1 goes over precisely how this derivation was performed. The result is an underdetermined system of equations that must be solved. As it is underdetermined, it is useful to add more constraints and optimization goals to improve the accuracy, stability, and statistical properties of the algorithm. These additional goals are listed in Section 3.4.2. Finally, a discussion on multiple estimates of y_{n+1} for adaptive time stepping is presented in Section 3.4.3.

3.4.1 Deriving the Coefficients of EL Methods

In order to generate an order p method, the easiest way is to perform a series expansion of $y(h)$ and $y_{\text{approx}}(h)$ with regard to h. An order p method arises if all terms $\mathcal{O}(h^m), m \leq p$ are equal between $y(h)$ and $y_{\text{approx}}(h)$. Alternatively, one could say that $y(h) - y_{\text{approx}}(h) = \mathcal{O}(h^{p+1})$. This derivation is performed similarly to how Runge-Kutta is derived [8]. The next two sections derive the series expansion for $y(h)$ and $y_{\text{approx}}(h)$ respectively.

3.4.1.1 Series Expansion of $y(h)$

The analysis of $y(h)$ begins with a Taylor series expansion about $h = 0$. The first few terms are as follows:

$$y(h) = y(0) + y'(0)h + \frac{y''(0)}{2!}h^2 + \mathcal{O}(h^3)$$

The ODE itself defines the first two terms. $y(0)$ is merely the initial condition for the step. y' is then the ODE itself: $y' = F(y)y$. The rest of the terms can be derived by taking successive derivatives of y'. To do this, it is useful to rewrite y' in the form of Equation (3.6). Here, f_{ij} is the value of the $[i, j]$ entry of matrix F.

$$y_i' = \sum_{j=1}^{N} f_{ij}(y)y_j \tag{3.6}$$

In this form, the chain rule can be easily applied. Whenever y' appears, Equation (3.6) can be substituted back into the equation. The result for the second derivative is Equation (3.7).

$$\begin{aligned} y_i'' = &\sum_{j=1}^{N} f_{ij}(y) \sum_{k=1}^{N} f_{jk}(y)y_k \\ &+ \sum_{j=1}^{N}\sum_{k=1}^{N} \frac{\partial f_{ij}(y)}{\partial y_k} y_j \sum_{l=1}^{N} f_{kl}(y)y_l \end{aligned} \tag{3.7}$$

Continuing the derivation with this particular form will rapidly become unwieldy, so a set of operators will be defined. The first three are listed in

Equation (3.8). The vectors u, v, w and x are arbitrary here.

$$\begin{aligned}
\mathbf{f}_u v &= \sum_{j=1}^{N} f_{ij}(u) v_j \\
\mathbf{f}_u{}'(v; w) &= \sum_{k=1}^{N} \sum_{j=1}^{N} \frac{\partial f_{ij}(u)}{\partial u_k} v_j w_k \\
\mathbf{f}_u{}''(v; w, x) &= \sum_{l=1}^{N} \sum_{k=1}^{N} \sum_{j=1}^{N} \frac{\partial^2 f_{ij}(u)}{\partial u_k \partial u_l} v_j w_k x_l
\end{aligned} \tag{3.8}$$

One can readily extend these to higher order by adding more derivatives and more vectors. These operators also have a few useful properties. The first is that the operators are linear over each vector. The second is that all vectors after the semicolon are interchangeable. The third is that the application of the chain rule on an operator allows for the construction of higher order derivatives efficiently:

$$\begin{aligned}
\frac{\partial \mathbf{f}_u{}^{(n)}(v; w, x, \ldots)}{\partial t} &= \mathbf{f}_u{}^{(n+1)}(v; w, x, \ldots, u') \\
&\quad + \mathbf{f}_u{}^{(n)}(v'; w, x, \ldots) \\
&\quad + \mathbf{f}_u{}^{(n)}(v; w', x, \ldots) \\
&\quad + \mathbf{f}_u{}^{(n)}(v; w, x', \ldots) + \ldots
\end{aligned}$$

Using these operators, Equation (3.6) and Equation (3.7) can be rewritten as Equation (3.9) and Equation (3.10), respectively.

$$y' = \mathbf{f}_y y \tag{3.9}$$

$$y'' = \mathbf{f}_y{}'(y, \mathbf{f}_y y) + \mathbf{f}_y \mathbf{f}_y y \tag{3.10}$$

The third derivative also has a somewhat compact form:

$$\begin{aligned}
y''' &= 2\mathbf{f}_y{}'(\mathbf{f}_y y; \mathbf{f}_y y) + \mathbf{f}_y \mathbf{f}_y{}'(y; \mathbf{f}_y y) + \mathbf{f}_y \mathbf{f}_y \mathbf{f}_y y \\
&\quad + \mathbf{f}_y{}''(y; \mathbf{f}_y y, \mathbf{f}_y y) + \mathbf{f}_y{}'(y; \mathbf{f}_y{}'(y; \mathbf{f}_y y)) \\
&\quad + \mathbf{f}_y{}'(y; \mathbf{f}_y \mathbf{f}_y y)
\end{aligned}$$

This can be continued for each derivative until the $y(h)$ series is known to suitable order.

3.4.1.2 Series Expansion of $y_{\mathbf{approx}}(h)$

The primary constituent of y_{approx} is that of a matrix exponential operating on a vector. These too can be decomposed into the same operator forms by leveraging the infinite sum definition of the matrix exponential:

$$u = \text{expm}\left(\sum_j h a_j F(x_j)\right) v$$
$$u = \sum_{k=1}^{\infty} \frac{1}{k!} \left(\sum_j a_j h \mathbf{f}_{x_j}\right)^k v$$

The resulting expansion of u truncated to second order is:

$$u = v + h\sum_i a_i \mathbf{f}_{x_i} v + \frac{h^2}{2} \sum_i \sum_j a_i a_j \mathbf{f}_{x_i} \mathbf{f}_{x_j} v + \mathcal{O}(h^3)$$

A comparable Taylor series can be found by taking the derivatives of u with respect to h as is done in the $y(h)$ case above. The derivatives of x_i and v will appear and can be substituted in. The process of setting $h = 0$ at the end will transform the $\mathbf{f}_{x_i}$ into $\mathbf{f}_{x_i(0)}$. If $x_i(0) = y(0)$, then the entire Taylor series of u will be available using the same operators as $y(h)$. y_{approx} is then the sum of several such series.

3.4.1.3 Constructing Constraints

From the above two sections, the series of $y(h)$ and $y_{\text{approx}}(h)$ are now known in terms of combinations of ${\mathbf{f}_{y(0)}}^{(m)}$ to arbitrary order. To generate the constraint equations, the series are subtracted from one another and all terms $\mathcal{O}(h^m), m \leq p$ are set to zero. An example set of coefficients are shown in Table 3.1. If all order ≤ 2 coefficients are set to zero and additionally $\sum_j d_{ij} = 1$ (this ensures that $x_i(0) = y(0)$), the result is a valid second order method. The last coefficient is notable in that it cannot be set to zero, indicating that a two-stage method is limited to second order.

3.4.2 EL Constraints and Optimization Goals

It is often possible to add a few more constraints to the system of equations for exponential linear as the number of unknowns and the number of constraints do not usually match exactly. These additional constraints can improve the stability of the algorithm or improve the accuracy. The next few sections discuss some of these additional constraints.

Term	Order	Coefficient
y	0	$-1 + d_{21} + d_{11}d_{22}$
$\mathbf{f}_y y$	1	$-1 + a_{211}d_{21} + a_{212}d_{11}d_{21} + d_{11}(a_{111} + a_{221} + a_{222}d_{11})d_{22}$
$\mathbf{f}_y\mathbf{f}_y y$	2	$\frac{1}{2}\left(-1 + (a_{211} + a_{212}d_{11})^2 d_{21} + d_{11}(a_{111} + a_{221} + a_{222}d_{11})^2 d_{22}\right)$
$\mathbf{f}_y{}'(y; \mathbf{f}_y y)$	2	$\frac{1}{2}\left(-1 + 2a_{111}d_{11}^2(a_{212}d_{21} + a_{222}d_{11}d_{22})\right)$
$\mathbf{f}_y\mathbf{f}_y\mathbf{f}_y y$	3	$\frac{1}{6}\left(-1 + (a_{211} + a_{212}d_{11})^3 d_{21} + d_{11}(a_{111} + a_{221} + a_{222}d_{11})^3 d_{22}\right)$
$\mathbf{f}_y(\mathbf{f}_y{}'(y; \mathbf{f}_y y))$	3	$\frac{1}{6}\left(-1 + 3a_{111}d_{11}^2(a_{212}(a_{211} + a_{212}d_{11})d_{21} + a_{222}d_{11}(a_{221} + a_{222}d_{11})d_{22})\right)$
$\mathbf{f}_y{}''(y; \mathbf{f}_y y, \mathbf{f}_y y)$	3	$\frac{1}{6}\left(-1 + 3a_{111}^2 d_{11}^3(a_{212}d_{21} + a_{222}d_{11}d_{22})\right)$
$\mathbf{f}_y{}'(y'; \mathbf{f}_y\mathbf{f}_y y)$	3	$\frac{1}{6}\left(-1 + 3a_{111}^2 d_{11}^2(a_{212}d_{21} + a_{222}d_{11}d_{22})\right)$
$\mathbf{f}_y{}'(\mathbf{f}_y y; \mathbf{f}_y y)$	3	$\frac{1}{6}\left(-2 + 3a_{111}d_{11}^2(a_{212}(a_{211} + a_{212}d_{11})d_{21} + a_{222}d_{11}(2a_{111} + a_{221} + a_{222}d_{11})d_{22})\right)$
$\mathbf{f}_y{}'(y; \mathbf{f}_y{}'(y; \mathbf{f}_y y))$	3	$-\frac{1}{6}$

Table 3.1: Formal series coefficients for $y - y_{\text{approx}}$ for a 2-stage exponential linear method to third order

3.4.2.1 Stability Constraints

In an actual depletion problem, it is impossible for any component of y to go below zero due to the physics. There are two ways that this can occur in an algorithm. The first is if any coefficient a_{ijk} is negative. This can allow a matrix to have negative reaction rates, at which point the algorithm is no longer stable. Similarly, if any d_{ij} is negative, the sum of vectors can be negative.

A more serious issue arises if *all* a_{ijk} for a given ij are negative. If this is the case, the resulting matrix is, in effect, going to have negative decay constants. Under these conditions, it is likely that some eigenvalues will become positive and very large in magnitude. This would prevent the use of the CRAM matrix exponential.

Ideally one would prefer all coefficients to be non-negative, but it may not necessarily be possible. During the derivation of the third order algorithm, no set of coefficients was found such that all a_{ijk} were non-negative. It is unclear if negative a_{ijk} are necessary to go beyond second order. For the fourth order algorithm, no set of coefficients was found such that all d_{ij} were non-negative. In this case, however, it may simply be due to the difficulty of solving such a huge number of constraints.

As a suitable compromise, all sets of coefficients were derived with Equation (3.11) added to the constraints list. For the third order method, Equation (3.12) was added. For the fourth order method, Equation (3.13) was

enforced. This yielded reasonable quality algorithms.

$$\sum_k a_{ijk} \geq 0 \tag{3.11}$$

$$d_{ij} \geq 0 \tag{3.12}$$

$$\sum_j d_{ij} \geq 0 \tag{3.13}$$

3.4.2.2 Accuracy and Statistics Goals

The other way to improve a method is not through constraints but through optimization goals. The first optimization goal uses an approach similar to [6]. In the case of an order p method, all terms $\mathcal{O}(h^m), m \leq p$ cancel. As a result, the asymptotic error is now driven by the remaining $\mathcal{O}(h^{p+1})$ terms. If one defines τ as in Equation (3.14), where $D_k^{(p+1)}$ are the order $p+1$ elementary differentials (chains of $\mathbf{f}_y$ from earlier), then one can also define Equation (3.15).

$$y - y_{\text{approx}} = \sum_k h^{p+1} \tau_k^{(p+1)} D_k^{(p+1)} + \mathcal{O}(h^{p+2}) \tag{3.14}$$

$$T_{p+1} = \sqrt{\sum_k \left(\tau_k^{(p+1)}\right)^2} \tag{3.15}$$

Ignoring high order effects, as T_{p+1} shrinks, the overall accuracy of the method improves. If it is zero, the method becomes $p+1$ order. The minimization of T_{p+1} was added to the optimization goal for each order p method.

The other optimization goal was to minimize statistical uncertainty. In Section 3.5, a method for generating a crude first-order approximation of the growth of uncertainty relative to predictor is derived. This approximate ratio, $\sigma_{\text{method}}/\sigma_{\text{predictor}}$, was also added to the optimization goal. The resulting overall goal is Equation (3.16).

$$G = T_{p+1} + \frac{\sigma_{\text{EL}p}}{\sigma_{\text{predictor}}} \tag{3.16}$$

The combination of the constraints and objectives was fed into the Mathematica `NMinimize` function [49] to generate coefficients. High precision was used where possible. As `NMinimize` is stochastic, 10 random sequences were run and the one with the lowest possible objective function was selected. The result is the set of coefficients listed in Appendix A.

3.4.3 Multiple Solution Estimates in EL Methods

Another useful feature for the exponential-linear family would be the implementation of multiple estimates of the final solution of differing order. For example, a third order method with an embedded second order method could leverage the difference between the two for adaptive time stepping as shown in Section 3.6.

Developing a multiple-solution EL method is relatively easy. One just needs to derive the equations for one more stage than necessary. Then, instead of setting $y_{n+1} = x_{s+1}$ and determining constraints that way, one sets:

$$y_{n+1} = x_s$$
$$\hat{y}_{n+1} = x_{s+1}$$

The result is that x_s becomes the order $p+1$ estimate to y. Then, one computes the p order constraints for x_{s+1} and solves those too such that x_{s+1} is an order p estimate. If x_{s+1} is allowed to use $F(x_s)$, the algorithm becomes what is known as First Same As Last (FSAL), as the value $F(x_s)$ can be reused in the next time step. Integration is then continued with the higher order y_{n+1}.

The question then is what constitutes a "good" second estimate. The method used here is again modeled after [6] in which three coefficients are optimized. Here, $\hat{T}_p$ is simply T_p but for $\hat{y}_{n+1}$.

$$B = \frac{\hat{T}_{p+1}}{\hat{T}_p}$$
$$C = \frac{\sqrt{\sum \left[\hat{\tau}_k^{(p+1)} - \tau_k^{(p+1)}\right]}}{\hat{T}_p}$$
$$R = \frac{\max |\hat{\tau}_k^{(p)}|}{\min |\hat{\tau}_k^{(p)}|}$$

The optimal value for each coefficient is rather complicated. For B, a large value indicates the low order solution will be dominated by the high order error at large h, leading to poor estimates of error. A small value of B will make adaptive time stepping less efficient. For C, a large coefficient indicates that the error estimate can be dominated by $p+1$ order error terms. For R, a large coefficient indicates that some elementary differentials are weighted much more heavily than others. Further, in the exponential-linear case, some $\hat{\tau}_k^{(p)}$ will always be zero as a valid EL method will also cancel out some of the higher order terms. These must be excluded from R to prevent an infinite value. The

overall objective for the second estimate is given as:

$$G_2 = |1 - B| + |1 - C| + R + \frac{\hat{\sigma}_{\mathrm{EL}p}}{\sigma_{\mathrm{predictor}}}$$

3.5 The Relative Variance of Methods

One problem with depletion when performed with Monte Carlo is that A is sampled from a distribution instead A being of known exactly. Under these circumstances, it is important to choose an algorithm that minimizes the variance of the final result. Thus, a simple estimate comparing algorithms in the form of a ratio of standard deviations would be useful. The following estimate, while incredibly crude, can yield an estimate quite close to the real value, as will be shown in Section 4.2. The process makes the following assumptions:

- $A(y) = f(y) + x$, where $x = \mathcal{N}(0, \sigma)$ and f is an arbitrary scalar function.
- Every evaluation of A for a different y yields a statistically independent value.
- The first order variance formula (Equation (3.17)) is valid.
- At the limit of $h \to 0$, the ratio of the variance of one method to the other can be given by the first term of the Taylor series.
- The variance in A is proportional to the number of neutrons and thus the runtime of the simulation. This is not always the case in a reactor simulation.

$$\sigma_f^2 = \left(\frac{\partial f}{\partial a}\right)^2 \sigma_a^2 + \left(\frac{\partial f}{\partial b}\right)^2 \sigma_b^2 + \dots \tag{3.17}$$

Using these assumptions, the ratio σ_a/σ_b for two methods can easily be computed. The ratio of CE/CM to predictor will be used as an example. Starting with predictor, the standard deviation is given by Equation (3.18).

$$\begin{aligned} y_{n+1} &= \exp\left[h(f(y_n) + x_1)\right] y_n \\ \sigma^2 &= \left(\frac{\partial y}{\partial x_1}\right)^2 \sigma_1^2 \\ \sigma_{\mathrm{pred}}^2 &= \exp\left[2hf(y_n)\right] y_n^2 h^2 \sigma_1^2 \end{aligned} \tag{3.18}$$

As only one function evaluation is performed for the predictor method, $\sigma_1 = \sigma$. The same can be done for CE/CM, yielding the far more complex Equation (3.19).

$$\begin{aligned} y_{n+1/2} &= \exp\left[\frac{h}{2}(f(y_n) + x_1)\right] y_n \\ y_n &= \exp\left[h(f(y_{n+1/2}) + x_2)\right] y_n \\ \sigma^2_{\text{CE/CM}} &= \frac{1}{4}\exp\left[2hf(y_{n+1/2})\right] y_0^2 h^2 \left(4\sigma_2^2 + \exp\left[hf(y_0)\right] y_0^2 h^2 \sigma_1^2 f'\left(y_{n+1/2}\right)^2\right) \end{aligned} \tag{3.19}$$

To keep an equivalent runtime cost to the predictor method, each simulation is run with half the number of neutrons. As a result, $\sigma_1 = \sigma_2 = \sqrt{2}\sigma$.

The ratio of $\sigma_{\text{CE/CM}}$ and $\sigma_{\text{predictor}}$ can be computed using just these values, but without knowing what f is, the ratio is not useful. However, by performing a Taylor series of the ratio about $h = 0$ and keeping only the first term, f disappears. The final result for the ratio of CE/CM to predictor is then given by Equation (3.21).

$$\frac{\sigma^2_{\text{CE/CM}}}{\sigma^2_{\text{predictor}}} = 2 + \mathcal{O}(h^1) \tag{3.20}$$

$$\frac{\sigma_{\text{CE/CM}}}{\sigma_{\text{predictor}}} = \sqrt{2} + \mathcal{O}(h^1) \tag{3.21}$$

This ratio is surprisingly accurate, as will be shown in Section 4.2. As computed via Monte Carlo sampling, this ratio is 1.439 ± 0.022, which is within statistics of $\sqrt{2}$. This holds for most every other algorithm tested. So, while this process has numerous approximations, it can get quite close to the real answer.

3.6 Adaptive Time Stepping

There are multiple goals of an adaptive time stepping scheme. The first is to allow a method to reduce or increase the time step without user input. A good example for the utility of this is an unexpected rapid transient in the problem. If a user uses fixed time steps of length much longer than this transient, the smearing might substantially reduce the accuracy of the method. Conversely, if the problem starts approaching a steady state, it would be useful to increase the time step to reduce costs.

The second goal is to provide some degree of certainty to the solution. If one chooses a tolerance τ of 0.1, and then re-runs the problem with a tolerance of 0.01, one would prefer the second solution's global error to be 0.1 times the first run. Adaptive time stepping makes this approximately so. However,

it is critical to note two things. First, this is not a guarantee. Second, the global error of the second run is not 0.01, nor can the global error be estimated without re-running the simulation and performing a comparison.

An extremely thorough discussion of adaptive time stepping can be found in [44]. In summary, one creates a numerical integrator with two solutions: a pth order solution and a $p+1$th order solution. At each step, the difference between the two solutions is used to compute some sort of local error estimate. Then, the ratio of this estimate to the tolerance is used to compute the next time step. An example updating scheme is shown in Equation (3.22). Here, θ is a safety factor to reduce the number of thrown-away error estimates. More complex methods might ensure that the step size does not increase or decrease too far (for example, by a factor of two).

$$h_{n+1} = \theta h_n \left(\frac{\tau}{\text{err}}\right)^{1/(p+1)} \tag{3.22}$$

Of all the algorithms previously discussed, only a few have the correct components to provide such an error estimate. In CE/LI, the extrapolation stage is first order and the interpolation stage is second. Similarly, LE/QI has a second order extrapolation and third order interpolation. EL3, as discussed earlier, has a wide variety of second order estimates available. Finally, if a Runge-Kutta method with multiple order solutions is used, the resulting RKMK method would also provide such an estimate.

The interesting part comes from the different types of depletion problems. For a deterministic problem, the only issue is the choice of error to converge with. This will be discussed in Section 3.6.1. For a stochastic problem, however, this error value is essentially sampled from a distribution in which the true mean is the exact error estimate. A time stepping scheme that utilizes this will be discussed in Section 3.6.2.

3.6.1 Choice of Error

During an adaptive time stepping process, the algorithm will attempt to keep some error estimate constant. As such, the choice of error estimate is quite important. A good error estimate will have the following properties:

- Scale invariant
- Geometry invariant
- Captures some physical phenomenon of interest
- Does not allow unmeasured error to grow unbounded

Starting with scale invariancy, it is important to have an error estimate such that when unknowns change rapidly in magnitude, the method is still consistent. For example, if one were using absolute error of $^{157}\mathrm{Gd}$, an error of 10^{15} atoms is minuscule when the gadolinium quantity is at 10^{20} atoms, but could completely change the problem if the gadolinium is supposed to be 10^{14} atoms.

Geometry invariancy is quite similar. If one runs a simple pincell with a tolerance τ, and then runs an entire assembly with the same tolerance τ, a good method would end up with approximately the same error.

For physical phenomenon, there are a few that would be interesting. The most obvious is the nuclide quantity itself. As the integrator is directly computing nuclide quantity, this is easy to track. A second choice would be reaction rate. This metric poses a bit of a problem in that the reaction rate at the end of step is not computed until the beginning of the next step. Some of the adaptive time stepping methods described later compute new neutrons per simulation in response to the error. As such, the dependency chain is cyclical. An approximation would be simply to use the beginning of step reaction rate as a weight for the end of step nuclide quantites.

Finally, for the unmeasured error, it is critical that whatever is measured is "important" to the problem at hand. The error in, say, $^{28}\mathrm{Cl}$ is so loosely coupled to the problem that using it for an error estimate would yield poor results. Similarly, only considering nuclides with a particular effective half life would lead to time steps on the order of that half life, ignoring all nuclides with shorter ones.

As such, two different error estimates will be tested. The first is a nuclide-based error estimate. The mean relative difference and standard deviation of the relative difference between the high and low order estimate is computed. Then, for all nuclides with quantities above 10^{10} in the high order estimate, the root-mean-square of the difference and standard deviation is computed. The relative difference makes the method scale invariant, the root-mean-square makes it geometry invariant, and almost all nuclides are tracked. The downside of this method is that root-mean-square can be overly sensitive to outliers. The

resulting method is:

$$\begin{aligned}
\delta^{(i)} &= \frac{\hat{y}_{n+1}^{(i)} - y_{n+1}^{(i)}}{y_{n+1}^{(i)}}, i = 1 \to s \\
\mu_{\text{vec}} &= \frac{1}{s} \sum_{i}^{s} \delta^{(i)} \\
\sigma_{\text{vec}} &= \sqrt{\frac{1}{s-1} \sum_{i}^{s} \left(\delta^{(i)} - \mu_{\text{vec}}\right)} \\
\mu &= \text{RMS}\left(\mu_{\text{vec}}, y_{n+1} > 10^{10}\right) \\
\sigma &= \text{RMS}\left(\sigma_{\text{vec}}, y_{n+1} > 10^{10}\right)
\end{aligned}$$

where s is the number of samples, $\hat{y}$ is the low order estimate, and y is the high order estimate. Each calculation is done element-wise for vectors.

The second error estimate is a reaction rate-based method. The difference between nuclide quantities in the high and low order estimator is computed and weighted by the sum of all beginning of step reaction rates. From this, the mean and standard deviation are evaluated and summed together. To make the method scale and geometry invariant, the end of step nuclide quantity is weighted by the beginning of step reaction rate and used to normalize the mean and standard deviation. The resulting method is:

$$\begin{aligned}
\delta_{a}^{(i)} &= \left(\hat{y}_{a,n+1}^{(i)} - y_{a,n+1}^{(i)}\right) \sum_{b} R_{a \to b,n}, i = 1 \to s \\
\mu_{\text{vec}} &= \frac{1}{s} \sum_{i}^{s} \delta^{(i)} \\
\sigma_{\text{vec}} &= \sqrt{\frac{1}{s-1} \sum_{i}^{s} \left(\delta^{(i)} - \mu_{\text{vec}}\right)} \\
\mu &= \frac{\sum_{a} |\mu_{\text{vec},a}|}{\sum_{a} \left(y_{a,n+1} \sum_{b} R_{a \to b,n}\right)} \\
\sigma &= \frac{\sum_{a} \sigma_{\text{vec},a}}{\sum_{a} \left(y_{a,n+1} \sum_{b} R_{a \to b,n}\right)}
\end{aligned}$$

Of course, there is a very wide variety of other possible error estimates that can be used in analysis. As such, it is recommended that users consider their special needs to generate their own.

3.6.2 Uncertain Error

In the case of Monte Carlo, $A(y,t)$ is no longer exactly known but instead sampled from a distribution. This distribution, at the limit of infinite neutrons, approaches the true value of $A(y,t)$. Similarly, integrating with a sampled A will result in a sampled y_{n+1} and $\hat{y}_{n+1}$, each from its own distribution. If the standard deviations of these distributions are small enough, adaptive time stepping can be carried out exactly as before. However, it is worthwhile to develop a method that simultaneously updates the time step and particle count to maximize the probability that a time step is valid while minimizing the total computational resources consumed during a simulation.

As will be demonstrated in Section 4.3.1, if one takes the difference between y_{n+1} and $\hat{y}_{n+1}$, the distribution formed by this value has a relatively simple structure:

$$\begin{aligned} y_{n+1} - \hat{y}_{n+1} &\approx \mathcal{N}(\mu, \sigma) \\ |\mu| &\propto h^{p+1} + \text{bias} \\ \sigma &\propto \frac{h}{\sqrt{N}} \end{aligned} \tag{3.23}$$

where N is the number of neutrons and p is the order of the low order estimate $\hat{y}_{n+1}$. If the error is sampled over multiple nuclides, it is suggested that μ and σ are computed per-nuclide and then combined (using 2-norm, root-mean-square, etc.) after the fact into a single value as is done in the previous section. The next few sections will discuss how one can sample from this distribution to estimate the next N and h.

3.6.2.1 Naïve Sampling

The simplest approach to the problem is to simply keep N constant and perform time stepping as one would do with a deterministic method. If μ and σ are sufficiently small, then there is high probability that the error estimate will be below the user-defined tolerance. Conversely, a large μ and σ would yield a high probability that a time step is rejected. When the time step is decreased, both μ and σ would decrease.

While this method has the advantage of being simple to implement, there are two issues with this scheme. The first is that if a problem is uncertainty-dominated, updating the time step $\propto (\tau/\text{err})^{1/(p+1)}$ is incorrect and numerous rejections might have to occur before a time step is accepted. The second, and much more severe one, is that the global error does not end up being linearly proportional to the tolerance. These two aspects will be demonstrated in Section 4.3.3.

3.6.2.2 Sample the Error Multiple Times at Each Step

This second method is far more direct. Each time step is run several times to estimate the underlying distribution of the error. The mean and the standard deviation of the mean are computed, yielding the approximate distribution $\mathcal{N}(\mu, \sigma)$. y_{n+1} is taken as the mean of each time step sample of the high order result. A time step is kept if the probability that the distribution is less than the tolerance exceeds some user-defined confidence c as shown in Equation (3.24).

$$\int_{-\tau}^{\tau} \mathcal{N}(x; \mu, \sigma) dx > c \tag{3.24}$$

Computing the new h is also a bit more complex. As the error is distributed as $\mathcal{N}(\mu, \sigma)$, there is no single error estimate. Even worse, there is a nonzero probability for any choice of error. The solution is to compute the largest error to confidence c. This largest error, m, is given by Equation (3.25).

$$\begin{aligned} \int_{\mu-\hat{m}}^{\mu+\hat{m}} \mathcal{N}(x; \mu, \sigma) dx &= c \\ \hat{m} &= \sqrt{2}\sigma \text{erf}^{-1}(c) \\ m &= |\mu| + \hat{m} \end{aligned} \tag{3.25}$$

The other component required to estimate the next time step would be a target error. In this case, the raw tolerance as-is would be a poor choice. If the next time step μ is very close to τ, the step will almost always be rejected via Equation (3.24). Instead, the target error can be set as $\theta_h \tau$, with θ_h below 1. The resulting time step update is given by Equation (3.26).

$$h_{n+1} = h_n \left(\frac{\theta_h \tau}{m} \right)^{1/(p+1)} \tag{3.26}$$

Using the ratios from Equation (3.23), the new N takes the form of Equation (3.27). Similar to h, τ is scaled by the hyperparameter θ_n.

$$N_{n+1} = N_n \frac{h_{n+1}}{h_n} \left(\frac{\sigma}{\theta_n \tau} \right)^2 \tag{3.27}$$

All that is left is to select good values for the hyperparameters θ_h and θ_n. If the time step update works perfectly and $\mu = \theta_h \tau$ and $\sigma = \theta_n \tau$, the next time step should not be rejected. Noting the step rejection criterion Equation (3.24),

at minimum Equation (3.28) must hold.

$$\int_{-\tau}^{\tau} \mathcal{N}(x; \theta_h \tau, \theta_n \tau) dx \leq c \tag{3.28}$$

However, that leaves quite a bit of flexibility in the precise values. An excessively small θ_n will only improve accuracy up to the numerical limit. Similarly, an excessively small θ_h will only improve accuracy up to the stochastic limit. An optimization analysis is performed in Section 4.3. There, several models were run with several numerical algorithms. In general, $\theta_h = 0.5$, $\theta_n = 0.1$ is a suitable general value, but these values are problem-dependent. The complete algorithm is shown in Algorithm 4.

Algorithm 4 Sample Error Multiple Times at Each Step

$\theta_n, \theta_h, c, \tau$ are user-selected
$N_0, h_0 \leftarrow$ an initial guess
$y_0 \leftarrow$ initial condition
$s \leftarrow$ number of samples
while $t \leq t_{\text{end}}$ **do**
  for $i = 1 : s$ **do**
    $y_{n+1}^{(i)}, \hat{y}_{n+1}^{(i)} \leftarrow$ new value using integrator of choice
    $\epsilon^{(i)} \leftarrow$ error estimate
  end for
  $\mu \leftarrow \text{mean}(\epsilon^{(i)})$
  $\sigma \leftarrow \frac{\text{std}(\epsilon^{(i)})}{\sqrt{s}}$
  $h_{n+1} \leftarrow$ Equation (3.26)
  $N_{n+1} \leftarrow$ Equation (3.27)
  if Equation (3.24) is true **then**
    $y_{n+1} \leftarrow \text{mean}(y_{n+1}^{(i)})$
    $t \leftarrow t + h_n$
  end if
end while

While this method is far more effective than naïve sampling, it does have drawbacks. First, tracking the distribution requires additional memory. Second, the decreased neutrons per simulation increases the bias. Bias is when a sampled distribution differs from the true value being sampled, and often appears in Monte Carlo as one decreases the number of neutrons per batch [7]. The next algorithm helps mitigate bias at a slight increase in cost.

3.6.2.3 Update N Per Second Only Occasionally

As will be demonstrated in Section 4.3.3, the necessary value for N/h slowly varies with time (roughly proportional to the square of the power). As such, instead of calculating an estimate for N/h at every time step, it is only necessary to occasionally compute it. This modified algorithm is shown as Algorithm 5.

Algorithm 5 Occasional N/h ATS Scheme

$N_0, h_0 \leftarrow$ an initial guess.
$y_0 \leftarrow$ initial condition
while $t \leq t_{\text{end}}$ **do**
  if N/h needs updating **then**
    $s \leftarrow$ number of samples
    $N \leftarrow \frac{(N/h)\times h}{s}$
    Use the method in Section 3.6.2.2 with N, h to get new N, h
    $N/h \leftarrow \frac{Ns}{h}$
  end if
  $y_{n+1}, \hat{y}_{n+1} \leftarrow$ new value using integrator of choice
  $\sigma \approx \theta_n \tau$
  Reject or keep step and get h_{n+1} using Section 3.6.2.2
  $t \leftarrow t + h_n$ if step kept
end while

It is important to note that the sampling stage uses a factor of s neutrons fewer per simulation than the actual integration stage, yet runs s more simulations. The result is that the cost of the N/h update step is approximately equal to the actual integration. With more neutrons per simulation, the bias will be reduced.

Chapter 4

Small-Scale Test Problem

In order to analyze the properties of the algorithms discussed in the last few chapters, a test problem of some sort is needed. Ideally, it would replicate depletion quite well, have a deterministic mode, and have a stochastic mode. Then, numeric and statistic convergence can be tested simultaneously on the exact same problem. This is where the facsimile pin model comes in.

The facsimile pin model is a single fuel pin with one depletion zone and a simplified decay chain. The difference is that the transport operator has been replaced by a curve fit. In replacing the transport operator with a curve fit, very rapid, detailed analysis can be performed. This curve fit has two components. The first is the mean of the reaction rates (n, γ) for ^{135}Xe and ^{235}U and fission for ^{235}U. These were chosen due to the strong interaction between the reaction rates of both nuclides. For this model, these reaction rates were considered only as a function of ^{135}Xe concentration to further simplify the model. The second contains the covariance matrix of these reaction rates. The covariance matrix indicates how each component of the reaction rates vary with one another statistically. As such, both deterministic and stochastic analysis can be performed on the exact same problem.

The simplified decay chain takes the form presented in Figure 4-1. This 11 nuclide chain (including ^{235}U which fissions to yield these nuclides) was chosen to provide a sufficient approximation to the production of ^{135}Xe during depletion. There are several decay paths that also lead to removal from the system.

The process used to generate these curve fits, as well as a thorough description of the decay chain, is presented in Section 4.1. Then, this model is tested with a variety of fixed time step numerical integrators in Section 4.2. Finally, this model is integrated with adaptive time stepping schemes in Section 4.3.

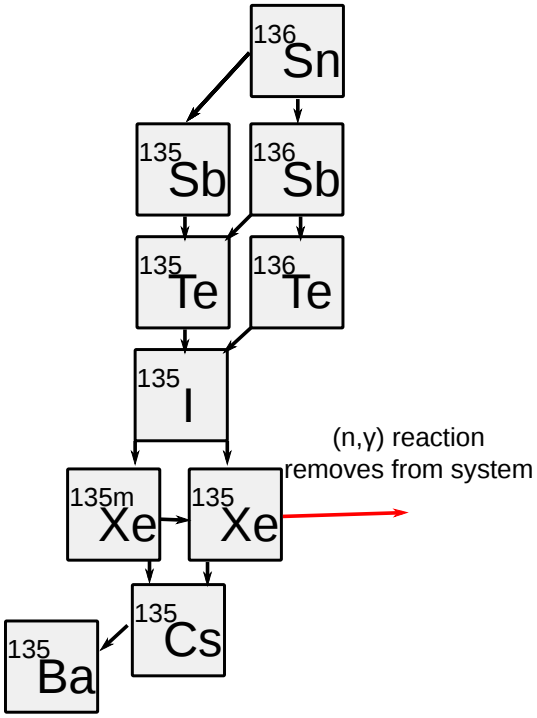


Figure 4-1: Decay chain for the facsimile pin problem

4.1 The Facsimile Pin Derivation

In order to form a test problem that imitates depletion well without too much complexity, a single pincell was considered [32]. The precise model is that of `pincell/build-xml.py` from the OpenMC examples [42], with some modifications. All materials and geometry were kept identical except for the fuel material, in which the nuclide concentrations in Table 4.1 were used instead.

For each of the 40 different concentrations of ^{135}Xe, OpenMC was run with 25 different random sequences with 100 batches (90 active) of 100 neutrons. While this is quite a small number, the accuracy of the OpenMC simulation is irrelevant for the rest of the model. For each simulation, the fission and (n, γ) reaction rates were tallied for ^{235}U and ^{135}Xe. With these 25 samples, the mean and covariance matrix were formed.

Now that there are 40 different logarithmically spaced means and covariances as a function of ^{135}Xe concentration, the goal is to curve fit these results. The mean, μ, is relatively trivial. It was fit with a 9-term polynomial on a log-log basis. The covariance, Σ, is a bit more complicated, as the curve fit matrix must itself still be symmetric positive semidefinite. Each matrix was eigendecomposed. The eigenvalues were then log-log curve fit with a 9-term polynomial, which guarantees positivity. Although the eigenvector matrix has 9 indexes, it is orthonormal. As such, with only the [0, 0], [0, 1] and [1, 0]

Nuclide	Atoms / cm^3
^{234}U	$4.484\,597\,159\,19 \times 10^{18}$
^{235}U	$5.581\,825\,646\,7 \times 10^{20}$
^{238}U	$2.240\,829\,511\,35 \times 10^{22}$
^{16}O	$4.592\,451\,256\,13 \times 10^{22}$
^{17}O	$1.741\,198\,940\,47 \times 10^{19}$
^{135}Xe	10 to 10^{18} in 40 logarithmically spaced values

Table 4.1: Facsimile Pin Fuel Materials

Nuc.	Ind.	^{235}U γ	$T_{1/2}$, s	Branch Ratio	Target 1	Target 2
^{235}U	0	—	—	—	—	—
^{136}Sn	1	$1.569\,89 \times 10^{-7}$	2.5000×10^{-1}	6×10^{-5}	^{136}Sb	^{135}Sb
^{135}Sb	2	$1.451\,37 \times 10^{-3}$	1.6790	7.8×10^{-1}	^{135}Te	—
^{136}Sb	3	$1.148\,92 \times 10^{-4}$	9.2300×10^{-1}	5.231×10^{-1}	^{136}Te	^{135}Te
^{135}Te	4	$3.216\,18 \times 10^{-2}$	1.9000×10^{1}	1.0	^{135}I	—
^{136}Te	5	$1.317\,33 \times 10^{-2}$	1.7500×10^{1}	9.869×10^{-1}	—	^{135}I
^{135}I	6	$2.927\,37 \times 10^{-2}$	2.3652×10^{4}	$8.349\,11 \times 10^{-1}$	^{135}Xe	^{135m}Xe
^{135m}Xe	7	$1.781\,22 \times 10^{-3}$	9.1740×10^{2}	3×10^{-3}	^{135}Cs	^{135}Xe
^{135}Xe	8	$7.851\,25 \times 10^{-4}$	3.2904×10^{4}	1.0	^{135}Cs	—
^{135}Cs	9	$2.454\,83 \times 10^{-6}$	7.2582×10^{13}	1.0	^{135}Ba	—
^{135}Ba	10	$3.819\,73 \times 10^{-10}$	—	—	—	—

Table 4.2: Facsimile Pin Nuclide Data

indices of the matrix, all other terms can be reconstructed. These three entries were then curve fit with a log-linear polynomial as well. The complete set of equations used to reconstruct the reaction rates are presented in Section B.1.

With curve fits for both μ and Σ as a function of ^{135}Xe density, X, the unnormalized reaction rates can be sampled from the multivariate normal distribution $\hat{R} = \mathcal{N}(\mu(X), \sigma(X))$. Alternatively, if only μ is used, the problem can be integrated without stochastic noise.

To extend this to a depletion problem, a simple decay chain was made. The nuclides used are listed in Table 4.1. Here, all data is from ENDF/B-VII.1 [14]. All nuclides that do not appear inside of this table are ignored and decay to these nuclides is treated as annihilation. For the branching ratio, "Target 1" is generated proportional to the branching ratio, and "Target 2" is generated proportional to (1 − branching ratio).

The final piece of the puzzle is the power with which to normalize the reaction rates. The obvious first choice is a constant power, here shown as Equation (4.1). This represents an average geometry power of 166 W, which

corresponds to roughly 343.5 W/cm^3 of fuel. This choice yields an exceptionally easy problem to the point where some analysis becomes uninformative. As such, this will only be used to compare first order methods. For the higher order methods, the power is replaced by the form in Equation (4.2). This cosine-shaped power profile has a period of 2 days.

$$P_{\text{const}}(t) = 166 \tag{4.1}$$

$$P_{\cos}(t) = 166\left[1 + \cos\left(\frac{\pi t}{86400}\right)\right] \tag{4.2}$$

All power in this facsimile problem is generated solely via ^{235}U fission, in which 193.41 MeV of energy per fission is recoverable. All other fissions are ignored. The reaction rates, generated from the curve fits earlier, are normalized as follows:

$$\hat{P}(y,t) = 3.099 \times 10^{-11} \hat{R}_f^{235} \left(\frac{y_8(t)}{V}\right) y_0(t)$$

$$R(y,t) = \hat{R}\left(\frac{y_8(t)}{V}\right) \frac{P(t)}{\hat{P}(y,t)}$$

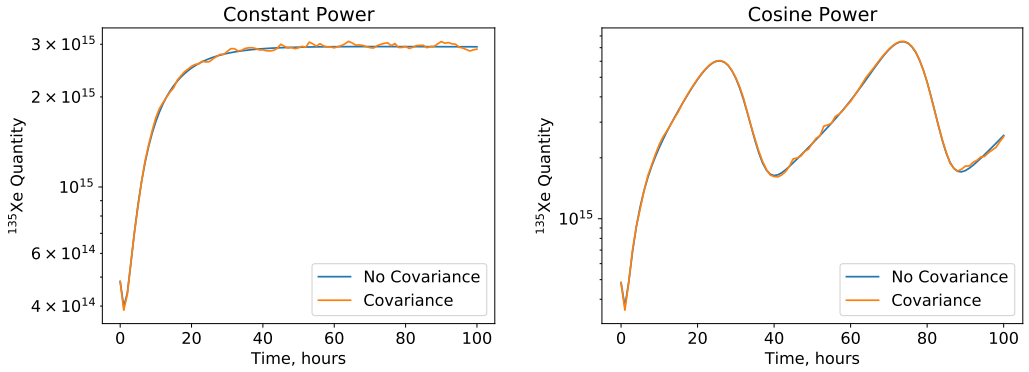
Here, y is the vector of total nuclide count, sequenced as in Table 4.1, and V is the volume of the geometry, which is $\pi(0.39218)^2$ cm^3. y_8 is the total quantity of ^{135}Xe atoms and y_0 is the total number of ^{235}U.

For clarity, all equations are listed in Section B.2, along with initial conditions and reference solutions generated with arbitrary precision arithmetic. All non-reference simulations were performed in double precision arithmetic. The matrix exponential used was the one built-in to Julia [4], which is a scaling-and-squaring type algorithm with a Padé inner exponential [25]. Each problem is integrated to 100 hours.

Figure 4-2 shows the trajectory for both the constant and cosine power versions of the model with and without stochastic noise. The constant power model rapidly approaches steady state as theory would predict. The cosine power has large scale oscillations that make it more challenging to numerically integrate. When the reaction rates are sampled from a distribution, noise is evident along the trajectory. The dip at the beginning of each plot is due to the ^{135}I initial condition being zero.

4.2 The Facsimile Pin Convergence Results

Using the test problem, the algorithms presented in both Chapter 2 and Chapter 3 are tested both numerically and statistically. As there are so many algorithms to test, they will be batched into four categories: first order methods,



(a) With a constant power

(b) With a cosine power

Figure 4-2: The evolution of ^{135}Xe with time in the facsimile pin problem

second order methods, third order methods, and higher order methods. For each algorithm, four tests will be performed with the goal to identify relative accuracy and statistical performance.

Test 1: This is a purely numerical test. No sampling is performed and $A(y,t)$ is constructed using the mean, μ, of the reaction rates. Time steps spanned 10^{-3} hours to 20 hours. The relative error at 100 hours in ^{135}Xe number is then plotted. The goal is to see how close each method gets to its theoretical convergence order.

Test 2: This test is a repeat of the first, except $A(y,t)$ is constructed using samples of $\mathcal{N}(\mu,\Sigma)$. In this mode, every integration samples the relative error of ^{135}Xe from an unknown distribution. This distribution was repeatedly sampled until the relative standard deviation of the mean was less than 0.01. Another consideration is that some algorithms require more than one evaluation of A. To keep the effective number of neutrons constant per time step, Σ was multiplied by n, where n is the number of function evaluations per time step. This is equivalent to scaling the number of neutrons by $1/n$. The goal of this test is to see how temporal convergence is affected by statistics.

Test 3: In the third test, a fixed 0.02 hour time step was used and Σ was scaled to mimic increasing the number of neutrons per simulation. In typical Monte Carlo simulations, the covariance Σ is inversely proportional to the number of neutrons. As such, doubling the number of neutrons will halve Σ. The goal of this test is to see how improvements in reaction rate statistics impact the end result statistics.

Test 4: The fourth test is then designed to see how error changes when a fixed neutronics cost is imposed. As such, when the number of time steps is doubled, the number of neutrons per simulation is halved. The exact formula for Σ (including the number of substeps, n) is given by Equation (4.3). The

factor 10^4 is chosen to make sure that Σ does not become too large. Since the reaction rates are described by a multivariate normal, a large Σ would lead to the possibility of negative reaction rates.

$$\Sigma_{\text{real}} = \frac{nN_{\text{steps}}\Sigma}{10^4} \tag{4.3}$$

There are two interesting properties that can be estimated from the above plots as well. The first is the approximate convergence order of the method. The convergence order can be approximated by fitting Equation (4.4) (where ϵ is the relative error) using ordinary least-squares in a log-log fashion. The value s then approximates the convergence order. If ϵ is stochastically distributed, 10^4 samples are taken and the distribution of s computed.

$$\epsilon = Bh^s \tag{4.4}$$

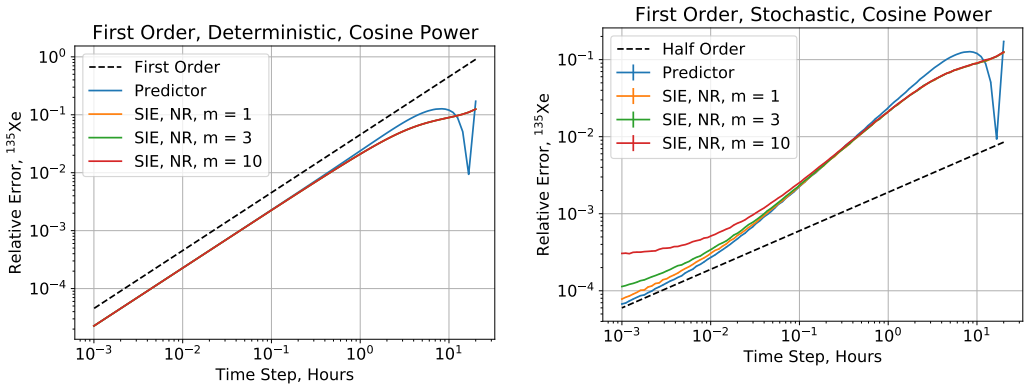
The second is the stochastic figure of merit. In the stochastic regime, ϵ converges roughly $\mathcal{O}(h^{1/2})$. However, the leading coefficient varies between methods. This figure of merit, C, can be estimated by fitting Equation (4.5) using Levenberg-Marquardt. The ratio C_a/C_b between two methods is very closely related to the estimated ratio of standard deviations σ_a/σ_b from Section 3.5 in that in the stochastic region, errors are proportional to σ. In a ratio, these proportionalities cancel. As such, the values can be directly compared. The values estimated using the technique from Section 3.5 will be denoted as "Theoretical C_a/C_b."

$$\epsilon = C\sqrt{h} \tag{4.5}$$

4.2.1 First Order Methods

The first order family of methods includes the predictor method (Section 2.1.1) as well as the stochastic implicit Euler method (SIE, Section 2.1.2.3). Predictor has a single run mode. SIE however has the option to vary the number of substeps to further converge the implicit portion of the algorithm. In addition, there is the nuclide (NR) and reaction rate (RR) relaxation modes, of which only nuclide relaxation was tested initially. For this test, the number of substeps, m, was set at 1, 3, and 10.

These algorithms were first tested with the $P_{\cos}$ power distribution. The plot of relative error against the number of time steps for both the deterministic and stochastic test are shown in Figure 4-3. Unfortunately, the $m = 3$ and $m = 10$ results appear to be converging to the wrong solution for very short time steps, indicating bias effects. This prevents the analysis of the stochastic



(a) Test 1, deterministic

(b) Test 2, stochastic, fixed neutrons / A evaluation

Figure 4-3: Convergence of first order methods, cosine power problem

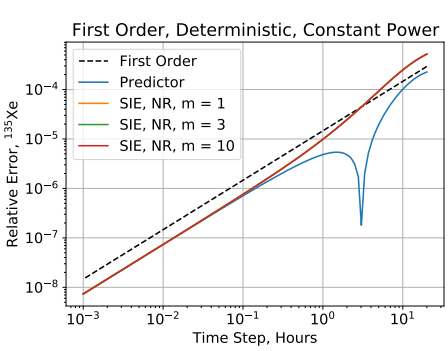
convergence.

To continue analysis, the test problem was switched to constant power. This much easier problem yields the results shown in Figure 4-4. Starting with the deterministic results, the asymptotic region appears to be converging first order, as theory would predict. To further verify this, the deterministic results were curve fit over the domain $h \in [10^{-3}, 10^{-2}]$, with the estimated order listed in Table 4.3. The fit order is extremely close to the mathematical order of the method, as would be expected. Additionally, the relative error in the final point is nearly identical between methods. Only in the non-asymptotic regime do the methods appreciably differ.

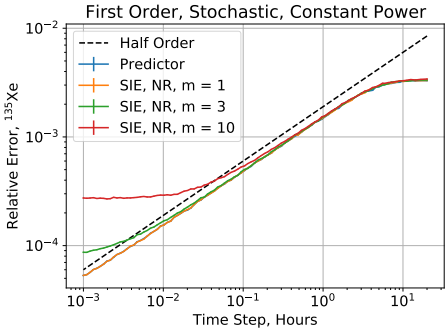
Continuing onward to the stochastic results, a rather interesting phenomenon appears. While convergence is initially proportional to half-order, the high m SIE methods cease converging. This indicates a statistical bias effect of some form, perhaps induced by the nonlinear transformation through the matrix exponential of the very large variance used for sampling. Both the relative figure of merit and convergence order were curve fit over the domain $h \in [0.3, 2.5]$, to avoid the bias region. The results are tabulated in Table 4.4. In summary, the methods are converging at roughly half-order over this domain, and the relative figure of merit is very close to unity. These values very closely correspond to the predicted figure of merit derived in Section 3.5.

The next component considered is that of the fixed time step, variable variance. The results are plotted in Figure 4-4c. At very large number of neutrons, all methods coincide. On the other end, the $m = 3, 10$ methods both have biases that appear as the variance increases.

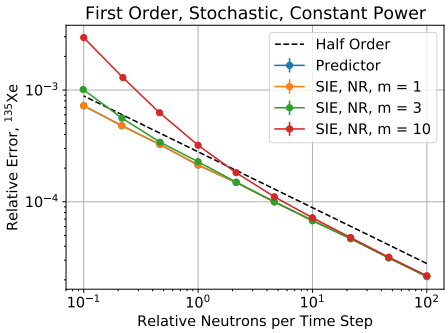
Then, the constant cost data is plotted. As one would anticipate from Figure 4-4b and Figure 4-4c, the decreasing the number of neutrons proportionally



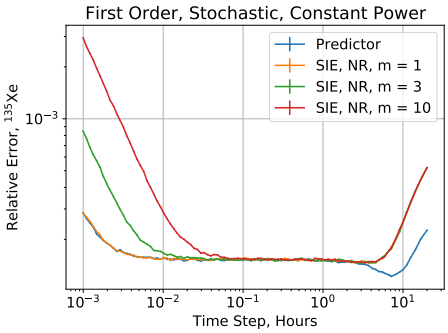
(a) Test 1, deterministic



(b) Test 2, stochastic, fixed neutrons/A evaluation



(c) Test 3, stochastic, convergence with neutrons



(d) Test 4, stochastic, fixed neutrons/simulation

Figure 4-4: Convergence of first order methods, constant power problem

Method	Order (Fit)	Relerr, $h = 0.001$
Predictor	0.99975	7.318×10^{-9}
SIE, $m = 1$	1.00025	7.356×10^{-9}
SIE, $m = 3$	1.00024	7.356×10^{-9}
SIE, $m = 10$	1.00025	7.355×10^{-9}

Table 4.3: Deterministic performance of the first order set, constant power

Method	Order (Fit)	C/C_{pred}	Theoretical C/C_{pred}
Predictor	0.48400 ± 0.00003	—	—
SIE, $m = 1$	0.47982 ± 0.00003	0.998529 ± 0.000004	1.0
SIE, $m = 3$	0.48277 ± 0.00003	1.009650 ± 0.000004	1.0
SIE, $m = 10$	0.47029 ± 0.00003	1.020940 ± 0.000005	1.0

Table 4.4: Statistical performance of the first order set, constant power

to the increase in time steps yields a cancellation of error. A large basin appears in which accuracy is entirely insensitive to choice of time step. As time step shrinks, however, bias effects begin to appear for all algorithms. At $h = 10^{-3}$, the $m = 10$ SIE has a bias an order of magnitude larger than $m = 1$.

Finally, it is worth briefly comparing the reaction rate relaxed and nuclide relaxed methods, as the different relaxation techniques might reduce the rather extreme bias effects seen with nuclide relaxation. The fixed neutrons / simulation run was repeated with reaction rate relaxation. The results are shown in Figure 4-5. Surprisingly, the results are nearly identical.

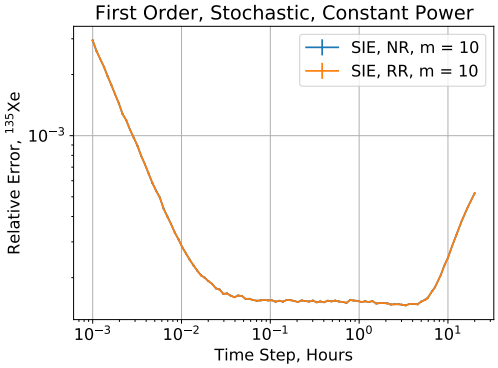


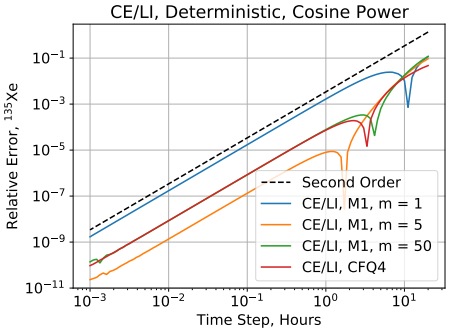
Figure 4-5: Comparison of nuclide and reaction rate relaxation

4.2.2 Second Order Methods

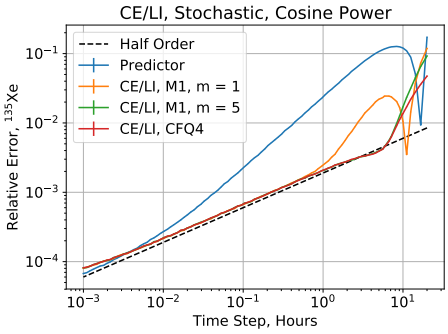
The second order family is a bit larger than the first order family. In this set are the CE/CM method (Section 2.1.2), CE/LI (Section 2.1.2.1), and two EPC methods (Section 3.3). In the case of CE/LI, there are a wide variety of ways to perform the second integration.

To keep things manageable, CE/LI will be considered separately. In the following plots, "M1" is the first order substepping scheme discussed in Section 2.2.1, where m is the number of substeps. "CFQ4" is the fourth order commutator-free integrator discussed in the same section. The results are plotted in Figure 4-6, with deterministic performance values listed in Table 4.5 and stochstic performance values listed in Table 4.6. Fits were performed over $h \in [0.02, 0.2]$ for all but predictor, which was performed over $h \in [0.001, 0.002]$ to avoid numerical convergence issues. All simulations were performed with cosine power.

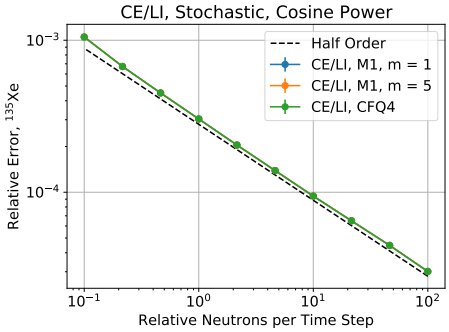
For the most part, these algorithms behave as anticipated. Fitted deterministic convergence is approximately second order and the relative statistical performance is almost equivalent to the theoretical values. The most important result is that the choice of secondary integrator has a major impact on the



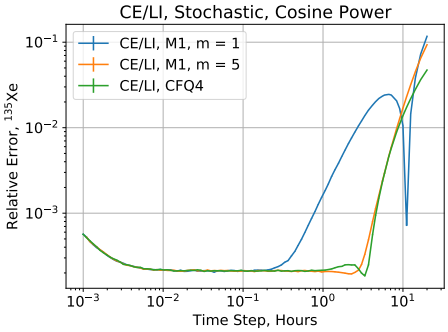
(a) Test 1, deterministic



(b) Test 2, stochastic, fixed neutrons per A evaluation



(c) Test 3, stochastic, convergence with neutrons



(d) Test 4, stochastic, fixed neutrons per simulation

Figure 4-6: Convergence of CE/LI, cosine power problem

Method	Order (Fit)	Relerr, $h = 0.0102$
CE/LI, M1, $m = 1$	1.99989	1.778×10^{-7}
CE/LI, M1, $m = 5$	1.98505	1.415×10^{-9}
CE/LI, M1, $m = 50$	1.99747	8.805×10^{-9}
CE/LI, CFQ4	1.99689	8.878×10^{-9}

Table 4.5: Deterministic performance of CE/LI, cosine power

Method	Order (Fit)	C/C_{pred}	Theoretical C/C_{pred}
CE/LI, M1, $m = 1$	0.49243 ± 0.00003	1.018 ± 0.016	1.0
CE/LI, M1, $m = 5$	0.49211 ± 0.00003	1.021 ± 0.016	1.0
CE/LI, CFQ4	0.49317 ± 0.00003	1.022 ± 0.016	1.0

Table 4.6: Statistical performance of CE/LI, cosine power

results. Converting from $m = 1$ to $m = 5$ reduces asymptotic errors by over two orders of magnitude. Continuing onward to $m = 50$ increases the error again, which might be due to error cancellation. The CFQ4 algorithm matches the performance of $m = 50$, despite using a factor of 25 fewer matrix exponentials. In the stochastic case, however, $m = 5$ and CFQ4 end up extremely close in capability.

Returning to the general second order family, similar results are plotted in Figure 4-7, with coefficients tabulated in Table 4.7 and Table 4.8 for deterministic and stochastic results respectively. In this comparison, the non-CE/LI algorithms perform nearly identically to one another numerically. Statistically, the CE/LI algorithm is best, followed by EPC-RK4, EPC-RK45, and then CE/CM. Bias increases with increasing number of substeps, as was indicated with the SIE algorithms earlier.

For all second order methods, the CE/LI method (optimally with M1, $m \geq 5$ or CFQ4) has the best performance. It has the best statistical performance, closely approaching that of predictor, while massively outperforming predictor numerically. In the fixed neutrons per simulation results, the CE/LI algorithms have the widest and deepest optimal basins. This means that they are less sensitive to choice of time step and the optimal time step will yield superior results per neutron.

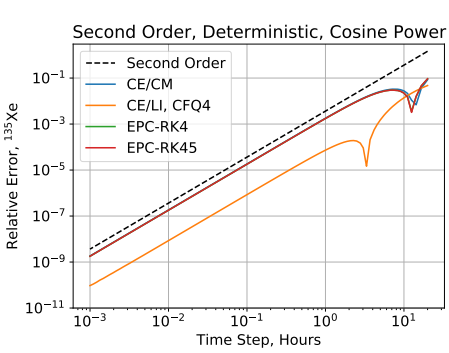
4.2.3 Third Order Methods

The third order family contains the LE/QI methods (Section 2.1.2.2), the EL3 method (Section 3.4), and the CF3 method (Section 2.2.3). Similar to CE/LI, LE/QI has two non-y dependent ODEs to solve, and as such there is a variety of choice when it comes to integration. These options will be considered first.

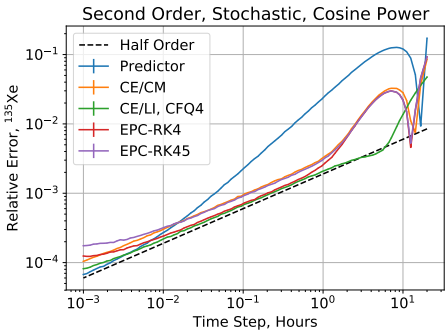
The LE/QI results are plotted in Figure 4-8, with deterministic results tabulated in Table 4.9 and stochastic results tabulated in Table 4.10. The curve fits are performed over the same domains used in the second order methods. These results are extremely similar to the previous results, except with regard to order. As mentioned in Section 3.1, the choice of secondary integrators can impact the order of the algorithm. M1 is second order, and this is evident. As m increases, the order approaches that of third order, as the asymptotic second order regime is pushed to smaller h. Only the CFQ4 secondary integrator achieves true third order.

Statistically, the curve fit relative figure of merit outperforms the theoretical value by quite a bit. This may be owed to a few approximations made in the derivation of the theoretical value, the most significant being that there is no covariance between simulations. The reuse of A prevents this from being even approximately accurate.

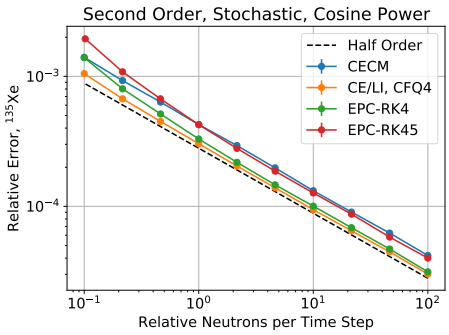
The rest of the third order family is plotted in Figure 4-9, with tabulated



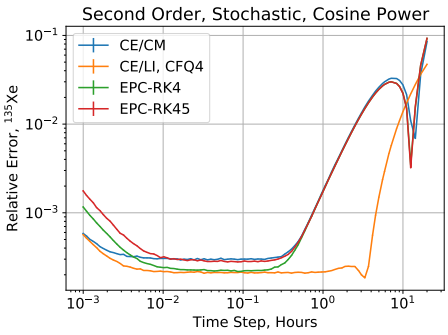
(a) Test 1, deterministic



(b) Test 2, stochastic, fixed neutrons/A evaluation



(c) Test 3, stochastic, convergence with neutrons



(d) Test 4, stochastic, fixed neutrons/simulation

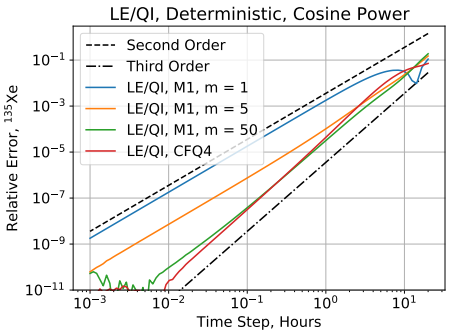
Figure 4-7: Convergence of Second Order Methods, Cosine Power Problem

Method	Order (Fit)	Relerr, $h = 0.0102$
CE/CM	1.99987	1.908×10^{-7}
CE/LI, CFQ4	1.99689	8.878×10^{-9}
EPC-RK4	1.99981	1.867×10^{-7}
EPC-RK45	1.99981	1.867×10^{-7}

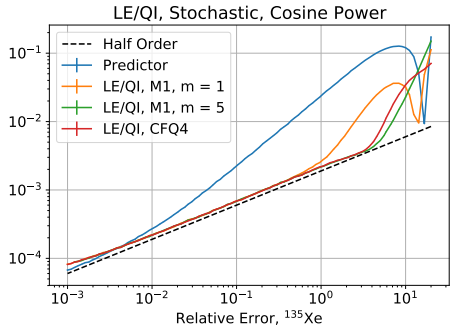
Table 4.7: Deterministic Performance of the Second Order Set, Cosine Power

Method	Slope	C/C_{pred}	Theoretical C/C_{pred}
CE/CM	0.49297 ± 0.00003	1.439 ± 0.022	1.41421
CE/LI, CFQ4	0.49311 ± 0.00003	1.022 ± 0.016	1.0
EPC-RK4	0.47903 ± 0.00003	1.079 ± 0.017	1.05409
EPC-RK45	0.48129 ± 0.00003	1.385 ± 0.022	1.34055

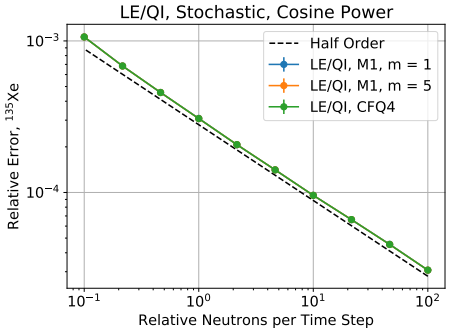
Table 4.8: Statistical Performance of the Second Order Set, Cosine Power



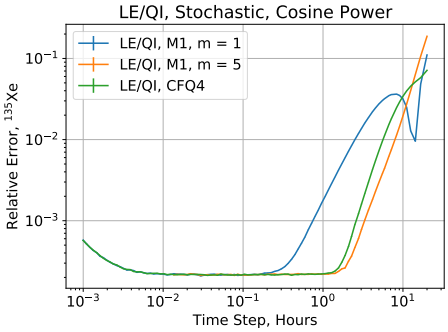
(a) Test 1, deterministic



(b) Test 2, stochastic, fixed neutrons per A evaluation



(c) Test 3, stochastic, convergence with neutrons



(d) Test 4, stochastic, fixed neutrons per simulation

Figure 4-8: Convergence of LE/QI, Cosine Power Problem

Method	Order (Fit)	Relerr, $h = 0.104$
LE/QI, M1 $m = 1$	2.00100	1.944×10^{-5}
LE/QI, M1 $m = 5$	2.02913	8.112×10^{-7}
LE/QI, M1 $m = 50$	2.72058	4.257×10^{-8}
LE/QI, CFQ4	3.05897	3.556×10^{-8}

Table 4.9: Deterministic Performance of LE/QI, Cosine Power

Method	Order (Fit)	C/C_{pred}	Theoretical C/C_{pred}
LE/QI, M1 $m = 1$	0.49784 ± 0.00003	1.043 ± 0.016	1.118
LE/QI, M1 $m = 5$	0.50089 ± 0.00003	1.042 ± 0.016	1.118
LE/QI, CFQ4	0.50079 ± 0.00003	1.039 ± 0.016	1.118

Table 4.10: Statistical Performance of LE/QI, Cosine Power

values in Table 4.11 and Table 4.12. Overall, the ranking of methods is LE/QI, EL3, and CF3 for both numeric and stochastic performance. While LE/QI holds a strong lead in the third order method family, it is still outperformed by CE/LI CFQ4 on a stochastic problem of fixed cost.

4.2.4 Higher Order Methods

Finally, the high order methods are considered. This set includes EL4 (Section 3.4), CF4 (Section 2.2.3), RKMK-4, and RKMK-DOPRI5 (2.2.2). The results are shown in Figure 4-10, with tabulated values in Table 4.13 and Table 4.14.

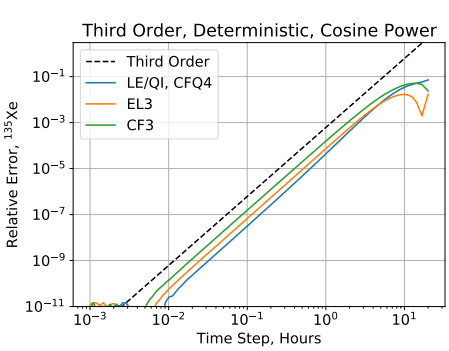
These algorithms begin to show some stability flaws, with the majority failing due to negative/zero concentrations which can lead to a divide-by-zero or taking the logarithm of a negative value. This causes the simulation to crash. As such, not all methods have solutions up to $h = 20$ hours. The oscillatory nature of the cosine-power problem prevents a clean analysis of the order of these methods, as the error itself becomes oscillatory. Overall, the CF4 algorithm has the most reliable performance. It did not have stability issues and its statistical performance is decent.

4.3 Adaptive Time Stepping on the Facsimile Pin

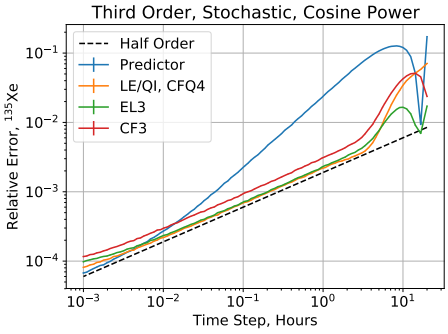
The development of an adaptive time stepping (ATS) scheme for a stochastic problem is a complicated one. Several properties of the ATS algorithm need to be known to derive an effective time step, such as how the time step affects the stochastic uncertainty of the local and global error. This will be investigated in Section 4.3.1. This information was used in Section 3.6.2 to derive several schemes. Some of these schemes have hyperparameters that need to be optimized to perform effectively. This optimization is performed in Section 4.3.2. Finally, the relative capabilities of all the ATS algorithms will be tested on the facsimile problem in Section 4.3.3.

4.3.1 Properties of Distributions During Time Integration

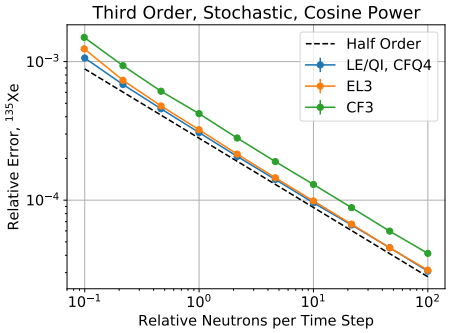
In order to make an effective adaptive time stepping scheme, it is important to first investigate how the error estimate interacts numerically and stochastically with the problem. As such, a single time step for the facsimile problem was run 500 times with varying h and relative number of neutrons with the EL3



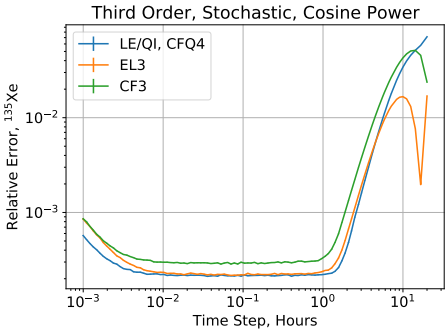
(a) Test 1, deterministic



(b) Test 2, stochastic, fixed neutrons/A evaluation



(c) Test 3, stochastic, convergence with neutrons



(d) Test 4, stochastic, fixed neutrons/simulation

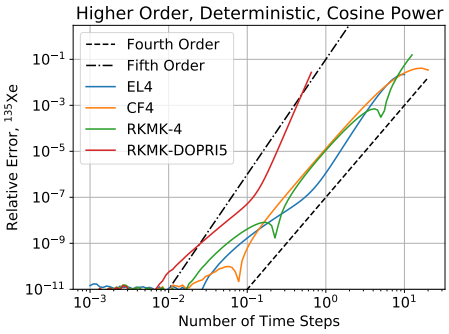
Figure 4-9: Convergence of third order methods, cosine power problem

Method	Order (Fit)	Relerr, $h = 0.104$
LE/QI, CFQ4	3.05897	3.556×10^{-8}
EL3	3.03360	7.603×10^{-8}
CF3	3.02302	1.739×10^{-7}

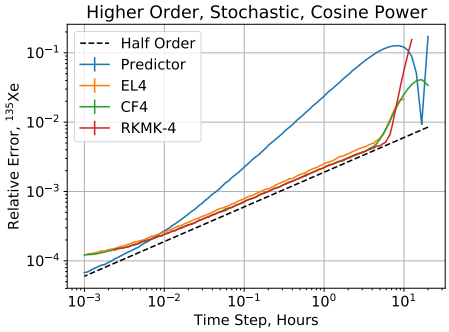
Table 4.11: Deterministic performance of the third order set, cosine power

Method	Order (Fit)	C/C_{pred}	Theoretical C/C_{pred}
LE/QI, CFQ4	0.50079 ± 0.00003	1.039 ± 0.016	1.118
EL3	0.49369 ± 0.00003	1.073 ± 0.017	1.032
CF3	0.49832 ± 0.00003	1.414 ± 0.022	1.369

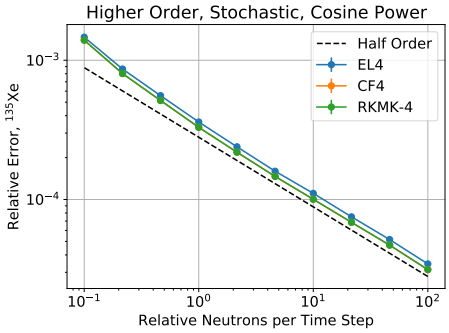
Table 4.12: Statistical performance of the third order set, cosine power



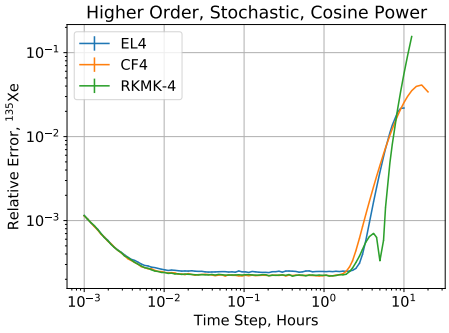
(a) Test 1, deterministic



(b) Test 2, stochastic, fixed neutrons/A evaluation



(c) Test 3, stochastic, convergence with neutrons



(d) Test 4, stochastic, fixed neutrons/simulation

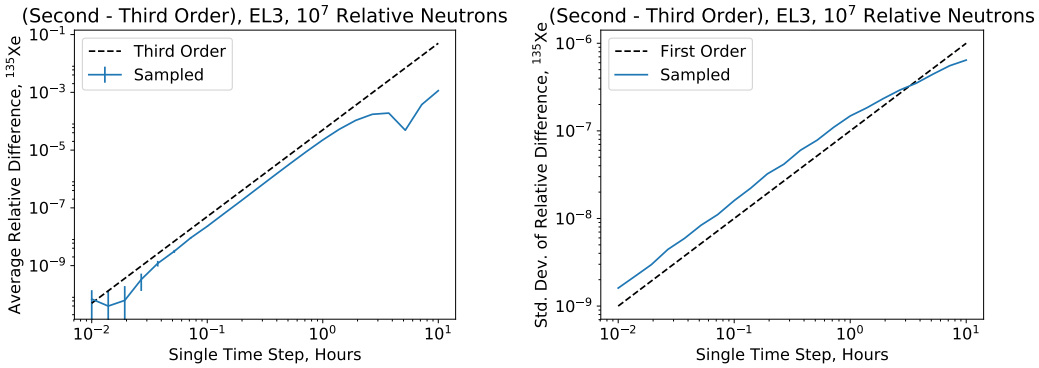
Figure 4-10: Convergence of higher order methods, cosine power problem

Method	Order (Fit)	Relerr, $h = 0.104$
EL4	3.68980	1.876×10^{-9}
CF4	3.99179	7.876×10^{-10}
RKMK-4	3.45626	5.944×10^{-8}
RKMK-DOPRI5	4.89031	5.944×10^{-8}

Table 4.13: Deterministic performance of the higher order set, cosine power

Method	Order (Fit)	C/C_{pred}	Theoretical C/C_{pred}
EL4	0.48892 ± 0.00003	1.200 ± 0.019	1.120
CF4	0.48237 ± 0.00003	1.086 ± 0.017	1.054
RKMK-4	0.48930 ± 0.00003	1.085 ± 0.017	1.054

Table 4.14: Statistical performance of the higher order set, cosine power



(a) Mean of Relative Error (b) Standard Deviation of Relative Error

Figure 4-11: Fixed N, variable h comparison of error between second and third order estimates, EL3, constant power

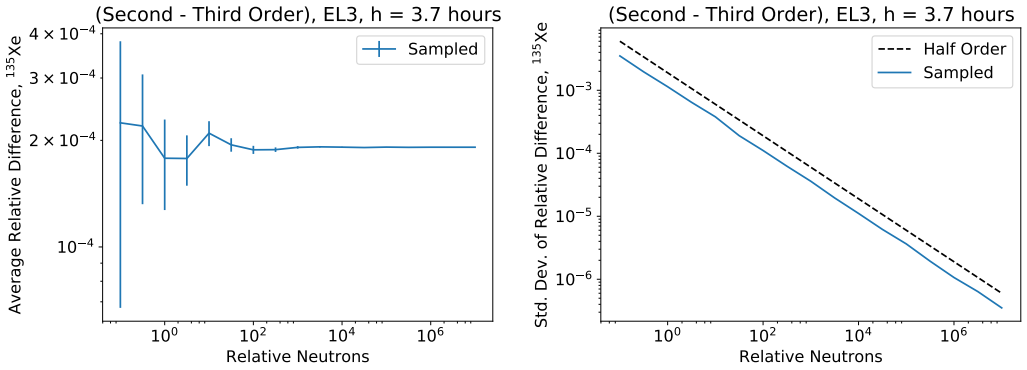
algorithm from Section 3.4. The second order error estimate used was the non-FSAL algorithm.

From the derivation of the EL3 algorithm, the difference between the third order estimate and the second order estimate should be $\mathcal{O}(h^3)$. Ideally, it should be $\mathcal{O}(h^3)$ over as much of the domain as possible, as that allows for a very high quality estimate of the next time step. In Figure 4-11a, this is tested with a very large number of neutrons (Σ is $1/10^7$ that of the nominal value). Over the majority of the domain, the difference between the two distributions is third order as theory would predict. It only falters for longer time steps.

More interesting is the standard deviation as a function of time step, as shown in Figure 4-11b. In this plot, the single-step standard deviation growth is linear with time step. In results from the prior section, the global statistical uncertainty is $\mathcal{O}(\sqrt{h})$ instead of $\mathcal{O}(h)$. This is not surprising. Take for example the function $\prod x_i$, where all $x_i = \mathcal{N}(1, \sigma^2)$ and are independently sampled. Using the first order variance formula, the standard deviation of this product is given by Equation (4.6).

$$\begin{aligned} f &= \prod_{i=1}^{n} x_i \\ \sigma_f &= \sqrt{\sum_{i=1}^{n} \sigma^2 \frac{\partial f}{\partial x_i}} \\ \sigma_f &= \sqrt{n}\sigma \end{aligned} \tag{4.6}$$

The result is that if one doubles n and halves σ (as would occur in halving h), the result is a decrease of σ by only $\sqrt{2}$.



(a) Mean of Relative Error (b) Standard Deviation of Relative Error

Figure 4-12: Fixed h, variable N comparison of error between second and third order estimates, EL3, constant power

It is important to note that this is only true outside of equilibrium. In the cosine power model, ^{135}Xe is always out of equilibrium. When in equilibrium, integrating with slight perturbations will yield errors that will not grow unbounded with time step. In addition, the errors are not multiplicative and often dampen and cancel out. As such, the uncertainty of a value in equilibrium will be insensitive to time step.

The next consideration is the fixed time step, variable N results, which are plotted in Figure 4-12. The results here are expected. The average relative error is insensitive to statistics until the uncertainty becomes large enough to dominate the solution. The standard deviation itself is $\mathcal{O}(1/\sqrt{N})$, which is also anticipated.

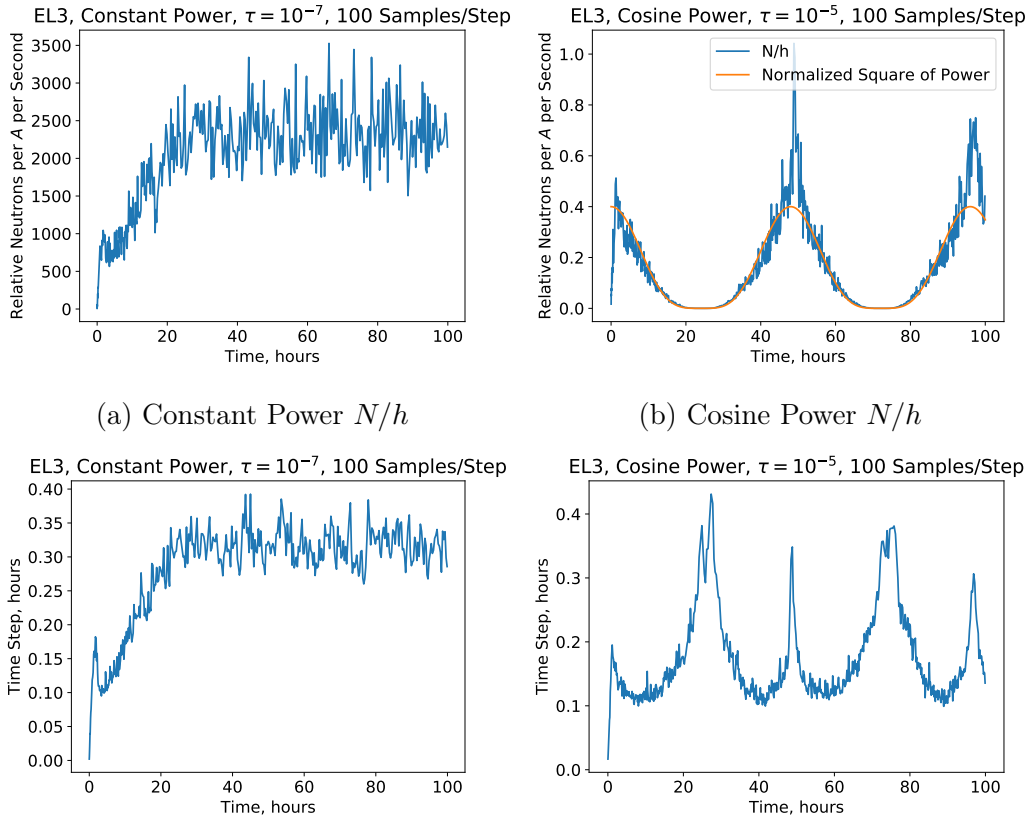
Overall, the result is that for a single step, the standard deviation and the mean of the local error estimate for EL3 are governed by:

$$|\mu| \propto h^3$$
$$\sigma \propto \frac{1}{\sqrt{N}} \text{ in equilibrium and } \frac{h}{\sqrt{N}} \text{ if not.}$$

However, the global error of y_{n+1} is distributed slightly differently:

$$|\mu_{\text{global}}| \propto h^3$$
$$\sigma_{\text{global}} \propto \frac{1}{\sqrt{N}} \text{ in equilibrium and } \sqrt{\frac{h}{N}} \text{ if not.}$$

The majority of nuclides of interest are not in equilibrium, so the algorithms presented in Section 3.6.2 use the non-equilibrium estimates. If equilibrium nuclides begin to dominate the problem, however, the update process for N



(a) Constant Power N/h

(b) Cosine Power N/h

(c) Constant Power h

(d) Cosine Power h

Figure 4-13: The evolution of the optimal N/h and h for fixed and cosine power distributions

and h should be able to compensate.

In order to analyze how the optimal N/h evolves with time, both the cosine power and constant power simulations were run with the algorithm described in Section 3.6.2.2. The results are plotted in Figure 4-13. Here, 100 samples were taken before computing a new N and h. In the constant power case, the N/h value slowly climbs during the initial transient. Once the ^{135}Xe concentration stabilizes, the N/h value does as well. Similarly, the time step stabilizes. The cosine power results are more interesting. The N/h value closely follows the square of the power. In addition, when the derivative of power was small, the time step increased.

4.3.2 Hyperparameters of the Multi-Sample Algorithm

The ATS scheme from Section 3.6.2.2 runs each time step multiple times to sample a distribution of time step end points. The distribution of errors between

Parameter	Value	Units
θ_h	10^{-2} to 1 in 25 logarithmically spaced points	—
θ_n	10^{-2} to 1 in 11 logarithmically spaced points	—
τ	10^{-2} to 10^{-4} in 8 logarithmically spaced points	—
# Samples per τ	5	—
h_0	60	seconds
N_0	1	Relative
$\min N_i$	0.01	Relative

Table 4.15: ATS hyperparameter parametric analysis

the high order and low order end point estimates can be leveraged to update the time step and number of neutrons per time step. However, there are two hyperparameters in the algorithm, θ_h and θ_n, which are constants used to scale the target error and standard deviation relative to the tolerance. The choice of θ_h and θ_n have a very strong impact on the resulting accuracy and performance of the algorithm.

In order to test the relative merits of each θ pair, the facsimile test problem was integrated using the coefficients listed in Table 4.15. Four parametric analyses were performed, with either cosine or constant power and with either the EL(3,2) algorithm with non-FSAL second order estimate or the LE/QI algorithm. A simulation was skipped if Equation (3.28) was not met.

As would be expected from previous analysis, the global error should be proportional to $1/\sqrt{N_t}$, where N_t is the total effective number of neutrons in the simulation. This is not always true in the real simulation. One reason could be numerical error dominating stochastic error. However, this allows for the computation of a useful figure of merit. Since N_t is proportional to simulation time and error should be proportional to $1/\sqrt{N_t}$, one can derive the figure of merit given in Equation (4.7). Here, ϵ is the relative error in ^{135}Xe. A large value indicates a lower error per unit runtime cost.

$$FOM = \frac{1}{\epsilon\sqrt{N_t}} \tag{4.7}$$

The plots in Figure 4-14 show the average FOM for this parametric study. Unfortunately, due to the intensely stochastic nature of this figure of merit computation combined with the run time cost of increasing the number of samples, these charts are quite noisy. However, a few features can be distinguished. There is one dominant band of optimal θ corresponding roughly to $\theta_h = 5\theta_n$ on all plots. Increasing θ_h and shrinking θ_n results in a reduction in figure of merit

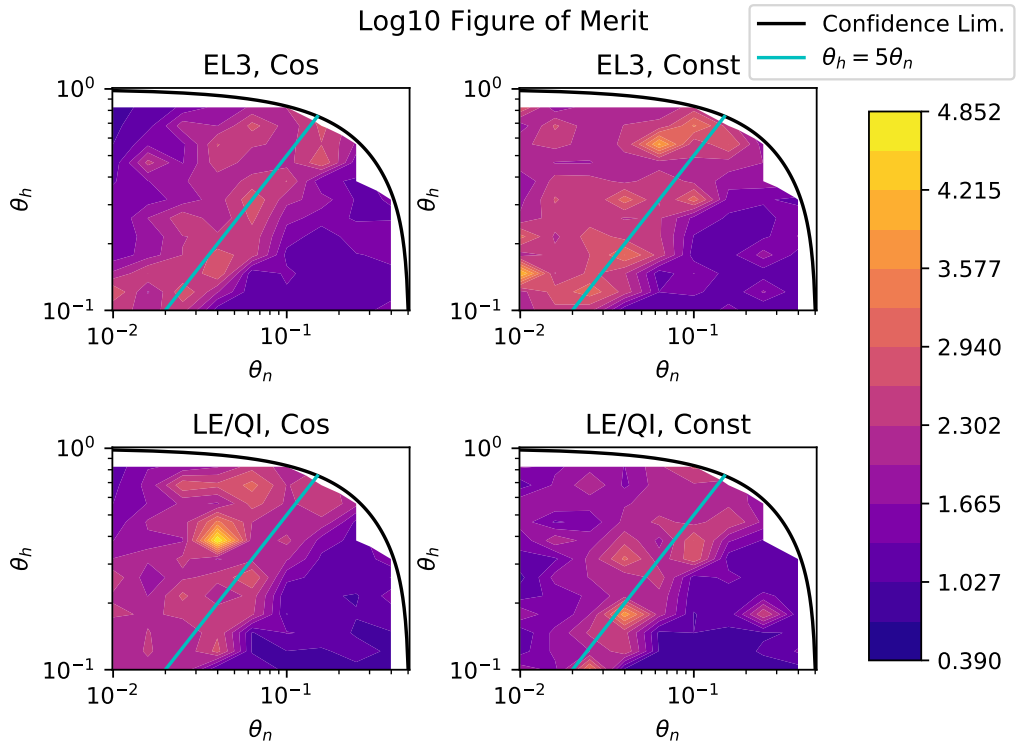


Figure 4-14: Contour plots of the log10 figure of merit for a variety of combinations of θ_h and θ_n as well as algorithms and test problems

as the numerical error dominates. Increasing θ_n and shrinking θ_h also results in a reduction in figure of merit due to the dominance of stochastic error. In addition, the easier constant power problem has a larger figure of merit, which is consistent to expectations.

4.3.3 Comparison of Adaptive Time Stepping Algorithms on the Facsimile Problem

Now that the algorithms have been developed and the necessary hyperparameters have been estimated, it is worthwhile to test their performance. Four methods will be tested. The first is the naïve sampling approach, where the neutrons per simulation is fixed. The second is to sample the distribution of errors at every step and using the average sampled y_{n+1} to continue. This process is described in Section 3.6.2.2, and will be called "Mode 1" for short. The third option is to only sample the distribution at the start to compute N/h. This is the algorithm described in Section 3.6.2.3 with sampling only performed once. It will be labeled "Mode 2" for short. The final algorithm is to update the distribution every 10 steps, and will be labeled "Mode 3." The

Parameter	Value	Units
θ_h	0.5	—
θ_n	0.1	—
τ	`logspace(-2, -5, 25)`	—
# Samples per τ	50	—
h_0	60	seconds
N_0	1	Relative
min N_i	0.01	Relative

Table 4.16: ATS Parameter Set

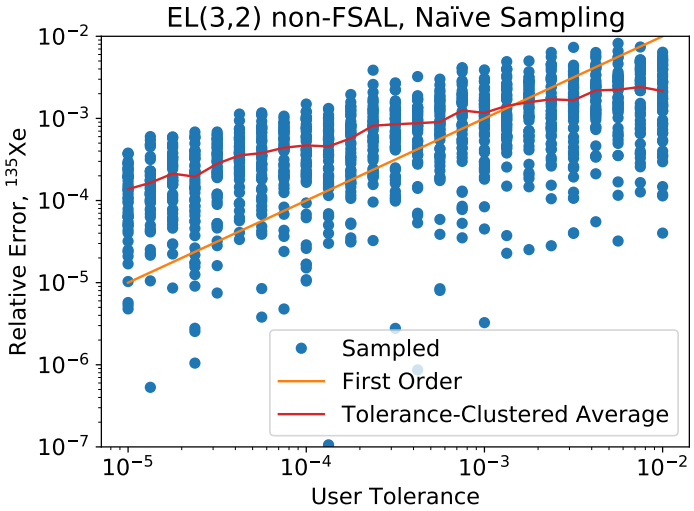


Figure 4-15: Naïve sampling, EL3, error at 100 hours as a function of user-defined tolerance

parameter set used during simulation is given in Table 4.16. The value used to compute μ and σ is the relative difference of ^{135}Xe between y_{n+1} and $\hat{y}_{n+1}$.

Starting with naïve sampling, the relative error in ^{135}Xe at the end of simulation as compared to the user-defined tolerance is plotted in Figure 4-15. This demonstrates the primary failing of this method in that while decreasing tolerance decreases error, it does not do it in a one-to-one way, as should be the case. In theory, with a sufficiently high number of neutrons, this method will work. However, this quantity is not known a priori, and prevents naïve sampling from being useful.

This failing does not occur in the other three methods. The Mode 1 results are shown in Figure 4-16. For this particular algorithm, all parameters vary as would be anticipated. The error is linear to the user-defined tolerance,

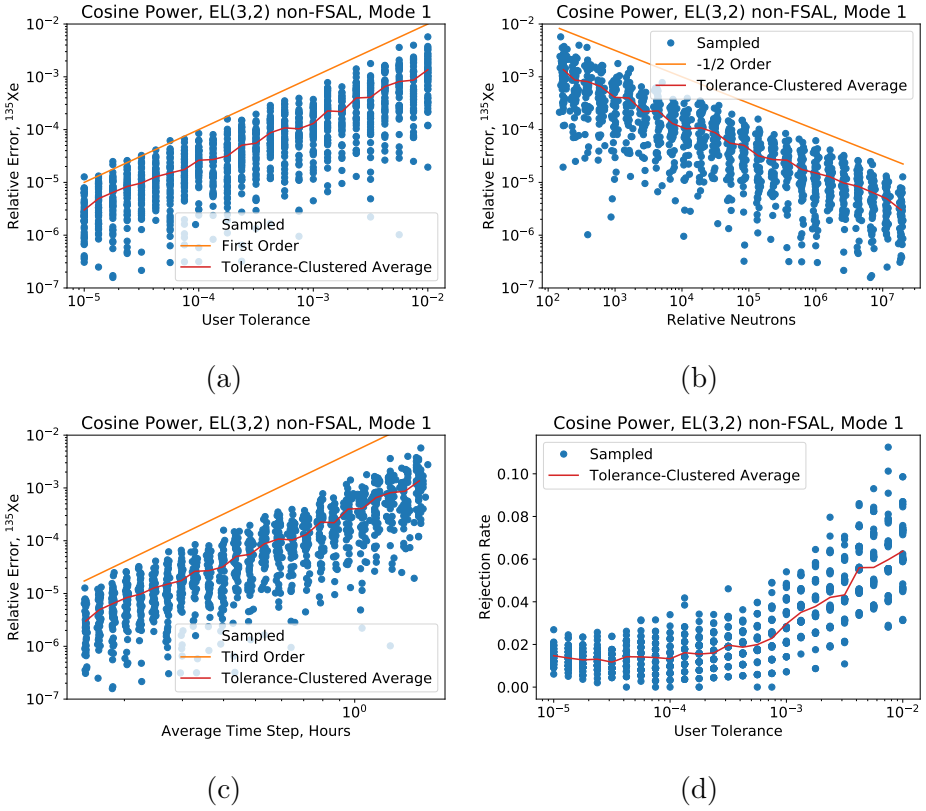


Figure 4-16: Mode 1 ATS scheme results

proportional to the square root of neutrons, and third order with average step size. The step rejection rate approaches 2% as the tolerance tightens. Using the same scheme from the hyperparameter optimization stage, the figure of merit for Mode 1 is 305.

For Mode 2, similar results are shown in Figure 4-17. The key difference is that only sampling at the beginning of time reduces the accuracy of the N/h value. This harms the figure of merit, which for this method is 127. In addition, the average time step variance is much higher than the other ATS schemes.

For Mode 3, a new N/h was computed every 10 times steps. For this particular algorithm, the figure of merit improved beyond even the repeated sampling to 451, perhaps owing to the reduced bias effects from the larger neutron per A evaluations.

4.4 Facsimile Pin Conclusions

The facsimile test problem allows for a very thorough analysis of numeric and stochastic performance under effectively ideal conditions. First, the problem

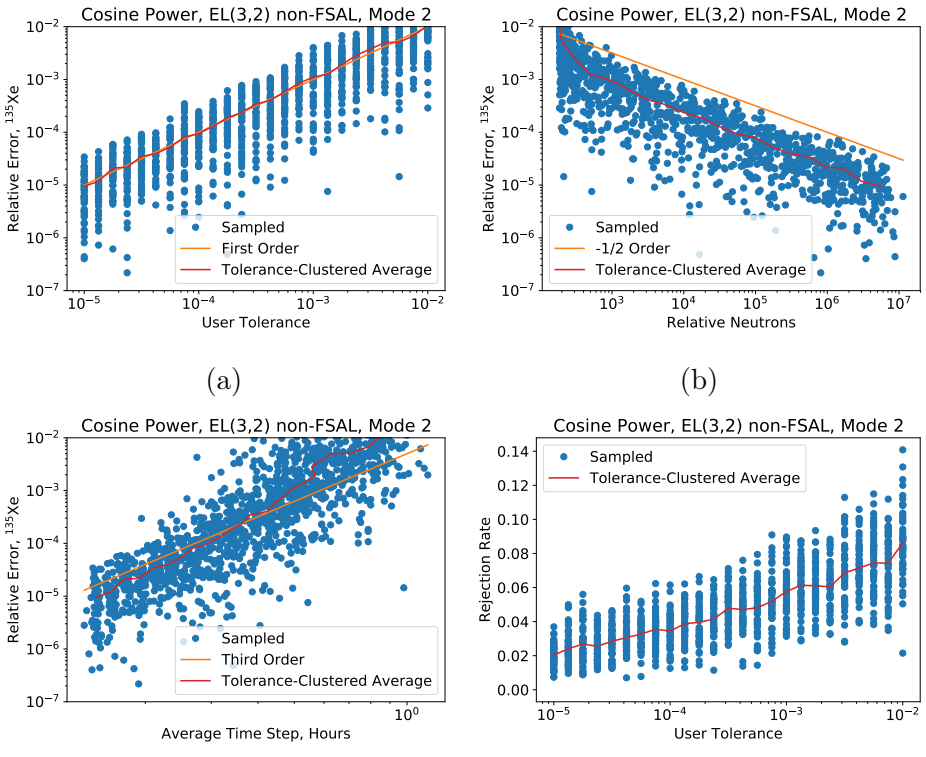


Figure 4-17: Mode 2 ATS scheme results

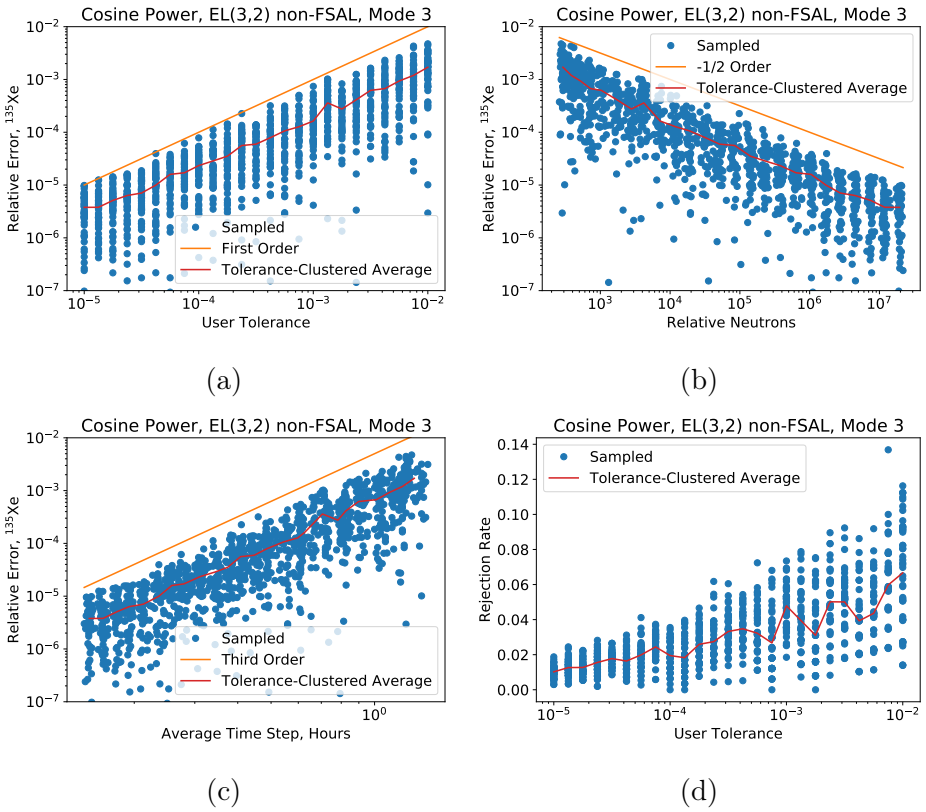


Figure 4-18: Mode 3 ATS scheme results

is simple enough that the scaling-and-squaring matrix exponential is valid and sufficiently accurate. Second, the statistics as a function of neutrons are precisely defined. Third, there is no spatial effects to allow for spatial oscillations which can plague certain simulations. While the facsimile test problem does not ideally emulate real reactor core depletion, its simplicity allows for rapid analysis and good precision. As a result, the conclusions that can be drawn are similarly incomplete. However, a few interesting points can be made already.

The primary question of this document is "what algorithm is best?" These results do not indicate a single perfectly optimal algorithm for either a deterministic or stochastic analysis. In the deterministic case, in general one will want to use a high order method for a high accuracy goal and a low order method for a low accuracy goal. The reason is displayed rather clearly in Figure 4-19. In this particular case, CE/LI would be a good choice for low accuracy and long time step purposes, while CF4 will run a factor of ten faster if an error of 10^{-8} was needed. In addition, high order methods have drawbacks, from loss of stability for long time steps to increased memory usage.

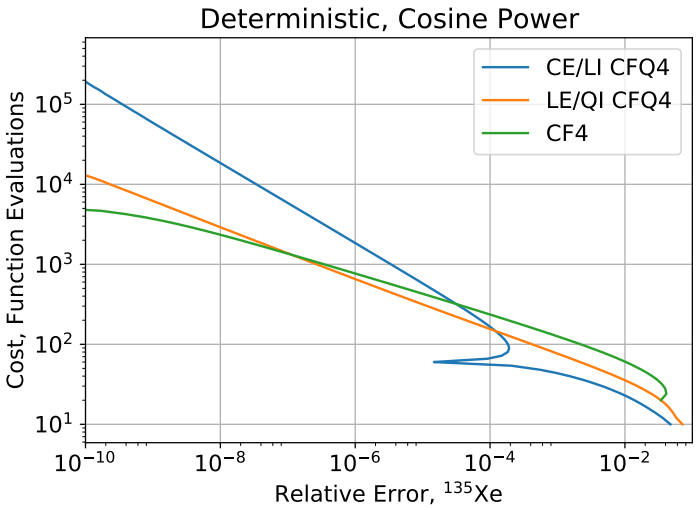


Figure 4-19: Cost as a function of accuracy, facsimile pin, cosine power

In the case of stochastic methods, any method in a stochastically limited regime will operate similarly to any other. High order methods converge to the stochastically limited regime faster. After that, there are slight variations in error propagation, with CE/CM performing worst with an approximately $\sqrt{2}$ increased mean error per neutron.

On the note of order, every single algorithm asymptotically matched the theoretical error. LE/QI, an up to third order method, had the most interesting results. The substepping / one term Magnus integrator (M1) was asymptotically

second order. However, as the number of steps increased, the domain in which the algorithm acted third order expanded. For a large number of substeps, the method would be indistinguishable from third order. However, integrating the nonautonomous ODE with a single step of the commutator-free fourth order integrator (CFQ4) transformed the method into a true third order one.

The adaptive time stepping methods form a promising capability. Deterministic adaptive time stepping methods can eliminate the need to know the time grid ahead of time. The stochastic adaptive time stepping can further eliminate the need to know how many neutrons are necessary as well. However, the sensitivity of the stochastic adaptive time stepping to hyperparameters might cause some problems to integrate slowly.

The next chapter will use some of the more promising algorithms on two full depletion problems in order to fill some gaps in the understanding of these algorithms. The first, a 2D model, is used to test the accuracy of the methods in a realistic environment. The second, a 3D fuel pin, is used to test how the algorithms interact with spatial oscillatory phenomenon.

Chapter 5

Full Depletion Tests

While the facsimile pin problem from Chapter 4 is a useful problem for rapid analysis of a numerical method, it does not have the level of complexity of a nuclear reactor core simulation. As such, two full depletion test problems were run. The first test problem, described in Section 5.1, is a 2D 3x3 fuel pin array with radially discretized fuel pins. The goal of this geometry is to maximize the numerical error in order to investigate the numerical capabilities of the algorithms. The second test problem, described in Section 5.2 and originally presented in [15] is a single 3D fuel pincell with axial discretization. In the first simulation, the geometry is reflected on the top, bottom, and all four sides of the pincell. As each region should be identical, it becomes easy to analyze stochastically induced stability issues. The second simulation replaced the top and bottom boundary conditions with vacuum.

These two different test problems will be integrated with many of the more promising algorithms tested in Chapter 4. This set of algorithms includes Predictor, SIE in reaction rate relaxation mode, CE/CM, CE/LI, LE/QI, EPC-RK4, EL3, and CF4. CE/LI was integrated in two different modes: substepping with $m = 5$ and a single step commutator-free integrator step. Similarly, LE/QI was integrated in three ways: substepping with $m = 5$, substepping with $m = 5$ with integral power normalization, and a single step commutator-free integrator step. In addition, the stochastic implicit CE/LI and LE/QI were tested.

5.1 2D Depletion Test

The goal for this first test is to create a realistic problem that is as challenging as possible. One of the hardest problems in depletion is that of the evolution of a burnable absorber under irradiation. The burnable absorber both spectrally and spatially shields itself and other nuclides. As such, as it burns up, the reaction rate will change very rapidly. A good burnable absorber to maximize this impact would be gadolinium due to its very high thermal cross section

and due to it being homogeneously mixed in the fuel.

The model used is a 2D 3x3 fuel pin array with 4.5% enriched UO_2 fuel. The center fuel pin has 2% by weight natural gadolinium added. All fuel pins are then discretized into 10 rings for depletion. The precise dimensions and initial compositions are listed in Appendix C. All boundaries are reflective. This geometry is shown in Figure 5-1. This fuel pin array operates at a power of 2.337×10^{15} MeV / second-cm (approximately 37.44 kW/m).

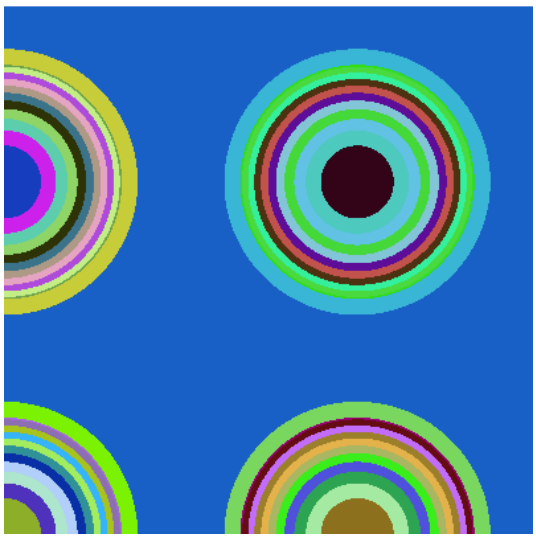


Figure 5-1: The 2D depletion geometry

For this first problem, a complete set of decay chains were used. This set contains 3821 nuclides from the ENDF-B/VII.1 decay library. For energy deposition, as mentioned in Section 1.2.1, all fissions are set to generate 200 MeV and no other energy deposition is tracked. For isomeric capture branching ratios, as mentioned in Section 1.2.2, the Serpent energy-independent ratios were used. To ensure the computation of all microscopic reaction rates at the beginning of time, an almost infinitely dilute quantity (10^8 at/cm^3) of all nuclides with cross sections in ENDF-B/VII.1 (423 nuclides) was added if the initial condition was zero.

In Section 5.1.1, a reference solution is computed and analyzed to determine when numerical error is at its maximum. This point in time is then used as the end point for the non-reference simulations. In Section 5.1.2, fixed time step methods will be analyzed. Finally, in Section 5.1.3, adaptive time step methods will be tested.

5.1.1 2D Model Reference Solution

The development of a reference solution for depletion in which transport is computed with Monte Carlo is a non-trivial one. As demonstrated in the prior chapter, there will be errors from both numerical and statistical effects. These will need to be quantified in order to determine if observed errors are actually real.

Statistical error is for the most part easy to quantify. One just needs to run the simulation several times and compute a distribution. With that in mind, each reference solution was run 25 times. The mean was taken as the reference solution, and the standard deviation used to estimate uncertainty bounds.

Numerical error is particularly challenging to estimate. One possibility is to perform the same integration twice with the time step halved. If the error between the high accuracy reference and low accuracy reference is much smaller than the error between the high accuracy reference and the integration under test, then it is expected that the errors are real. However, as enumerated in [44], there are many possible scenarios in which this tactic will fail.

For this particular problem, two solutions were computed. The high accuracy reference has 6 hour time steps until day 6 followed by 1 day time steps to day 200. The low accuracy reference has 12 hour time steps until day 6 followed by 2 day time steps to day 200. This grid allows for direct comparisons at every time for the low accuracy grid. Both solutions were computed with the CE/LI method with 5 substeps. The neutronics solver was run with the parameters listed in Table 5.1.

	High Accuracy	Low Accuracy
Batches Total	40	40
Batches Active	20	20
Neutrons / Batch	30000	30000
Active Neutrons / Entire Depletion	261.6 million	130.1 million
Active Neutrons Total	6.54 billion	3.27 billion

Table 5.1: 2D fuel pin array reference solution simulation parameters

In Figure 5-2, several global error measurements as a function of time are plotted. In order to compute standard deviations, each high accuracy answer was paired off to one low accuracy answer. Then, the error was computed for each of the 25 pairs. The standard deviation is then calculated from this set of 25 errors. For both relative error measurements, nuclides with quantites below 10^{10} were ignored. In this particular case, the 2-norm measurement slowly grows until around day 100 and then levels off. Both the relative error measurements are nearly constant over the entire time interval.

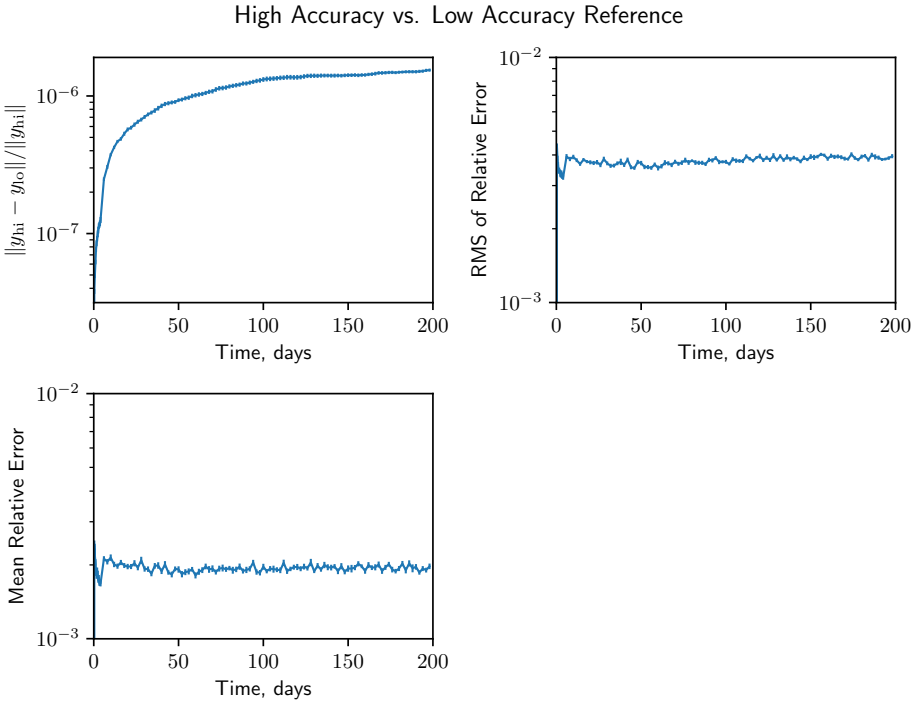


Figure 5-2: Global error comparison with time, reference comparison

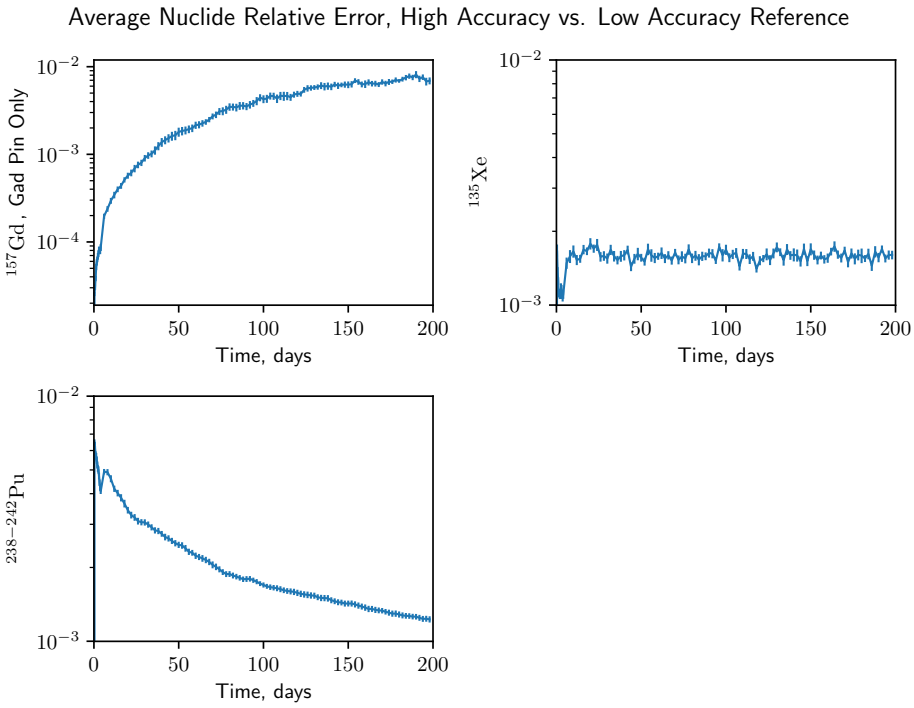


Figure 5-3: Nuclide error comparison with time, reference comparison

In Figure 5-3, a few groups of nuclides are singled out. The first one is ^{157}Gd, in which the relative error steadily grew with time up until around 150 days. This was when the outer layers of fuel were depleted of ^{157}Gd and the inner layers were just beginning to be depleted. The error in ^{135}Xe is fairly constant, but it also exhibits significantly more noise than the other components. This will be further investigated in Section 5.2. Finally, the average relative error for five plutonium nuclides from ^{238}Pu to ^{242}Pu is plotted. In this case, the errors are mostly in the first few time steps. As the population grows and stabilizes, the errors diminish.

As the goal was to maximize the gadolinium error, an end time of 150 days was chosen. The estimated errors of the low accuracy solution at a time of 150 days for each quantity are tabulated in Table 5.2. These errors are for the most part numerically dominated. As such, later comparisons will only use the mean of the high accuracy reference solution for comparison purposes.

Test	μ	σ	$\mu + 2\sigma$
2-norm	1.42×10^{-6}	4.36×10^{-8}	1.50×10^{-6}
RMS Relative Error	3.94×10^{-3}	8.94×10^{-5}	4.12×10^{-3}
Mean Relative Error	1.93×10^{-3}	5.75×10^{-5}	2.05×10^{-3}
^{157}Gd	6.24×10^{-3}	5.49×10^{-4}	7.33×10^{-3}
^{135}Xe	1.69×10^{-3}	9.51×10^{-5}	1.88×10^{-3}
$^{238\text{-}242}$Pu Average	1.43×10^{-3}	3.94×10^{-5}	1.50×10^{-3}

Table 5.2: Estimated errors at a time of 150 days for the reference depletion problem

5.1.2 2D Model Constant Time Step

Now that a high quality reference solution has been computed and the errors estimated, a rigorous analysis of performance can be performed. The goal was to see how each algorithm fared numerically and stochastically for different time steps and neutron totals.

With this in mind, each algorithm was run with time steps of 6, 10, 15, 25, 30, 50, 75, and 150 days. These were chosen due to being factors of 150. The exception to this are the three stochastic implicit algorithms. These were not run with a time step of 6 days, as the overhead of restarting the Monte Carlo simulation (due to loading of cross sections and geometry) has not yet been eliminated. Since these algorithms need to be restarted many times, this overhead often exceeded simulation time by a large margin.

Similarly, each simulation was run with a total active neutron quantity of 1, 2, or 4 million to allow analysis of stochastic convergence. Batches were kept

constant at 40 batches with 20 active. This was done to ensure that effects such as autocorrelation [24] had a constant impact. As a result, the neutrons per batch was the only value adjusted. Neutrons were evenly divided for each transport calculation and rounded to the nearest integer, with the exception of SI-CE/LI and SI-LE/QI. For these two algorithms, the first simulation was run with 10 times as many neutrons per batch as the rest, as discussed in Section 3.2. To compute statistics, each simulation was run 25 times with different random sequences.

In order to keep things manageable, the results are broken up into three chunks. The first chunk contains all explicit methods except for the CFQ4 or integral power normalized polynomial predictor-corrector results. The second batch contains a comparison of subintegrators for CE/LI and LE/QI. Finally, the implicit methods are compared.

5.1.2.1 General Comparison of Explicit Algorithms

Two views of the global errors (2-norm, RMS of relative error, mean of relative error) are plotted in Figure 5-4 and Figure 5-5 for the explicit algorithms. The nuclide errors are plotted in Figure 5-6 and Figure 5-7. The first figure of each set shows the convergence with time step for a fixed 4 million neutrons per depletion. The second figure shows the convergence with a fixed time step of 6 days for the algorithms that were stochastically bounded (as in, it did not appear that numerical convergence was still occuring).

The first order predictor method performs rather poorly. Over the entire domain, it is outperformed by all other algorithms, often by at least a factor of three. Even with a 6 day time step, the average ^{157}Gd error is approximately 43%. For other nuclides, errors approach 1% on average, indicating that the predictor method can be used with short time steps on easy problems.

The second order family (CE/CM, CE/LI, EPC-RK4) fare far better. Both CE/CM and CE/LI perform similarly to one another numerically, with the lines overlapping for large time steps. Both methods are stochastically bounded by 6 days. The primary difference between the two methods is the error in the statistical region. As predicted in Chapter 4, CE/CM has a higher error. In the 2-norm test, CE/CM has errors that are a factor of 1.36 higher than CE/LI on average. This is close to the predicted $\sqrt{2}$.

EPC-RK4 acts quite a bit different. In the ^{157}Gd test, the error is approximately an order of magnitude better than CE/CM or CE/LI. Similar gains appear in many of the other tests. Stochastically, EPC-RK4 consistently performed either best or nearly best of all explicit algorithms tested.

Moving on to the third order methods, LE/QI has a number of interesting results. First, it must be noted that LE/QI requires a different algorithm to start the integration. In this case LE/QI M1 used CE/LI M1. As such, the

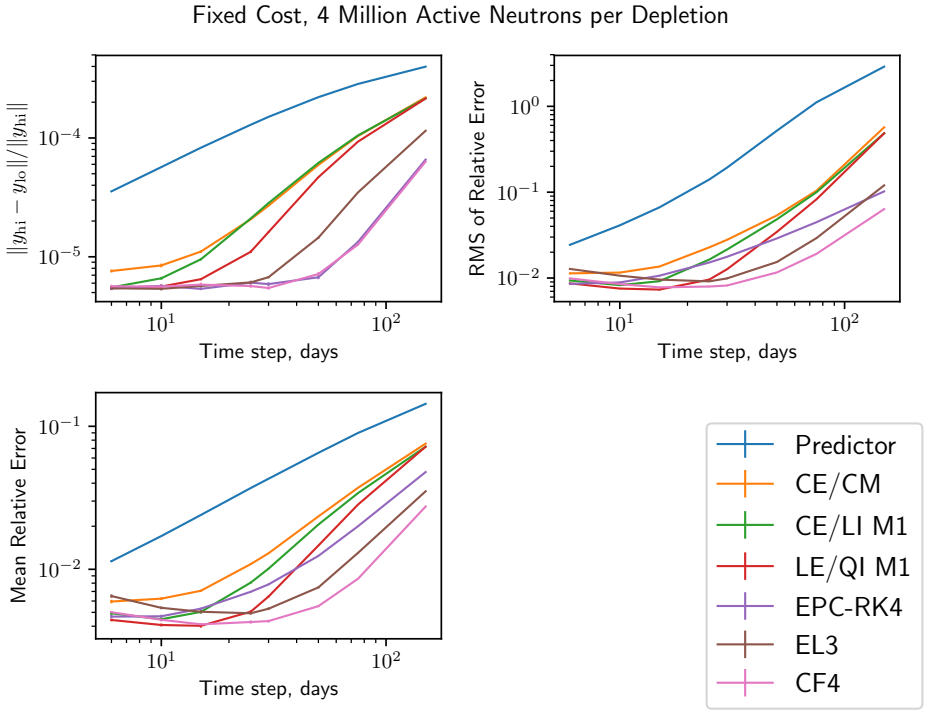


Figure 5-4: Global error comparison at $t = 150$ days, 4 million neutrons

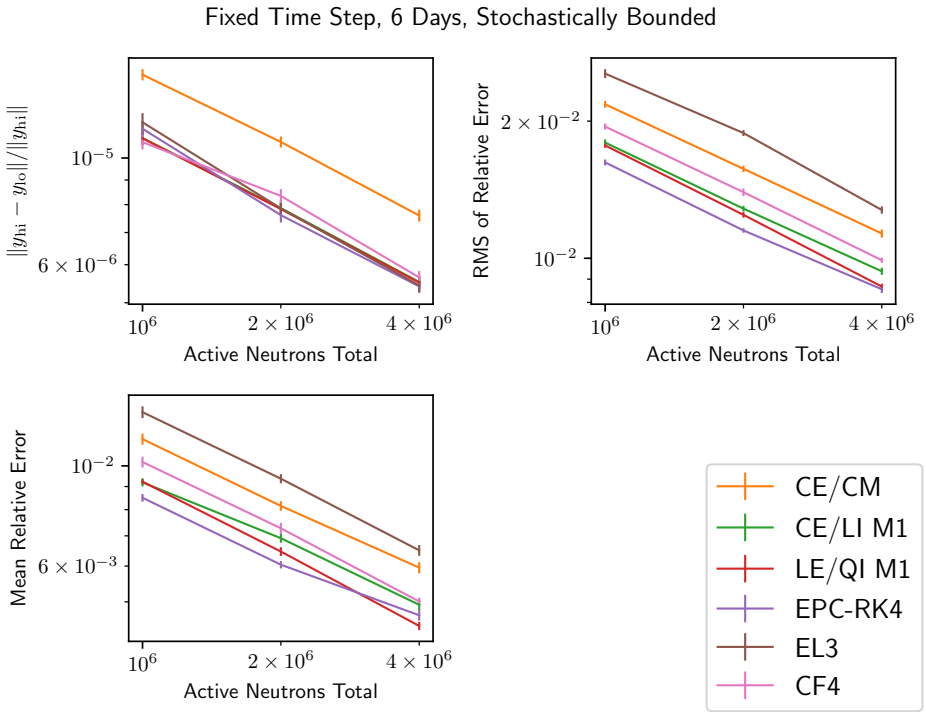


Figure 5-5: Global error comparison at $t = 150$ days, 6-day time step

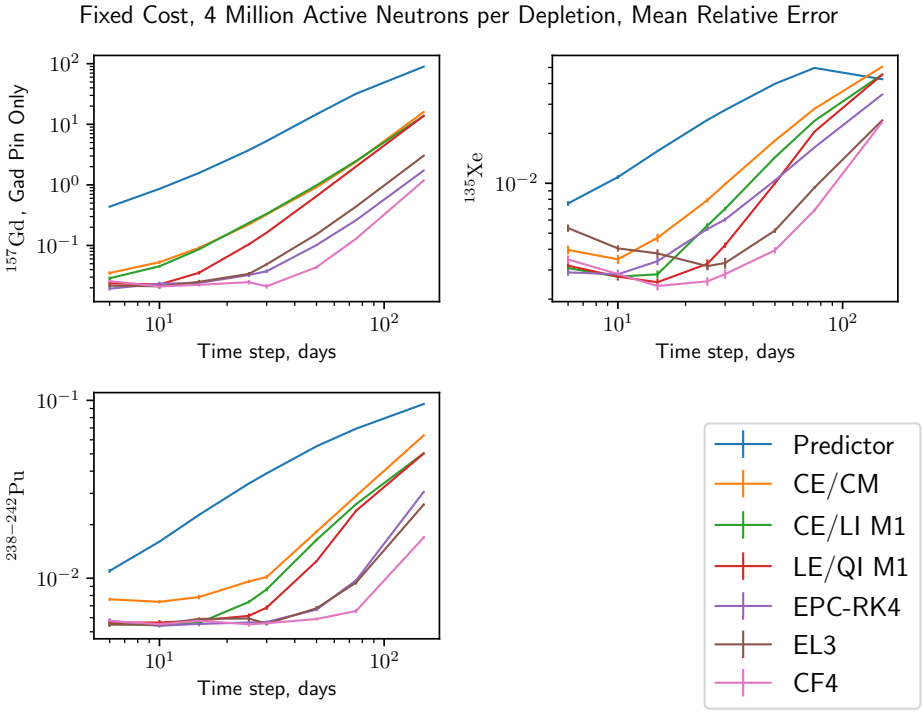


Figure 5-6: Nuclide error comparison at $t = 150$ days, 4 million neutrons

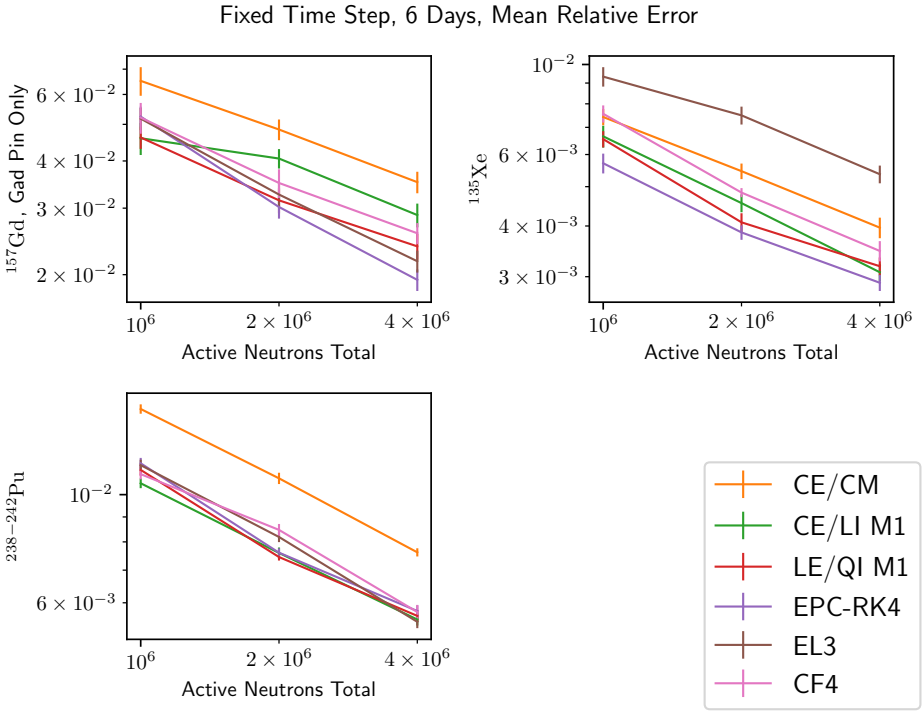


Figure 5-7: Nuclide error comparison at $t = 150$ days, 6-day time step

	^{157}Gd			^{135}Xe			$^{238\text{-}242}$Pu		
	10%	1%	0.1%	10%	1%	0.1%	10%	1%	0.1%
Predictor	1.37	0.137	0.0137	88.7	8.87	0.887	54.5	5.45	0.545
SIE, RR	2.79	0.279	0.0279	144.0	14.4	1.44	50.4	5.04	0.504
CE/CM	15.8	5.0	1.58	95.8	30.3	9.58	90.2	28.5	9.02
CE/LI M1	16.1	5.11	1.61	122.0	38.6	12.2	106.0	33.7	10.6
LE/QI M1	24.5	7.74	2.45	158.0	50.0	15.8	131.0	41.4	13.1
LE/QI CFQ4	24.9	11.5	5.36	102.0	47.4	22.0	87.4	40.6	18.8
EPC-RK4	49.6	15.7	4.96	152.0	48.2	15.2	241.0	76.2	24.1
EL3	41.3	19.2	8.9	169.0	78.4	36.4	168.0	77.9	36.2
CF4	68.1	38.3	21.5	165.0	92.7	52.1	182.0	102.0	57.4

Table 5.3: Estimated maximum time step (days) to achieve accuracy

errors correspond when there is only one time step, as they are effectively the same algorithm. Additionally, while LE/QI M1 is not truly third order, the substepping algorithm is sufficiently accurate to improve convergence as compared to CE/LI. Statistically, LE/QI was roughly on par with CE/LI.

The accuracy of EL3 is in general superior to LE/QI in the numerically bound region. In most tests, EL3 had errors approximately a factor of two lower than LE/QI. The gadolinium results were even better at a factor of 4.5. However, in the stocastically bounded region, there are issues. In ^{135}Xe and both relative error tests, the statistical error is the largest of all tested algorithms. This is likely due to instabilities, which will be examined more thoroughly in Section 5.2.

CF4, the sole fourth order algorithm, has even more impressive results. For all tests, it had the lowest error of the explicit algorithms. It was converged to the stochastically limited region by a time step of 30 days in all tests. The results at 6 days put the stochastic performance roughly in the middle of the pack.

It is possible to make some estimates on the requirements to achieve a given error using the data in Figure 5-6 and Figure 5-7. To estimate the necessary time step, a point was chosen for each algorithm from the time step convergence results and extrapolated using the theoretical order of the method (as in, LE/QI M1 is second order, LE/QI CFQ4 is third). For ^{157}Gd, the point was where the error was 10%. If this value did not exist, the most accurate value was used instead. Similarly, 1% was used for ^{135}Xe and $^{238\text{-}242}$Pu. The results are shown in Table 5.3.

In this particular case, ^{157}Gd forms the greatest challenge numerically. To achieve a 1% error, the first order methods need time steps on the scale of hours. The second order methods need time steps on the scale of a week. EL3

	^{157}Gd	^{135}Xe	$^{238\text{-}242}$Pu
CE/CM	4.16×10^8	5.32×10^6	2.07×10^7
CE/LI M1	2.59×10^8	3.70×10^6	1.05×10^7
CE/LI CFQ4	2.39×10^8	3.37×10^6	1.01×10^7
LE/QI M1	1.91×10^8	3.49×10^6	1.09×10^7
LE/QI CFQ4	2.24×10^8	3.24×10^6	1.11×10^7
EPC-RK4	1.79×10^8	2.88×10^6	1.15×10^7
EL3	2.00×10^8	9.40×10^6	1.16×10^7
CF4	2.34×10^8	4.55×10^6	1.19×10^7

Table 5.4: Estimated minimum total active neutrons to achieve 1% accuracy

approaches a month, and CF4 exceeds it. Converging even further to 0.1% yields absurd time steps for the first order methods. Predictor needs a time step that is less than an hour.

The other value extracted from the figures was the number of neutrons to achieve an error. For this problem, the average of $\epsilon\sqrt{N}$ at a time step of 6 days was taken and then Equation (5.1) was computed. This forms an approximation to the number of neutrons required to achieve a given error. This yields Table 5.4. To compute 10% and 0.1%, multiply the value by 10^{-2} and 10^2 respectively.

$$N_i = \frac{\left(\overline{\epsilon\sqrt{N}}\right)^2}{\epsilon_i^2} \tag{5.1}$$

Once again, ^{157}Gd is far more strict than the other nuclides. The only algorithm that performs significantly different is CE/CM, which requires roughly twice as many neutrons as the other methods. This echos the conclusions from Chapter 4. The exact same result appears in the plutonium comparison. However, in the ^{135}Xe case, EL3 performs abnormally poorly. It would require approximately three times as many neutrons as other methods. It is worth noting though that this is of little impact, as an accurate answer would need even more neutrons to converge the plutonium values.

5.1.2.2 CE/LI and LE/QI Subintegrator Comparison

With the overall comparison completed, it is worthwhile to see if the subintegrator in CE/LI and LE/QI has any impact on the solution whatsoever. The global results are plotted in Figure 5-8 and Figure 5-9. The specific nuclide results are plotted in Figure 5-10 and Figure 5-11.

For all practical purposes, all subintegrators performed identically to one another. However, this problem may simply mask the advantages of some

integrators. For example, LE/QI M1 should eventually converge second order, perhaps at some point with far tighter accuracy than measured here. The integral power normalization might benefit a problem in which full transient energy deposition is performed. This is not the case here, as fission heat is the only heat source and is treated promptly. Under these circumstances, since CFQ4 only requires 2 matrix exponentials in comparison to the 5 used in this particular substep, it is recommended.

5.1.2.3 Stochastic Implicit Methods

Finally, the stochastic implicit (SI) family of algorithms were tested. This includes SIE, SI-CE/LI, and SI-LE/QI. Due to the fact that these algorithms require many restarts of the transport operator, 10 per time step for SIE and 11 per time step for the other two, the 6-day time step simulations were skipped. The global comparison is plotted in Figure 5-12 and the nuclide comparison is plotted in Figure 5-13.

For large time steps, SIE is more accurate than predictor except on the plutonium error. As the number of time steps increases, the two algorithms begin to approach one another. Many of the conclusions made about predictor, that it would not be a good choice for ^{157}Gd and that it requires relatively short time steps, are the same for SIE.

The SI-CE/LI and SI-LE/QI methods also perform better initially than their non-implicit counterparts. Similar to SIE, the convergence initially is much lower than the asymptotic order. Due to this, extrapolation of required time step was not performed, as the extrapolation assumptions are invalid. However, it is expected that the required time step for both algorithms will be slightly better than their non-implicit forms.

The most interesting results come from the global relative error and ^{135}Xe values. Both of these algorithms perform better than even CF4 for these tests, owing to stabilized equilibrium nuclides. Performance improvements are less pronounced on "difficult" nuclides such as ^{157}Gd. For these nuclides, CF4 still outperforms.

5.1.3 2D Model Adaptive Time Step

For the adaptive time stepping, the non-first same as last EL(3,2) algorithm was used with the adaptive time stepping algorithm described in Section 3.6.2.3. N/h was updated every 10 time steps. The exact parameters are listed in Table 5.5. The minimum neutrons/batch was chosen to ensure that bias effects did not become too significant.

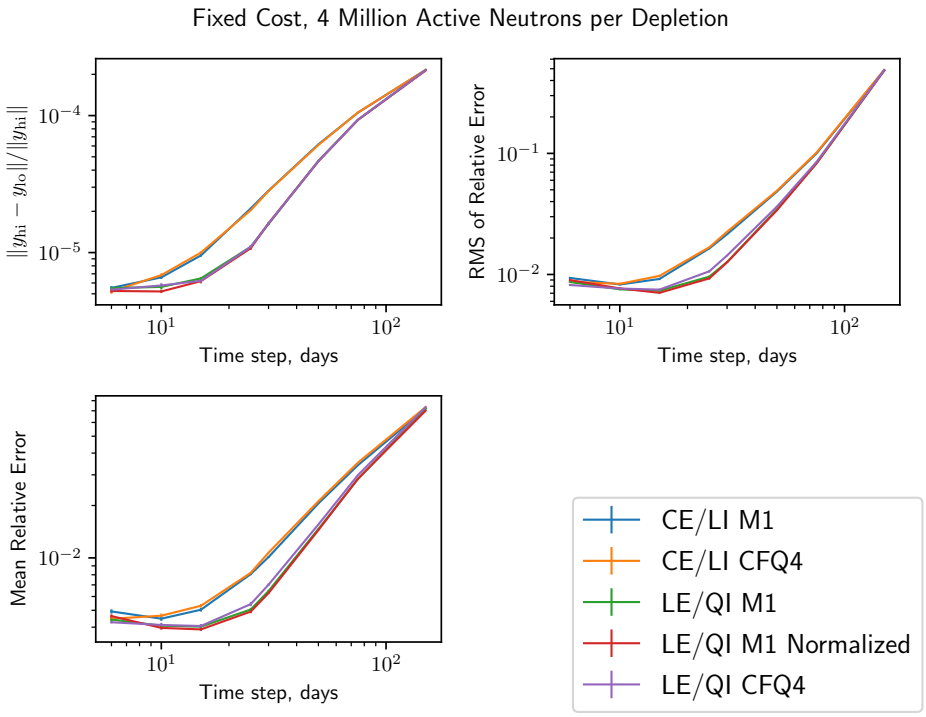


Figure 5-8: Global error comparison at $t = 150$ days, 4 million neutrons, subintegrators

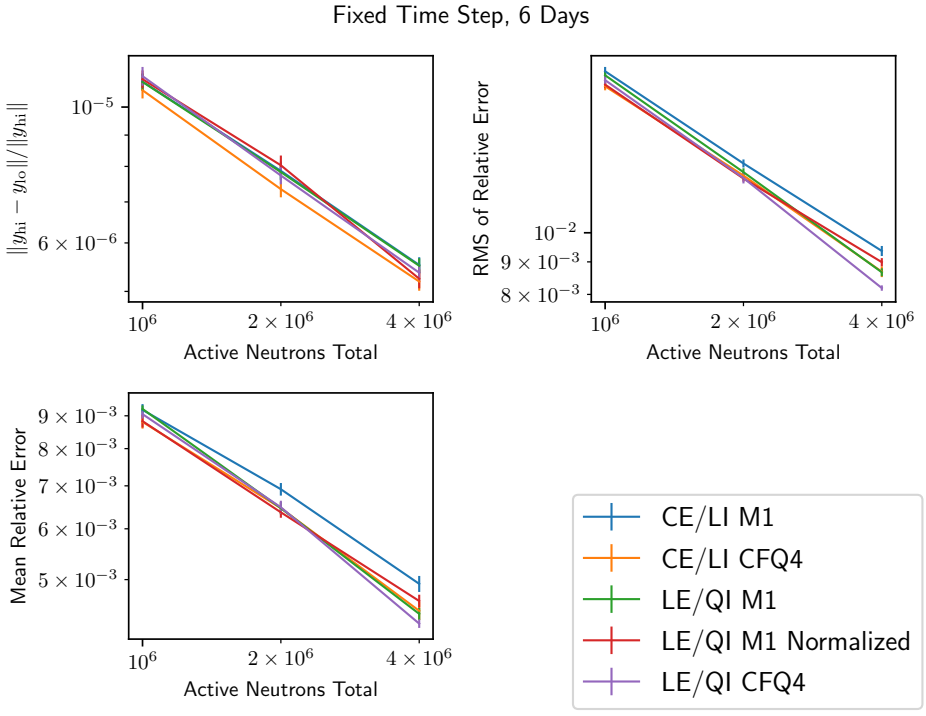


Figure 5-9: Global error comparison at $t = 150$ days, 6-day time step, subintegrators

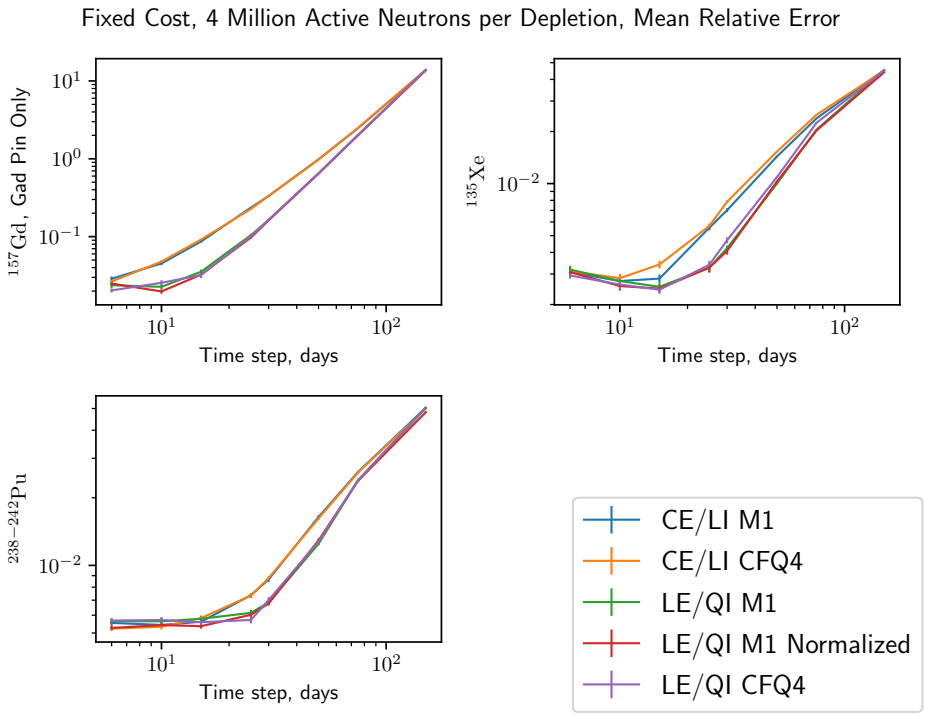


Figure 5-10: Nuclide error comparison at $t = 150$ days, 4 million neutrons, subintegrators

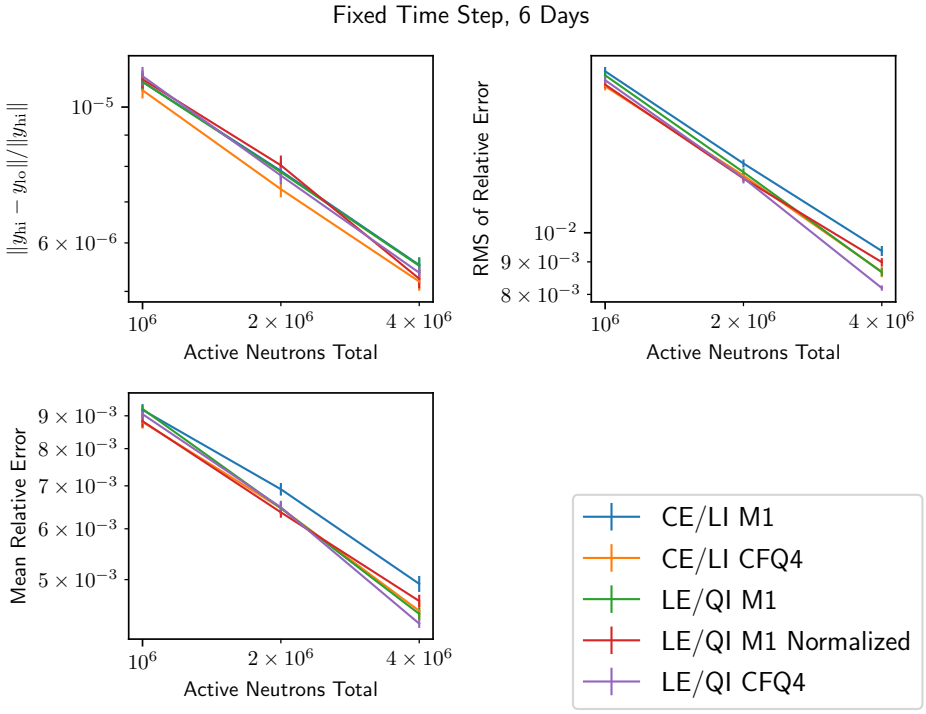


Figure 5-11: Nuclide error comparison at $t = 150$ days, 6-day time step, subintegrators

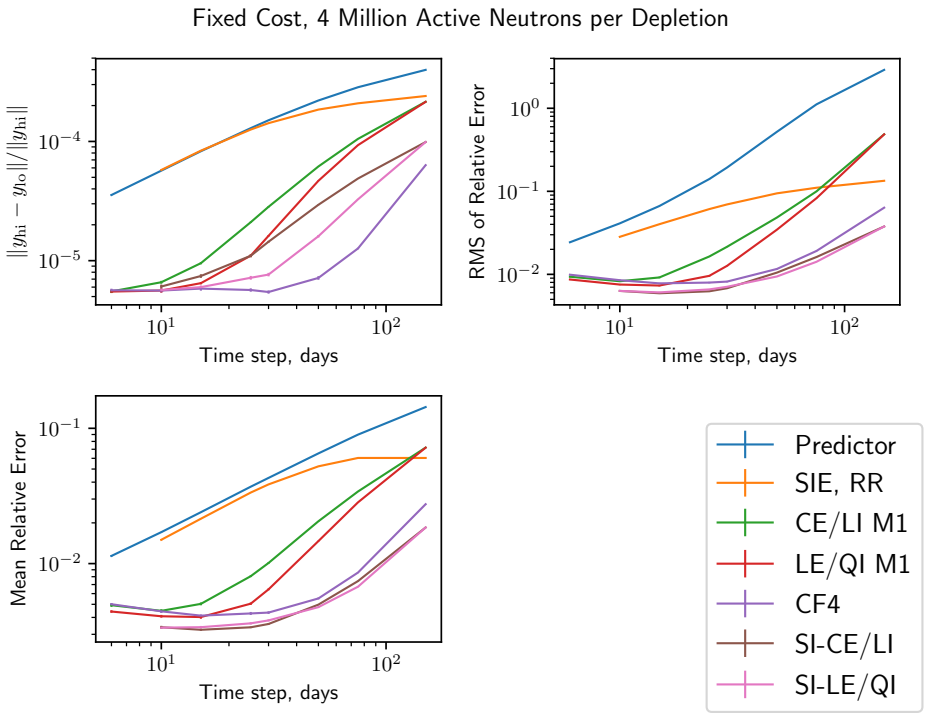


Figure 5-12: Global error comparison at $t = 150$ days, 4 million neutrons, stochastic implicit

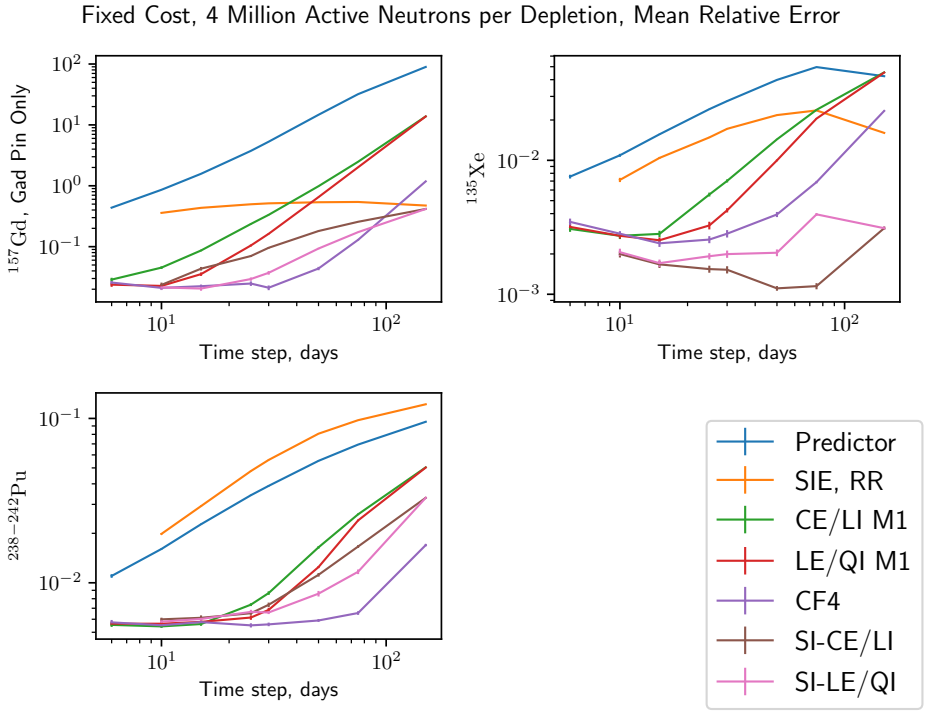


Figure 5-13: Nuclide error comparison at $t = 150$ days, 4 million neutrons, stochastic implicit

Parameter	Value
Initial Time Step	10 hours
Initial Neutrons/Batch	1000
Minimum Neutrons/Batch	1000
Batches	40
Active Batches	20

Table 5.5: Parameters for the adaptive time stepping simulation

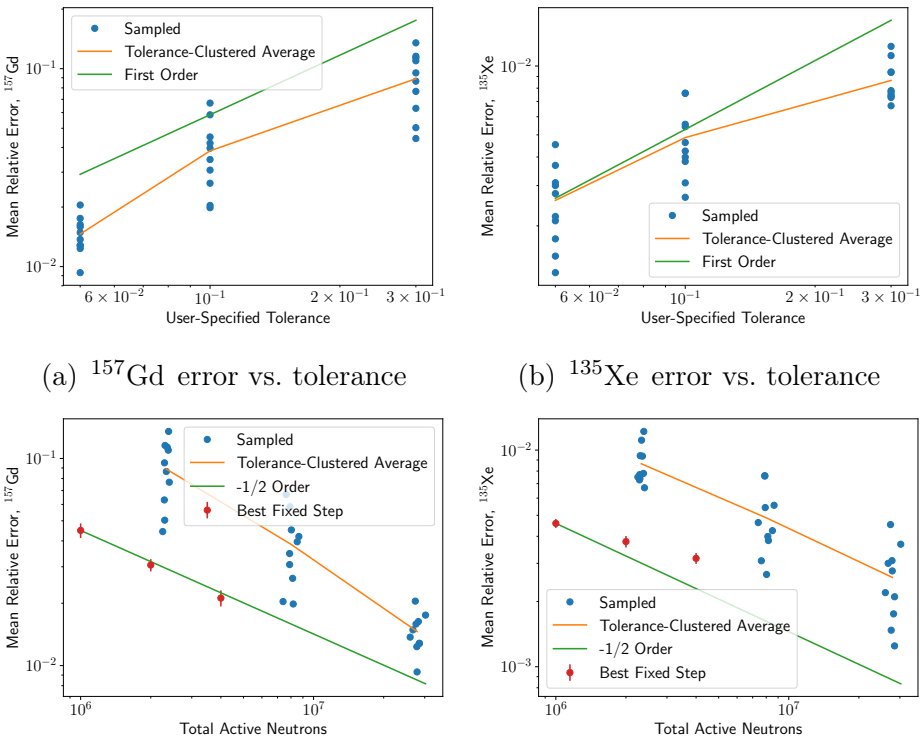
For each of the two error estimates described in Section 3.6.1, several simulations were run. For the nuclide based error estimate, integration was performed 10 times each for a tolerance of 0.05, 0.1, and 0.3. For the rate-based error estimate, integration was performed 10 times each for a tolerance of 0.001, 0.003, 0.01, and 0.03.

The results for the nuclide based error estimate are shown in Figure 5-14. The top row shows convergence of ^{157}Gd and ^{135}Xe as a function of user tolerance. For both nuclides, convergence is approximately first order, which is ideal. The second row shows how these nuclides converge with total active neutrons. In the ^{157}Gd case, convergence is slightly better than the anticipated $1/\sqrt{N}$, indicating at least some of the convergence is due to time step. ^{135}Xe is converging $1/\sqrt{N}$. However, for both results, errors are higher than the best fixed step results.

The reaction rate error estimate results are shown in Figure 5-15. The error of ^{157}Gd converges approximately linearly with tolerance, aside from an issue at very loose tolerances. ^{135}Xe is not converging linearly. Since the magnitude of error for ^{157}Gd is much higher, this could simply be due to the fact that the error estimate is largely composed of ^{157}Gd. The results for this error estimate are much better as a function of neutrons as compared to the nuclide based error. In addition, both nuclide errors are approaching the best fixed step results. ^{135}Xe is not converging $\mathcal{O}(1/\sqrt{N})$ here.

To further investigate the convergence with neutrons, the ^{135}Xe results were plotted against the average neutron per time step in Figure 5-16. In this case, the ^{135}Xe results are much closer to the -1/2 order. This indicates that nuclides in equilibrium can be more sensitive to neutrons per time step than total neutrons.

Overall, these results indicate that the adaptive time stepping scheme works in essence, but performance is very sensitive to the choice of error estimate. While the reaction rate error estimate works the best of the two, it still depends on what exactly the user cares about.



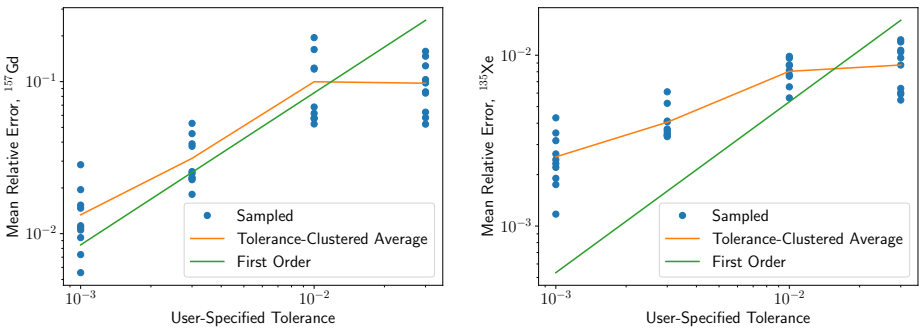
(a) ^{157}Gd error vs. tolerance

(b) ^{135}Xe error vs. tolerance

(c) ^{157}Gd error vs. neutrons

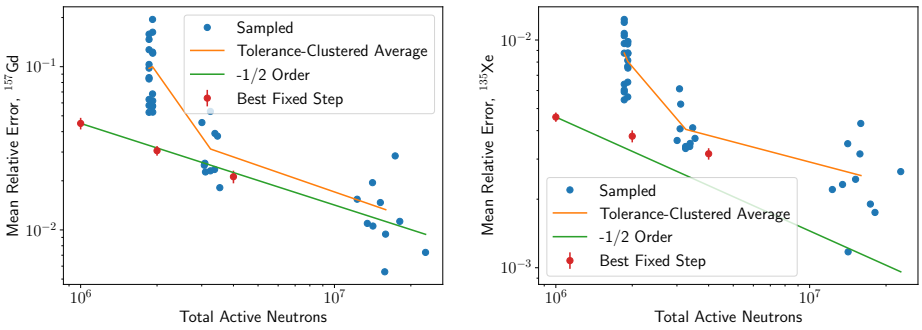
(d) ^{135}Xe error vs. neutrons

Figure 5-14: Adaptive time stepping results for the nuclide based error estimate



(a) ^{157}Gd error vs. tolerance

(b) ^{135}Xe error vs. tolerance



(c) ^{157}Gd error vs. neutrons

(d) ^{135}Xe error vs. neutrons

Figure 5-15: Adaptive time stepping results for the reaction rate based error estimate

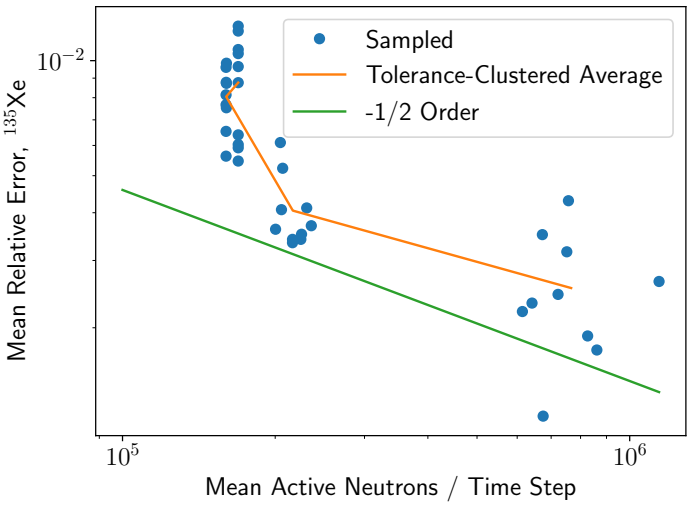


Figure 5-16: ^{135}Xe mean relative error vs. the mean active neutrons per time step for the reaction rate based error

5.2 3D Depletion Test

As mentioned in Section 1.1.3, another major issue is xenon instabilities. Given an extremely loosely coupled model with no feedback, oscillations can initiate and grow rapidly. With short time steps, these oscillations mimic what would physically happen with no feedback. With long time steps, this tendency to oscillate can prevent gathering useful information.

In order to test how some algorithms interact with oscillations, a 3D single fuel pin was discretized axially. The exact model comes from [15]. The fuel is 10.0 g/cm^3 UO_2 enriched 3.1% with a radius of 0.41 cm. The cladding is zirconium with an outer radius of 0.475 cm. As the density is not listed, a density of 5.68 g/cm^3 of natural zirconium was used. The coolant is 0.7 g/cm^3 light water with no boron. The pitch is 1.26 cm, and the height is 400 cm. The fuel is axially discretized into 8 zones.

In order to observe oscillations, a very large number of batches are required. The reason for this is that the geometry has a dominance ratio of nearly unity. If the ^{135}Xe in a region is slightly higher, it takes a very large number of iterations for the neutron population to leave that region. With that in mind, 5000 batches were used, with 2500 active. Each simulation was then run with several numbers of total neutrons, as listed in Table 5.6. Each of these was then run 10 times to allow for statistical analysis. It is worth mentioning that even with this number of neutrons, the neutrons per batch is quite small. For example, with predictor, the largest simulation has 32000 neutrons per batch. This is at least an order of magnitude too few to avoid clustering effects [17]. As

such, clustering will create oscillations. ^{135}Xe is sensitive to these oscillations and will allow them to magnify.

Time Step	7 days
End of Simulation	70 days
Batches Total Per Sim	5000
Batches Active Per Sim	2500
Active Neutrons Total	100, 200, 400, 800 million

Table 5.6: Oscillation simulation parameters

Even then, this very large quantity of neutrons poses some computational issues. The decay chain and reaction rate set used in the previous section performs quite poorly in OpenMC. To improve performance, the decay chain used in the facsimile pin in Chapter 4 was used instead with a single modification. In this case, the ^{135}I (n, γ) reaction was added back in, with nuclides undergoing this reaction being removed from the model. Since this model is only concerned with ^{135}Xe, this was considered a suitable compromise.

Finally, two different boundary conditions were considered. The first, a fully reflective boundary condition on all six sides, closely follows the original model. The second is modified such that the top and the bottom of the fuel pins have vacuum boundary conditions.

5.2.1 Reflective Boundary Condition

The original model that this is based on assumes a fully reflected boundary condition. This yields a rather interesting result. Since all regions in the fuel become neutronically identical, all regions should deplete to the same value. Stochastic oscillations will break this symmetry. Due to the loose coupling between regions, there is very little to dampen the oscillations. However, this also makes a very unrealistic geometry.

For each time step and each simulation, the relative difference between each cell ^{135}Xe concentration and the average was computed. In an ideal simulation, this value would be zero. Then, the average over each cell and each time step was computed for all 10 simulations. Using this value, the mean and standard deviation was computed. The results are then plotted in Figure 5-17 through Figure 5-19.

Starting with Figure 5-17, the results are very mixed. The majority of algorithms are loosely clustered together. EL3 performs quite poorly, with errors exceeding unity on average. Conversely, EPC-RK4 has errors over an order of magnitude below any other algorithm. In addition, it is the only algorithm that is converging with increasing number of neutrons. However,

combined with later results, it is expected that all algorithms will eventually converge as the number of neutrons is increased.

Moving on to Figure 5-18, while switching subintegrators does not improve results too much, the CFQ4 subintegrator is typically better than a $m = 5$ substepping integrator. In addition, CE/LI slightly outperforms LE/QI.

Finally, Figure 5-19 shows the three stochastic implicit methods. All three algorithms outperform any other explicit algorithm, with EPC-RK4 yielding results roughly a factor of 2.3 higher than the implicit methods. In addition, all three algorithms are also converging with neutrons, roughly $1/\sqrt{N}$.

5.2.2 Vacuum Top/Bottom Boundary Conditions

With the addition of vacuum boundary conditions on the top and bottom, the geometry becomes much more realistic. However, all cells are no longer identical. There is still some symmetry, so instead of comparing the ^{135}Xe values to the mean across the geometry, the same value is computed for each pair of symmetric cells. This does not detect some oscillatory modes, but it should detect most.

The results are plotted in Figure 5-20 through Figure 5-22. For this particular problem, the results are much the same as the reflected geometry. Excluding EL3 and EPC-RK4, the explicit algorithms are still bunched up together with relatively little variation. EL3 yields errors over an order of magnitude larger than this group. EPC-RK4 yields errors a factor of 3.5 lower. There are two differences. First, EPC-RK4 performs almost identically to the stochastic implicit methods. Second, with the exception of EL3, all methods are converging with number of neutrons. Since more algorithms are converging on this test problem, it indicates that there may be a minimum problem and algorithm dependent neutron threshold above which all methods will converge and below which the oscillations reach a fixed magnitude.

5.3 Summary of Full Depletion Tests

The two different depletion tests present a more complete result than the earlier facsimile pin, at the cost of detail. The 2D fuel pin array test demonstrates how each algorithm would interact with a realistic problem in the worst-case scenario. In addition, the 3D fuel pin allows for the analysis of spatial stability, which was not possible with the facsimile pin.

From these results, it is hard to recommend the predictor algorithm for any geometry with a strong burnable absorber. The accuracy of the gadolinium results are poor. To achieve 0.1% accuracy, it is estimated that a 20-minute time step is required. Ignoring gadolinium, a half day time step is sufficient to capture many plutonium nuclides to suitable accuracy. In the 3D pincell problem,

predictor performs similarly to the majority of the non-implicit algorithms. Stochastic Implicit Euler performed similarly in accuracy but had much smaller oscillations in the 3D pincell problem.

CE/CM and CE/LI both perform nearly identical to one another, with the exception that the error in the stochastically dominated region is around a factor of 1.4 larger than CE/LI. The primary advantage of CE/CM is that it requires slightly less memory than CE/LI. If that is not an issue, CE/LI should be used instead. For CE/LI, subintegrator did not matter in any tested simulation. For both methods, time steps on the order of approximately 1 day are sufficient to achieve 0.1% average error in ^{157}Gd. For plutonium, this increases to almost 10 days.

The LE/QI method has two major drawbacks: it must be started with another algorithm and it requires more memory to use than CE/LI. The convergence was effectively third order over the tested domain even with the second order substepping integrator. This in theory will not always be the case as the convergence is tightened. Assuming that LE/QI converges third order, a roughly 5-day time step is sufficient for 0.1% accurate ^{157}Gd.

The EPC-RK4 method is an interesting algorithm. Although it is mathematically second order, it often outperformed third order methods at the level of errors we are interested in. In addition, in the 3D pincell test, it outperformed all of the other explicit algorithms. Even converging at second order, EPC-RK4 does not need short time steps. A 5 day time step is sufficient for 0.1% accurate ^{157}Gd.

The EL3 algorithm was quite good in a stable problem such as the 2D fuel pin array. With adaptive time stepping, results approached that of optimal fixed time step solutions. On an unstable problem, the accuracy fell apart. The resulting oscillations were several orders of magnitude larger than many of the other algorithms. As such, it should not be used in such circumstances.

CF4 performed the best on the "difficult" nuclides such as ^{157}Gd and the five plutonium nuclides. Its fourth order performance also gave it the longest maximum time step required to achieve 0.1% average errors at roughly 20 days for ^{157}Gd and 50 days for plutonium. This accuracy makes it highly recommended for deterministic transport analyses.

Finally, SI-CE/LI and SI-LE/QI both performed very well. Accuracy was as good or better than their non-implicit counterparts for all tests and time steps. Both methods also were as stable as SIE, making all three good picks for highly oscillatory problems. When used with Monte Carlo, the stochastic implicit form is almost free as each of the m simulations can be run with $1/m$ particles. However, one needs to be careful not to make batch sizes too small. The added cost is increased memory which is used to store the average reaction rates, and the extra depletion operations necessary. Both algorithms are recommended.

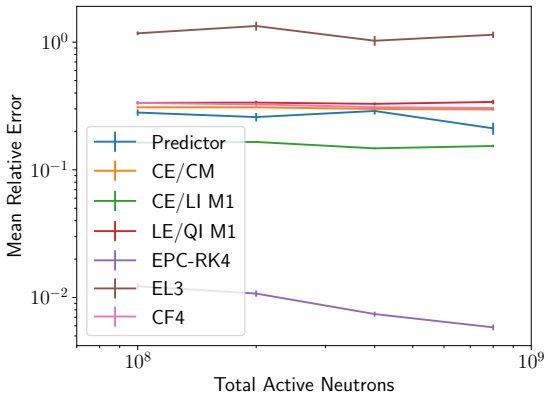


Figure 5-17: Mean coefficient of variation of ^{135}Xe for the fully reflected 3D pin problem, explicit comparison

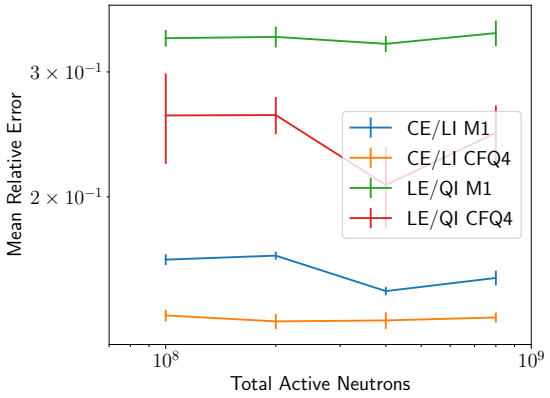


Figure 5-18: Mean coefficient of variation of ^{135}Xe for the fully reflected 3D pin problem, CE/LI and LE/QI subintegrator comparison

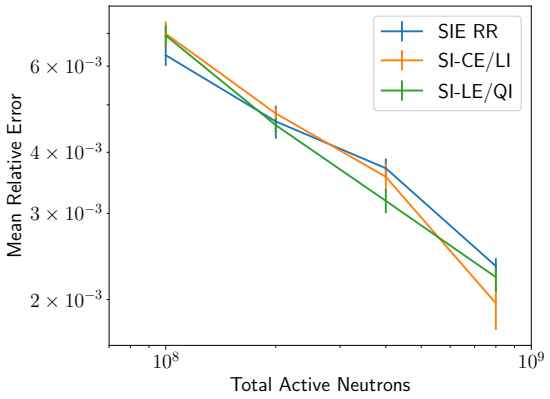


Figure 5-19: Mean coefficient of variation of ^{135}Xe for the fully reflected 3D pin problem, stochastic implicit

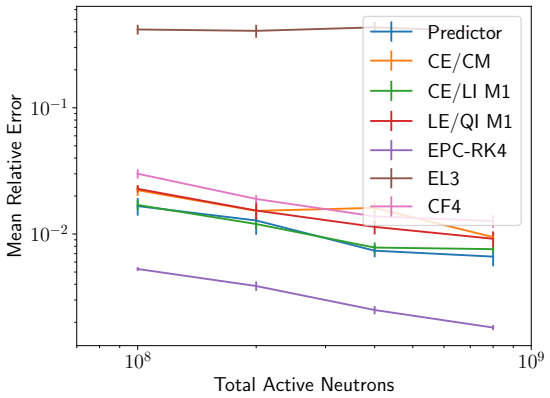


Figure 5-20: Mean coefficient of variation of ^{135}Xe for the vacuum top/bottom 3D pin problem, explicit comparison

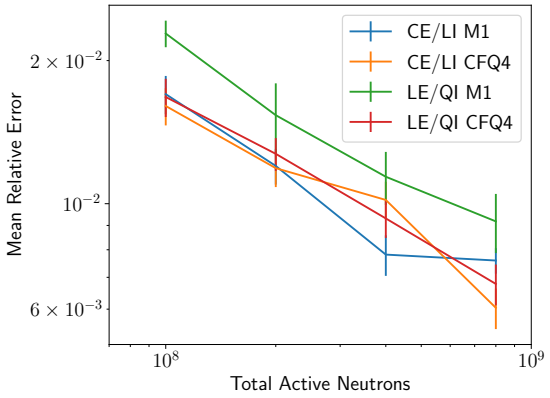


Figure 5-21: Mean coefficient of variation of ^{135}Xe for the vacuum top/bottom 3D pin problem, CE/LI and LE/QI subintegrator comparison

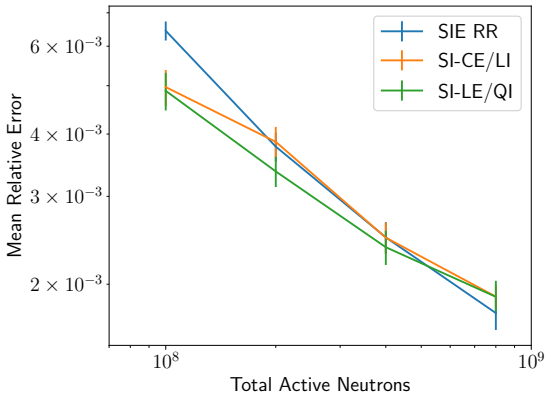


Figure 5-22: Mean coefficient of variation of ^{135}Xe for the vacuum top/bottom 3D pin problem, stochastic implicit

Chapter 6

Summary

Depletion, the analysis of the decay and transmutation of nuclear fuel, is an immensely challenging problem. In the particular case of analyzing the evolution of nuclear fuel for a full nuclear reactor core, there are issues of accuracy, stability, and memory utilization. Current techniques are often insufficient, requiring short time steps to get the right answer. In addition, it was unclear how these algorithms converged, especially in a stochastic environment.

In this document, a number of improved algorithms (SI-CE/LI, SI-LE/QI), algorithms from other fields (CF4), and purely new algorithms (EPC-RK4, EL3) were developed and tested on a variety of test problems. These tests ranged from simple toy problems such as the facsimile pin in Chapter 4 to a highly challenging 2D assembly in Section 5.1 to a geometry prone to oscillation in Section 5.2. Through these, the performance of each old and new algorithm was quantified.

With all of the analysis complete, it is worthwhile to go back to the goals of the project in Section 1.3 to see what goals were met. In that section, four goals were listed: reduction in arithmetic cost, improvement of accuracy, improvement of stability, and minimization of memory cost.

Starting with the reduction in arithmetic costs, the CFQ4 subintegrator for CE/LI and LE/QI reduced the number of matrix exponentials from a typical five to only two. In addition, the facsimile pin test indicates that the CFQ4 integrator allows LE/QI to achieve asymptotic third order convergence. Testing on the 2D fuel pin array indicates a relative insensitivity to subintegrator. As such, this switch is beneficial.

For the improvement of accuracy, one must consider deterministic transport and stochastic transport separately due to the differences in convergence for each method. For deterministic transport, the CF4 algorithm, which was developed in other fields, was found to have the highest accuracy per transport solve for ^{157}Gd and plutonium. The new EL3 algorithm was found to perform similarly to LE/QI. Both required around twice the number of transport solves

as CF4.

For stochastic transport, it was found that any algorithm, if sufficiently converged, would have errors dominated by stochastic noise. The magnitude of this stochastic noise is loosely sensitive to algorithm, with only CE/CM having a significantly larger value. However, the high accuracy methods discussed in the deterministic transport methods have larger stochastically bound regions. This makes the choice of time step easier. The new SI-CE/LI and SI-LE/QI outperform their non-implicit counterparts. While these methods require more transport solves, this is balanced by running proportionally fewer particles per simulation.

For stability, four algorithms performed especially well. The new EPC-RK4, SI-CE/LI, and SI-LE/QI had stability on par with the stochastic implicit Euler method while having far superior numerical performance. Conversely, the new EL3 had poor stability performance.

Finally, with regard to memory, it becomes a balancing act. Many of these improvements come at a cost of an increase in number of reaction rates stored. Exact numbers are listed in Section 6.1. As such, it is important to weigh the needs of accuracy and stability against the size of the problem under investigation. Compromises can be made in either direction, and it is up to the user to determine which compromises are worthwhile.

With regard to compromises, this chapter will discuss what approach one should take in choosing an algorithm. Section 6.1 goes over the memory, particle (for stochastic), and transport solve (for deterministic) costs for many of the algorithms tested. Once the cost is computed, then several recommendations are made in Section 6.2 for both deterministic and stochastic algorithms. Finally, Section 6.3 summarizes the entire project.

6.1 The Cost of Depletion

Before recommendations are made, the cost of depletion for each algorithm needs to be estimated. This will require three values. The first would be the memory requirements required to perform the depletion for each algorithm. The second would be an estimate of the number of neutrons required in a stochastic depletion. The final value would be the number of transport sweeps required to achieve a given error in a deterministic simulation.

Starting with memory, in Section 1.2.3, two reactor models were described and used to estimate the memory requirements for both the reaction rate and nuclide data sets. Each algorithm requires a different number of each in order to perform a simulation. Using these values, the memory requirements for each algorithm can be estimated. This is done in Table 6.1. This calculation is done assuming as little is stored as possible. Both EL3 and CF4 have slightly faster forms that require the storage of more nuclide vectors.

Algorithm	Nuclide Sets	Rate Sets	Model	
			Medium (GB)	High (TB)
Predictor	1	1	57	8.7
SIE RR	2	2	115	17.5
CE/CM	2	1	95	14.4
CE/LI	2	2	115	17.5
CE/LI Implicit	2	3	135	20.5
LE/QI	2	3	135	20.5
LE/QI Implicit	2	4	155	23.6
EPC-RK4	2	4	155	23.6
EL3	2	3	135	20.5
CF4	2	4	155	23.6

Table 6.1: Minimum memory requirements by algorithm, assuming none is written to disk or duplicated, for a full core

Overall, it is a balancing act. Low accuracy and low order methods such as predictor will require less memory than high order methods. CF4 requires 2.7 times as much memory as predictor, but provides a significant increase in accuracy. CE/CM is in a unique position, as it does not require the storage of the predictor reaction rate to carry out integration.

The neutron cost for each simulation is another difficult matter. It is difficult to extrapolate from one geometry to another how many neutrons will be required to achieve the same error. A very coarse estimate can be made by assuming that doubling the number of regions doubles the number of required neutrons. The values in Table 5.4 were extrapolated in such a way to generate Table 6.2. Solving the medium accuracy model to 1% on average in all cells has reasonable costs at roughly 40 billion neutrons per simulation day for gadolinium and 2.3 billion neutrons per simulation day for plutonium. Solving the high accuracy model to 0.1% on average in all cells would require 600 trillion neutrons per simulated day for gadolinium and 35 trillion neutrons per simulation day for plutonium, which is well beyond current capabilities. These values roughly double if CE/CM is used.

Finally, in order to estimate the cost of a deterministic transport, the number of transport solves to compute to 150 days is tabulated in Table 6.3. This is computed by dividing 150 days by the time step estimate in Table 5.3 and multiplying it by the number of stages in each algorithm. The results make it quite clear why the predictor method is not recommended. In addition, the fact that SIE requires 10 transport solves makes its performance exceptionally poor here. Interestingly, the extreme accuracy advantage of CF4 makes it either the fastest or almost the fastest for all error values. Due to other drawbacks, this does not make it always recommended as will be discussed in Section 6.2.1.

	Medium Acc. Model		High Acc. Model	
	^{157}Gd	$^{238\text{-}242}$Pu	^{157}Gd	$^{238\text{-}242}$Pu
CE/CM	1.27×10^{13}	6.31×10^{11}	1.93×10^{15}	9.63×10^{13}
CE/LI M1	7.90×10^{12}	3.20×10^{11}	1.21×10^{15}	4.88×10^{13}
LE/QI M1	5.83×10^{12}	3.32×10^{11}	8.88×10^{14}	5.07×10^{13}
EPC-RK4	5.46×10^{12}	3.51×10^{11}	8.32×10^{14}	5.35×10^{13}
EL3	6.10×10^{12}	3.54×10^{11}	9.30×10^{14}	5.39×10^{13}
CF4	7.14×10^{12}	3.63×10^{11}	1.09×10^{15}	5.53×10^{13}

Table 6.2: Estimated minimum number of neutrons necessary to achieve 1% accuracy on average in all cells over 150 days

	^{157}Gd			$^{238\text{-}242}$Pu		
	10%	1%	0.1%	10%	1%	0.1%
Predictor	110	1095	10949	3	28	276
SIE, RR	540	5380	53770	30	300	2980
CE/CM	20	60	190	4	12	34
CE/LI M1	20	60	188	4	10	30
LE/QI M1	14	40	124	4	8	24
LE/QI CFQ4	14	28	56	4	8	16
EPC-RK4	16	40	124	4	8	28
EL3	12	24	51	3	6	15
CF4	12	16	28	4	8	12

Table 6.3: Estimated minimum number of transport solves required to achieve given error for 150 days

6.2 Recommendations

Now that the cost of depletion is estimated, several recommendations can be made. These recommendations are split into the two different transport categories due to how each interacts with depletion. The deterministic recommendations are listed in Section 6.2.1. The stochastic recommendations are listed in Section 6.2.2.

6.2.1 Recommendations for Deterministic Simulations

The choice of algorithm for depletion with a deterministic transport solver is a fairly complicated matter. Nominally, high order methods are recommended for high accuracy and low order methods are recommended for low accuracy. In Table 6.3, however, it is estimated that CF4 will either cost less or as much as all other tested algorithms for reasonable engineering accuracies. This advantage will grow with tighter tolerances.

While this may lead one to believe that CF4 should always be used, there are four possible reasons one would rather choose a different algorithm. The first is if many time step values are necessary. Consider a user who needs to know reaction rates every 15 days during a simulation for the first 150 days on a model with no burnable absorber. The user wants to have plutonium accurate to 1% in each region on average. This 15 day time step would be sufficiently accurate for any algorithm other than the two first order methods. However, CE/LI would require 20 transport solves to the 40 for CF4 to get the temporal resolution. Under these circumstances, a method with fewer stages is recommended.

The second issue is memory. CF4 has one of the highest memory requirements of any tested algorithm. While it is not ideal, it may be necessary to choose a simpler algorithm simply to fit the model of interest into memory.

The third issue is stability. While deterministic solutions are less susceptible to initiating oscillations as compared to stochastic methods, oscillations can still occur. Algorithms other than CF4, such as EPC-RK4, were demonstrated in Section 5.2 to oscillate less in loosely coupled geometries.

The fourth issue is that CF4 is not yet available in a form that is compatible with adaptive time stepping. While CF4 with a fixed time grid may be quite efficient, an adaptive time stepping method may find a time grid that requires fewer function evaluations. An algorithm such as LE/QI can be run with adaptive time stepping by using the linear extrapolation as a second order estimate.

Based on the combination of Table 6.3 and the above four issues, it is recommended to have several algorithms. CF4 acts as the algorithm recommended for general use. LE/QI can be used when a higher temporal resolution is required.

In addition, it can be run with adaptive time stepping. Finally, for stability, one has two options. The first is to simply tighten the convergence of the transport operator. This will reduce the introduction of spurious eigenmodes. If that does not work, EPC-RK4 was more stable without the cost of iterating on an implicit algorithm.

6.2.2 Recommendations for Stochastic Simulations

In order to get a good answer with stochastic depletion, a few things need to be considered. In Section 4.1, it was noted that if the number of neutrons is held constant, there were three regions in a plot of error against time step. For very large time steps, error was numerically dominated. For very small time steps, the number of neutrons per simulation was small and bias effects appeared. For time steps in between, the error was bounded purely by stochastic noise. In addition, the error was smallest in this domain.

There are a variety of approaches to choose a time step such that the simulation falls in this stochastically bound region. Option one is to run a similar simulation and analyze how the method converges with time step, similar to what was done to generate Table 5.3. Option two is to simply take a high accuracy method and run it with a relatively short time step. For example, CF4 with a 5-day time step will be significantly more accurate than 0.1% on almost any problem. Option three is to use the stochastic-aware adaptive time stepping algorithm. In this case, the user provides an error estimate and a local tolerance for said error estimate. The algorithm will then find the optimal time step and particle combination to achieve this. Of course, this is predicated on the availability of a depletion algorithm with a reliable high and low order solution estimate. This would be limited to EL(3,2) and the predictor-corrector methods of the ones tested in this document.

Once in this stochastically bounded region, the actual magnitude of the error is only weakly sensitive to the algorithm. The most significant outlier is the CE/CM method, which yielded errors that were approximately a factor of 1.4 larger than other tested algorithms. In order to get the same error, CE/CM would require almost twice the number of neutrons as shown in Table 6.2. One should focus their attention on algorithms that were optimal or nearly so, such as CE/LI, LE/QI, EL3, and CF4.

The other way the error in the stochastically bound region was sensitive to algorithm is through instabilities. In the loosely coupled fuel rod tests in Section 5.2, many algorithms were found to oscillate. It was also noted in the more realistic problem that the solution does converge as the number of neutrons is increased. On one hand, a low neutron simulation would benefit from a more stable algorithm. On the other hand, a high accuracy simulation would require so many neutrons as to make oscillations irrelevant.

With all of this combined, a few criteria for a general default algorithm can be made. The first criteria is that the stochastically bound region is broad. Then, a user can simply choose a short time step and it will probably work. For this recommendation, a target accuracy of 0.1% with a 5-day time step is used. The second criteria is that the algorithm needs to propagate errors nearly optimally. The third criteria is that it needs to use as little memory as possible. Finally, while not as strictly a requirement as the others, more stable algorithms are preferred if it does not interfere with the others significantly.

With that in mind, there are two recommendations. For geometries without burnable absorber, the SI-CE/LI algorithm is recommended. For geometries with burnable absorber, the SI-LE/QI algorithm is recommended. For their respective problems, either algorithm would be near 0.1% error with a 5-day time step. The base algorithms propagate errors either optimally or nearly optimally. Both algorithms also had significantly reduced oscillations in the 3D pincell test.

Both algorithms have three downsides. In memory utilization, both algorithms are near the top due to the need to store the average reaction rate in memory. The non-implicit forms may make better choices under these circumstances. The second problem is that instead of running one simulation with a large number of neutrons, several simulations are run with a small number of neutrons. Care must be taken to avoid issues such as bias (eigenvalue, tally) or neutron clustering. The third problem is that since the predictor is not implicit, it is unclear how well adaptive time stepping would work.

6.3 Conclusions

Throughout this document, a number of new depletion techniques have been developed. These range from the appropriate way to perform a simulation to new algorithms for the depletion neutronics coupling. In general, this should provide a substantial cost savings as compared to current methods.

In the case of a deterministic transport solver, a number of algorithms are recommended in order to fill a variety of niches. The CF4 algorithm is an exceptionally efficient fourth order algorithm that can compute ^{157}Gd to 0.1% accuracy in less than half the time of the best current method. LE/QI is more expensive overall, but each individual time step is half the cost of CF4. For high temporal resolution, LE/QI can outperform CF4. In addition, LE/QI can be run with adaptive time stepping. Finally, for highly oscillatory problems, EPC-RK4 is reasonably accurate but very stable.

From a stochastic transport point of view, the major contribution is that once the time step is short enough, the problem is dominated by stochastic noise. In this region, so long as the total number of neutrons (and as a consequence, the simulation cost) is kept fixed, the error is insensitive to the time step. Of the

algorithms tested, the two recommendations are SI-CE/LI and SI-LE/QI for systems without and with burnable absorbers respectively. For either algorithm, a time step on the order of 5 days should be sufficient to achieve 0.1% numeric accuracy. The user can then adjust the number of neutrons to reach their target accuracy and simulation cost. Both algorithms were stable in all tested problems.

Based on these results, the cost for a high fidelity full core reactor is still quite large. Memory requirements are in the several terabytes. For stochastic methods, several quadrillion neutrons would be required. With current compute capability and with current codes, this is still outside what can be economically achieved. However, the techniques in this document can help reduce the cost for both the deterministic and stochastic applications.

Appendix A

Exponential-Linear Coefficient Tables

A.1 EL3

Coeff.	Value
c_1	0.0
d_{11}	1.0
a_{111}	$4.546\,892\,735\,758\,679\,705\,861\,255\,753\,176\,861\,047\,623\,431\,082\,543\,1 \times 10^{-1}$
c_2	$4.546\,892\,735\,758\,679\,705\,861\,255\,753\,176\,861\,047\,623\,431\,082\,543\,1 \times 10^{-1}$
d_{21}	$4.917\,209\,077\,482\,205\,644\,821\,739\,863\,302\,177\,074\,032\,672\,038\,040\,4 \times 10^{-1}$
a_{211}	$-9.357\,875\,044\,996\,728\,149\,500\,565\,490\,588\,424\,670\,599\,156\,421\,003\,4 \times 10^{-2}$
a_{212}	$8.796\,664\,339\,946\,539\,972\,637\,417\,207\,891\,215\,946\,976\,046\,028\,785\,8 \times 10^{-1}$
d_{22}	$5.082\,790\,922\,517\,794\,355\,178\,260\,136\,697\,822\,925\,967\,327\,961\,959\,6 \times 10^{-1}$
a_{221}	$-5.901\,222\,600\,691\,500\,940\,353\,488\,338\,871\,417\,273\,874\,284\,061\,489\,4 \times 10^{-1}$
a_{222}	$9.215\,206\,700\,379\,688\,392\,179\,593\,244\,526\,929\,706\,166\,983\,365\,631\,8 \times 10^{-1}$
c_3	$7.860\,876\,835\,446\,867\,157\,687\,360\,658\,832\,373\,479\,916\,130\,386\,685\,5 \times 10^{-1}$
d_{31}	$2.037\,856\,279\,024\,803\,972\,583\,036\,752\,857\,021\,136\,584\,569\,765\,008\,0 \times 10^{-2}$
a_{311}	$2.323\,861\,655\,916\,654\,467\,634\,713\,548\,382\,658\,761\,300\,953\,811\,054\,9 \times 10^{-1}$
a_{312}	$1.815\,990\,852\,230\,668\,926\,845\,511\,014\,751\,732\,983\,788\,154\,184\,427\,2 \times 10^{-1}$
a_{313}	$5.860\,147\,491\,852\,676\,605\,519\,775\,436\,865\,608\,254\,910\,892\,004\,517\,8 \times 10^{-1}$
d_{32}	$5.023\,605\,249\,880\,540\,168\,810\,241\,591\,182\,872\,807\,325\,254\,546\,908\,8 \times 10^{-1}$
a_{321}	$1.105\,777\,638\,860\,611\,222\,846\,680\,839\,492\,034\,927\,985\,431\,499\,018\,0 \times 10^{-2}$
a_{322}	$2.782\,278\,850\,129\,721\,676\,513\,356\,131\,877\,802\,470\,951\,630\,866\,514\,2 \times 10^{-2}$
a_{323}	$5.064\,301\,615\,342\,287\,004\,202\,740\,549\,686\,155\,212\,482\,862\,680\,903\,7 \times 10^{-1}$
d_{33}	$4.772\,609\,122\,216\,979\,433\,931\,454\,733\,531\,425\,079\,016\,288\,476\,590\,4 \times 10^{-1}$
a_{331}	$2.721\,240\,234\,997\,680\,146\,295\,252\,007\,220\,932\,963\,372\,603\,803\,292\,6 \times 10^{-2}$
a_{332}	$-1.076\,902\,478\,515\,720\,891\,022\,361\,378\,465\,291\,151\,057\,896\,168\,294\,6 \times 10^{-1}$
a_{333}	$2.943\,901\,619\,569\,085\,718\,705\,475\,518\,910\,824\,374\,804\,505\,401\,279\,8 \times 10^{-1}$

Table A.1: Coefficients for the exponential-linear 3rd order method EL3

Coeff.	Value
c_4	1.0
d_{41}	$6.990\,258\,593\,378\,940\,649\,862\,735\,967\,080\,976\,455\,200\,406\,842\,493 \times 10^{-1}$
a_{411}	$3.897\,195\,273\,106\,978\,550\,779\,025\,985\,311\,777\,787\,350\,723\,835\,169\,2 \times 10^{-1}$
a_{412}	$7.557\,833\,648\,862\,195\,338\,734\,610\,316\,191\,851\,543\,002\,939\,384\,986\,0 \times 10^{-2}$
a_{413}	$5.347\,021\,339\,187\,491\,270\,217\,180\,114\,416\,437\,852\,204\,686\,115\,263\,3 \times 10^{-1}$
d_{43}	$3.009\,741\,406\,621\,059\,350\,137\,264\,032\,919\,023\,544\,799\,593\,157\,506\,9 \times 10^{-1}$
a_{431}	$2.115\,123\,490\,367\,608\,864\,058\,399\,548\,634\,247\,261\,854\,805\,895\,801\,5 \times 10^{-1}$
a_{432}	$-5.845\,112\,888\,291\,053\,597\,274\,740\,008\,958\,756\,383\,146\,017\,300\,415\,5 \times 10^{-1}$
a_{433}	$5.869\,112\,615\,475\,443\,751\,692\,299\,663\,051\,443\,487\,757\,787\,743\,367\,5 \times 10^{-1}$

Table A.2: Additional coefficients for the second order component of the non-FSAL version of EL(3,2). All non-listed coefficients are zero.

Coeff.	Value
c_4	1.0
d_{41}	$6.822\,665\,910\,762\,302\,662\,348\,023\,370\,164\,057\,082\,200\,375\,847\,107\,347 \times 10^{-1}$
a_{411}	$-1.350\,235\,042\,574\,305\,689\,359\,845\,747\,787\,568\,537\,370\,730\,886\,394\,1 \times 10^{-1}$
a_{412}	$1.119\,140\,138\,384\,711\,083\,120\,160\,786\,612\,001\,867\,255\,264\,808\,072\,2$
a_{413}	$1.096\,892\,280\,325\,781\,829\,195\,189\,884\,557\,805\,821\,847\,388\,408\,673\,5 \times 10^{-1}$
a_{414}	$-9.380\,586\,215\,985\,869\,710\,369\,520\,028\,902\,559\,570\,293\,056\,030\,017\,5 \times 10^{-2}$
d_{42}	$3.177\,334\,089\,237\,697\,337\,651\,976\,629\,835\,942\,917\,799\,624\,152\,892\,653 \times 10^{-1}$
a_{421}	$7.732\,905\,236\,163\,448\,948\,799\,947\,492\,958\,133\,552\,497\,729\,754\,021\,9 \times 10^{-1}$
a_{422}	$-1.572\,203\,178\,401\,681\,105\,923\,445\,283\,677\,106\,160\,877\,737\,488\,410\,4$
a_{423}	$6.175\,851\,975\,066\,408\,257\,446\,666\,251\,602\,666\,453\,517\,719\,157\,713\,2 \times 10^{-1}$
a_{424}	$7.266\,381\,837\,028\,274\,147\,126\,583\,339\,033\,400\,555\,138\,494\,889\,825\,7 \times 10^{-1}$

Table A.3: Additional coefficients for the second order component of the FSAL version of EL(3,2). All non-listed coefficients are zero.

Coeff.	Value	Coeff.	Value
c_1	0.0	a_{333}	$2.240\,759\,630\,039\,589$
d_{11}	1.0	c_4	1.0
a_{111}	$2.638\,017\,781\,099\,526\,4 \times 10^{-1}$	d_{41}	$-2.540\,101\,046\,715\,893\,8 \times 10^{-2}$
c_2	$2.638\,017\,781\,099\,526\,4 \times 10^{-1}$	a_{411}	$6.342\,361\,480\,700\,457 \times 10^{-1}$
d_{21}	$4.714\,899\,766\,145\,780\,3 \times 10^{-1}$	a_{412}	$-1.426\,165\,912\,825\,637\,6$
a_{211}	$-1.096\,345\,914\,231\,227\,6 \times 10^{-1}$	a_{413}	$-7.209\,962\,986\,478\,266 \times 10^{-1}$
a_{212}	$7.549\,479\,388\,690\,35 \times 10^{-1}$	a_{414}	$2.512\,926\,068\,677\,481$
d_{22}	$5.285\,100\,233\,854\,22 \times 10^{-1}$	d_{42}	$2.913\,365\,964\,654\,815\,5 \times 10^{-1}$
a_{221}	$-8.139\,969\,413\,877\,527 \times 10^{-1}$	a_{421}	$5.602\,130\,520\,260\,26 \times 10^{-1}$
a_{222}	$1.195\,508\,497\,529\,188\,3$	a_{422}	$-1.036\,247\,635\,307\,391\,7$
c_3	$6.453\,133\,474\,459\,122\,4 \times 10^{-1}$	a_{423}	$1.403\,357\,266\,739\,732\,5$
d_{31}	$2.333\,112\,759\,614\,89 \times 10^{-1}$	a_{424}	$-1.911\,244\,663\,312\,152\,1 \times 10^{-1}$
a_{311}	$2.432\,927\,685\,490\,108$	d_{43}	$6.387\,934\,650\,493\,379 \times 10^{-1}$
a_{312}	$-1.886\,991\,744\,360\,153\,8$	a_{431}	$1.138\,564\,243\,974\,421\,3 \times 10^{-1}$
a_{313}	$4.540\,639\,985\,471\,296 \times 10^{-1}$	a_{432}	$1.137\,278\,934\,630\,576\,9 \times 10^{-1}$
d_{32}	$5.526\,116\,522\,082\,521 \times 10^{-1}$	a_{433}	$-3.355\,485\,694\,559\,844\,4 \times 10^{-1}$
a_{321}	$1.440\,240\,011\,283\,619\,1$	a_{434}	$4.626\,509\,125\,349\,493\,3 \times 10^{-1}$
a_{322}	$-1.999\,581\,093\,585\,001\,1$	d_{44}	$9.527\,094\,895\,233\,958 \times 10^{-2}$
a_{323}	$1.295\,539\,340\,166\,664$	a_{441}	$-1.138\,311\,740\,251\,085$
d_{33}	$2.140\,770\,718\,302\,588\,4 \times 10^{-1}$	a_{442}	$4.998\,539\,153\,859\,359\,3 \times 10^{-1}$
a_{331}	$-3.341\,457\,198\,009\,325\,5 \times 10^{-1}$	a_{443}	$1.196\,593\,771\,894\,506\,6$
a_{332}	$-1.551\,927\,277\,833\,745$	a_{444}	$-5.581\,359\,405\,254\,164 \times 10^{-1}$

Table A.4: Coefficients for the exponential-linear 4th order method EL4

Appendix B

The Facsimile Pin Equations

B.1 Un-Normalized Reaction Rate Reconstruction

Given a ^{135}Xe atom density X, the mean and covariance of the un-normalized reaction rates is given by the following equations. The coefficients are listed in Table B.1 through Table B.3. Since Σ is already in an eigendecomposed form, sampling R is made easy.

$$
\begin{aligned}
x &= \ln(X) \\
\hat{R}(x) &= \exp\left(\sum_{i=0}^{8} C_{R,i} x^{8-i}\right) \\
\lambda(x) &= \exp\left(\sum_{i=0}^{8} C_{\lambda,i} x^{8-i}\right) \\
m(x) &= \sum_{i=0}^{8} C_{m,i} x^{8-i}
\end{aligned}
$$

Ind	$\hat{R}_f^{235}$	$\hat{R}_\gamma^{235}$	$\hat{R}_\gamma^{135}$
0	$8.983\,922\,698\,496\,774 \times 10^{-13}$	$2.413\,780\,444\,082\,758 \times 10^{-11}$	$-1.061\,237\,508\,172\,745 \times 10^{-10}$
1	$-1.317\,544\,526\,257\,764 \times 10^{-9}$	$-4.782\,694\,269\,840\,324 \times 10^{-9}$	$1.563\,867\,426\,337\,918 \times 10^{-8}$
2	$1.596\,838\,385\,010\,340 \times 10^{-7}$	$3.727\,656\,629\,199\,686 \times 10^{-7}$	$-9.496\,041\,500\,720\,049 \times 10^{-7}$
3	$-8.013\,434\,637\,221\,426 \times 10^{-6}$	$-1.500\,278\,341\,380\,614 \times 10^{-5}$	$3.071\,346\,867\,571\,181 \times 10^{-5}$
4	$2.072\,237\,944\,231\,544 \times 10^{-4}$	$3.394\,134\,519\,764\,877 \times 10^{-4}$	$-5.714\,021\,823\,222\,694 \times 10^{-4}$
5	$-2.910\,174\,022\,481\,489 \times 10^{-3}$	$-4.365\,863\,728\,168\,456 \times 10^{-3}$	$6.164\,844\,857\,335\,721 \times 10^{-3}$
6	$2.160\,531\,158\,788\,778 \times 10^{-2}$	$3.057\,680\,760\,917\,871 \times 10^{-2}$	$-3.697\,896\,887\,692\,661 \times 10^{-2}$
7	$-7.605\,236\,853\,131\,267 \times 10^{-2}$	$-1.038\,615\,103\,462\,047 \times 10^{-1}$	$1.103\,106\,475\,773\,091 \times 10^{-1}$
8	$7.502\,361\,490\,000\,785$	$5.969\,314\,332\,406\,269$	$1.577\,288\,772\,806\,987 \times 10^{1}$

Table B.1: Polynomial Coefficients ($C_{R,i}$) for μ

$$
\begin{aligned}
m_{2,0} &= 1 - m_{0,0} - m_{1,0} \\
m_{1,1} &= \frac{m_{2,0} - m_{2,0}m_{0,1} + m_{0,0}m_{0,1}}{m_{2,0} - m_{1,0}} \\
m_{2,1} &= 1 - m_{0,1} - m_{1,1} \\
m_{0,2} &= -\frac{-m_{1,1}m_{2,0} + m_{1,0}m_{2,1}}{m_{0,1}m_{1,0} - m_{0,0}m_{1,1} - m_{0,1}m_{2,0} + m_{1,1}m_{2,0} + m_{0,0}m_{2,1} - m_{1,0}m_{2,1}} \\
m_{1,2} &= -\frac{-m_{0,1}m_{2,0} + m_{0,0}m_{2,1}}{-m_{0,1}m_{1,0} + m_{0,0}m_{1,1} + m_{0,1}m_{2,0} - m_{1,1}m_{2,0} - m_{0,0}m_{2,1} + m_{1,0}m_{2,1}} \\
m_{2,2} &= 1 - m_{0,2} - m_{1,2}
\end{aligned}
$$

$$
\begin{aligned}
\mu_1 &= \hat{R}_f^{235} \\
\mu_2 &= \hat{R}_\gamma^{235} \\
\mu_3 &= \hat{R}_\gamma^{135} \\
\Lambda_{i,i} &= \lambda_i(x) \\
Q_{i,j} &= \frac{m_{i,j}}{\sqrt{m_{0,j}^2 + m_{1,j}^2 + m_{2,j}^2}} \\
\Sigma &= Q\Lambda Q^T
\end{aligned}
$$

Ind	λ_1	λ_2	λ_3
0	$2.640\,248\,236\,467\,282 \times 10^{-10}$	$2.387\,131\,454\,133\,477 \times 10^{-10}$	$2.335\,213\,463\,078\,587 \times 10^{-10}$
1	$-5.528\,411\,280\,448\,152 \times 10^{-8}$	$-5.043\,954\,792\,963\,431 \times 10^{-8}$	$-4.094\,476\,499\,610\,042 \times 10^{-8}$
2	$4.635\,561\,239\,439\,506 \times 10^{-6}$	$4.366\,212\,890\,716\,993 \times 10^{-6}$	$2.968\,277\,717\,357\,294 \times 10^{-6}$
3	$-2.037\,472\,727\,294\,892 \times 10^{-4}$	$-2.014\,337\,799\,435\,205 \times 10^{-4}$	$-1.147\,477\,330\,672\,807 \times 10^{-4}$
4	$5.106\,204\,734\,219\,635 \times 10^{-3}$	$5.367\,376\,954\,493\,614 \times 10^{-3}$	$2.540\,343\,891\,279\,834 \times 10^{-3}$
5	$-7.376\,474\,181\,131\,773 \times 10^{-2}$	$-8.326\,718\,948\,125\,267 \times 10^{-2}$	$-3.209\,032\,536\,195\,246 \times 10^{-2}$
6	$5.867\,940\,315\,753\,537 \times 10^{-1}$	$7.155\,898\,207\,045\,706 \times 10^{-1}$	$2.164\,980\,698\,544\,974 \times 10^{-1}$
7	$-2.263\,592\,494\,654\,577$	$-2.972\,750\,253\,829\,436$	$-6.607\,055\,253\,352\,923 \times 10^{-1}$
8	$2.644\,730\,960\,562\,609 \times 10^{1}$	$7.900\,322\,026\,922\,797$	$8.995\,683\,247\,188\,696 \times 10^{-1}$

Table B.2: Polynomial Coefficients ($C_{\lambda,i}$) for λ in Σ

Ind	$m_{0,0}$	$m_{1,0}$	$m_{0,1}$
0	$1.830\,319\,642\,576\,281 \times 10^{-14}$	$4.302\,087\,575\,124\,011 \times 10^{-15}$	$-6.047\,943\,038\,256\,519 \times 10^{-12}$
1	$-2.821\,275\,947\,510\,134 \times 10^{-12}$	$-6.690\,250\,872\,516\,028 \times 10^{-13}$	$7.674\,138\,384\,563\,984 \times 10^{-10}$
2	$1.753\,220\,028\,372\,874 \times 10^{-10}$	$4.222\,834\,871\,982\,091 \times 10^{-11}$	$-3.293\,844\,721\,796\,796 \times 10^{-8}$
3	$-5.597\,419\,456\,151\,630 \times 10^{-9}$	$-1.384\,596\,587\,654\,212 \times 10^{-9}$	$2.972\,080\,518\,967\,480 \times 10^{-7}$
4	$9.650\,935\,268\,971\,758 \times 10^{-8}$	$2.498\,355\,034\,922\,093 \times 10^{-8}$	$1.836\,700\,919\,899\,789 \times 10^{-5}$
5	$-8.547\,115\,484\,371\,490 \times 10^{-7}$	$-2.398\,901\,742\,467\,470 \times 10^{-7}$	$-6.094\,118\,807\,155\,754 \times 10^{-4}$
6	$3.141\,678\,729\,062\,863 \times 10^{-6}$	$1.045\,658\,916\,647\,399 \times 10^{-6}$	$7.284\,695\,061\,253\,409 \times 10^{-3}$
7	$-4.232\,730\,218\,422\,221 \times 10^{-7}$	$-9.452\,745\,141\,054\,007 \times 10^{-7}$	$-3.460\,772\,200\,962\,809 \times 10^{-2}$
8	$1.576\,100\,569\,410\,218 \times 10^{-4}$	$2.683\,036\,673\,142\,188 \times 10^{-5}$	$8.385\,988\,387\,835\,685 \times 10^{-1}$

Table B.3: Polynomial Coefficients ($C_{m,i}$) for Reconstructing Q in Σ

B.2 Equations

The ODE describing the time evolution of this system is given below. Here, $\beta_{n\to m}$ is shorthand. If m corresponds with "Target 1", then it equals the branching ratio. If m equals "Target 2", it equals the one minus the branching ratio. Otherwise, it is zero.

$$\begin{aligned}
y'(t) &= A(y,t)y(t) \\
V &= \pi(0.39218)^2 \\
y_0(0) &= 5.5818256467 \times 10^{20} V \\
y_8(0) &= 1.0 \times 10^{15} V \\
y_n(0) &= 0.0 && \forall n, n \neq 0, 8 \\
A_{0,0}(y,t) &= -(R_f^{235}(y,t) + R_\gamma^{235}(y,t)) \\
A_{n,0}(y,t) &= \gamma_n R_f^{235}(y_8(t)) && \forall n > 0 \\
A_{8,8}(y,t) &= -\lambda_8 - R_\gamma^{135}(y,t) \\
A_{n,n}(y,t) &= -\lambda_n && \forall n > 0, n \neq 8 \\
A_{n,m}(y,t) &= \lambda_n \beta_{n\to m} && \forall n, m, n \neq m
\end{aligned}$$

The values R are the power-normalized product of microscopic cross section and flux, and is computed as follows, where the choice in P depends on the difficulty needed:

$$\begin{aligned}
P_{\text{const}}(t) &= 166 \\
P_{\cos}(t) &= 166\left[1 + \cos\left(\frac{\pi t}{86400}\right)\right] \\
\hat{P}(y,t) &= 3.099 \times 10^{-11} \hat{R}_f^{235}\left(\frac{y_8(t)}{V}\right) y_0(t) \\
R(y,t) &= \hat{R}\left(\frac{y_8(t)}{V}\right)\frac{P(t)}{\hat{P}(y,t)}
\end{aligned}$$

Finally, $\hat{R}$ is the unnormalized $\sigma\phi$ product, and is sampled from the distributions that were curve fit prior, which are given in Section B.1.

The solution at 100 hours as calculated using Mathematica is shown in Table B.2. Integration was performed with a working precision of 50 digits and an accuracy/precision goal of 18 digits using the `StiffnessSwitching` algorithm.

Nuclide	Total Number, P_{const}	Total Number, $P_{\cos}$
^{235}U	$2.673\,777\,777\,000\,643 \times 10^{20}$	$2.672\,890\,201\,627\,905 \times 10^{20}$
^{136}Sn	$3.032\,985\,139\,924\,452 \times 10^{5}$	$5.659\,647\,207\,941\,255 \times 10^{5}$
^{135}Sb	$1.883\,375\,251\,975\,68 \times 10^{10}$	$3.514\,508\,994\,529\,071 \times 10^{10}$
^{136}Sb	$8.195\,068\,290\,275\,61 \times 10^{8}$	$1.529\,240\,399\,570\,006 \times 10^{9}$
^{135}Te	$4.896\,600\,155\,926\,137 \times 10^{12}$	$9.139\,623\,789\,398\,314 \times 10^{12}$
^{136}Te	$1.789\,660\,033\,949\,795 \times 10^{12}$	$3.340\,371\,442\,443\,112 \times 10^{12}$
^{135}I	$1.147\,752\,958\,065\,247 \times 10^{16}$	$1.819\,431\,171\,895\,863 \times 10^{16}$
$^{135\text{m}}$Xe	$8.612\,294\,020\,844\,927 \times 10^{13}$	$1.394\,482\,839\,090\,513 \times 10^{14}$
^{135}Xe	$2.941\,846\,195\,896\,216 \times 10^{15}$	$2.579\,397\,246\,698\,126 \times 10^{15}$
^{135}Cs	$1.996\,046\,816\,936\,932 \times 10^{16}$	$2.630\,331\,215\,881\,463 \times 10^{16}$
^{135}Ba	$7.675\,071\,291\,487\,011 \times 10^{8}$	$8.078\,015\,375\,036\,282 \times 10^{8}$

Table B.4: Facsimile Pin Total Atoms, 100 Hours, Reference Solution

Appendix C

Initial Condition for Full Depletion Problem

The geometry used is a 3x3 assembly that has been reflected through the center pin to reduce complexity. All pins are 4.5% enriched UO_2. The center pin additionally contains 2% by weight natural gadolinium. All pins are discretized into 10 rings, each with equal volume. The geometry is 2D. The exact geometry is given in Table C.1. The compostion of the fuel is listed in Table C.2. The composition of the clad, gap, and coolant is listed in Table C.3, Table C.4, and Table C.5, respectively.

Dimension	Value (cm)
Pitch	1.26197
r_{fuel}	0.412275
r_{gap}	0.418987
r_{clad}	0.476121

Table C.1: Geometry for Fuel

Nuclide	Center Pin at/cm^3	Other Pin at/cm^3
^{234}U	$8.239\,937\,430\,68 \times 10^{18}$	$8.408\,099\,419\,06 \times 10^{18}$
^{235}U	$1.025\,597\,003\,37 \times 10^{21}$	$1.046\,527\,554\,46 \times 10^{21}$
^{238}U	$2.148\,242\,346\,21 \times 10^{22}$	$2.192\,084\,026\,75 \times 10^{22}$
^{16}O	$4.492\,537\,042\,59 \times 10^{22}$	$4.584\,221\,472\,03 \times 10^{22}$
^{17}O	$1.706\,732\,538\,54 \times 10^{19}$	$1.741\,563\,814\,84 \times 10^{19}$
^{18}O	$9.008\,305\,462 \times 10^{19}$	$9.192\,148\,430\,62 \times 10^{19}$
^{152}Gd	$1.577\,444\,818\,62 \times 10^{18}$	
^{154}Gd	$1.719\,414\,852\,3 \times 10^{19}$	
^{155}Gd	$1.167\,309\,165\,78 \times 10^{20}$	
^{156}Gd	$1.614\,514\,771\,86 \times 10^{20}$	
^{157}Gd	$1.234\,350\,570\,57 \times 10^{20}$	
^{158}Gd	$1.959\,186\,464\,73 \times 10^{20}$	
^{160}Gd	$1.724\,147\,186\,76 \times 10^{20}$	

Table C.2: Fuel Composition, Full Depletion Problem

Nuclide	at/cm^3
^{16}O	$3.074\,27 \times 10^{20}$
^{17}O	$7.488\,68 \times 10^{17}$
^{50}Cr	$3.296\,20 \times 10^{18}$
^{52}Cr	$6.356\,39 \times 10^{19}$
^{53}Cr	$7.207\,63 \times 10^{18}$
^{54}Cr	$1.794\,13 \times 10^{18}$
^{54}Fe	$5.573\,50 \times 10^{18}$
^{56}Fe	$8.749\,21 \times 10^{19}$
^{57}Fe	$2.020\,57 \times 10^{18}$
^{58}Fe	$2.689\,01 \times 10^{17}$
^{50}Cr	$3.296\,20 \times 10^{18}$
^{52}Cr	$6.356\,39 \times 10^{19}$
^{53}Cr	$7.207\,63 \times 10^{18}$
^{54}Cr	$1.794\,13 \times 10^{18}$
^{58}Ni	$2.516\,31 \times 10^{19}$
^{60}Ni	$9.692\,78 \times 10^{18}$
^{61}Ni	$4.213\,38 \times 10^{17}$
^{62}Ni	$1.343\,41 \times 10^{18}$
^{64}Ni	$3.431\,27 \times 10^{17}$
^{90}Zr	$2.183\,20 \times 10^{22}$
^{91}Zr	$4.761\,04 \times 10^{21}$
^{92}Zr	$7.277\,34 \times 10^{21}$
^{94}Zr	$7.374\,94 \times 10^{21}$
^{96}Zr	$1.188\,14 \times 10^{21}$
^{112}Sn	$4.673\,52 \times 10^{18}$
^{114}Sn	$3.179\,92 \times 10^{18}$
^{115}Sn	$1.638\,14 \times 10^{18}$
^{116}Sn	$7.005\,46 \times 10^{19}$
^{117}Sn	$3.700\,27 \times 10^{19}$
^{118}Sn	$1.166\,94 \times 10^{20}$
^{119}Sn	$4.138\,72 \times 10^{19}$
^{120}Sn	$1.569\,73 \times 10^{20}$
^{122}Sn	$2.230\,76 \times 10^{19}$
^{124}Sn	$2.789\,66 \times 10^{19}$

Table C.3: Clad Composition, Full Depletion Problem

Nuclide	at/cm^3
^{16}O	$7.865\,48 \times 10^{18}$
^{17}O	$2.995\,48 \times 10^{15}$
^{14}N	$3.386\,46 \times 10^{19}$
^{15}N	$1.237\,17 \times 10^{17}$

Table C.4: Gap Composition, Full Depletion Problem

Nuclide	at/cm^3
^{1}H	$4.680\,63 \times 10^{22}$
^{16}O	$2.334\,27 \times 10^{22}$
^{17}O	$8.890\,86 \times 10^{18}$

Table C.5: Coolant Composition, Full Depletion Problem

Bibliography

[1] Default isomeric branching ratios. [Online; accessed 2-Jun-2017]. URL: `http://serpent.vtt.fi/mediawiki/index.php/Default_isomeric_branching_ratios`.

[2] Awad H Al-Mohy and Nicholas J Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3):970–989, 2009.

[3] JR Askew. A characteristics formulation of the neutron transport equation in complicated geometries. Technical report, United Kingdom Atomic Energy Authority, 1972.

[4] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[5] Sergio Blanes, Fernando Casas, JA Oteo, and José Ros. The Magnus expansion and some of its applications. *Physics Reports*, 470(5):151–238, 2009.

[6] P Bogacki and Lawrence F Shampine. An efficient Runge-Kutta (4, 5) pair. *Computers & Mathematics with Applications*, 32(6):15–28, 1996.

[7] Forrest Brown. A review of Monte Carlo criticality calculations – convergence, bias, statistics. *Los Alamos National Laboratory, Los Alamos, NM, LA-UR-08-6558*, 2008.

[8] John Charles Butcher. On fifth and sixth order explicit Runge–Kutta methods: order conditions and order barriers. *Canadian Applied Mathematics Quarterly*, 17(3):433–445, 2009.

[9] John Charles Butcher. *Runge–Kutta Methods*, chapter 3, pages 143–331. John Wiley & Sons, Ltd, 2016. `doi:10.1002/9781119121534.ch3`.

[10] David C Carpenter. A comparison of constant power depletion algorithms. In *International Conference on Mathematics and Computational Methods & Reactor Physics*, 2009.

[11] LL Carter and ED Cashwell. Particle-transport simulation with the Monte Carlo method. Technical report, Los Alamos Scientific Laboratory, Los Alamos, NM, TID-26607, 1975.

[12] Jeff R Cash and Alan H Karp. A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software (TOMS)*, 16(3):201–222, 1990.

[13] Elena Celledoni, Arne Marthinsen, and Brynjulf Owren. Commutator-free Lie group methods. *Future Generation Computer Systems*, 19(3):341–352, 2003.

[14] MB Chadwick, M Herman, P Obložinskỳ, Michael E Dunn, Y Danon, AC Kahler, Donald L Smith, B Pritychenko, Goran Arbanas, R Arcilla, et al. ENDF/B-VII.1 nuclear data for science and technology: cross sections, covariances, fission product yields and decay data. *Nuclear Data Sheets*, 112(12):2887–2996, 2011.

[15] Jan Dufek and J Eduard Hoogenboom. Numerical stability of existing Monte Carlo burnup codes in cycle calculations of critical reactors. *Nuclear science and engineering*, 162(3):307–311, 2009.

[16] Jan Dufek, Dan Kotlyar, and Eugene Shwageraus. The stochastic implicit Euler method–a stable coupling scheme for Monte Carlo burnup calculations. *Annals of Nuclear Energy*, 60:295–300, 2013.

[17] Eric Dumonteil, Fausto Malvagi, Andrea Zoia, Alain Mazzolo, Davide Artusio, Cyril Dieudonné, and Clélia De Mulatier. Particle clustering in Monte Carlo criticality simulations. *Annals of Nuclear Energy*, 63:612–618, 2014.

[18] Matthew S. Ellis, Colin Josey, Benoit Forget, and Kord Smith. Spatially continuous depletion algorithm for Monte Carlo simulations. *Transactions of the American Nuclear Society*, 115:1221–1224, 2016.

[19] John T Goorley, Michael R James, Thomas E Booth, Forrest Brown, J Bull, Lawrence J Cox, J Durkee, J Elson, M Fensin, RA Forster, et al. Initial MCNP6 release overview-MCNP6 version 1.0. *Los Alamos National Laboratory, Los Alamos, NM, LA-UR-13-22934*, 2013.

[20] David P Griesheimer, David C Carpenter, and Mark H Stedry. Practical techniques for large-scale Monte Carlo reactor depletion calculatons. *Progress in Nuclear Energy*, 2017.

[21] Wim Haeck, B. Cochet, and L. Aguiar. A burnup-dependent isomeric production branching ratio treatment. *Nuclear Science and Engineering*, 171(1):52–68, 2012. doi:10.13182/NSE10-99.

[22] Wim Haeck and Bernard Verboomen. An optimum approach to Monte Carlo burnup. *Nuclear Science and Engineering*, 156(2):180–196, 2007.

[23] John S Hendricks, Gregg W McKinney, Michael L Fensin, Michael R James, Russell C Johns, Joe W Durkee, Joshua P Finch, Denise B Pelowitz, Laurie S Waters, M William Johnson, et al. MCNPX 2.6.0 extensions. *Los Alamos National Laboratory, Los Alamos, NM, LA-UR-08-2216*, 2008.

[24] Bryan R Herman, Benoit Forget, Kord Smith, Paul K Romano, Thomas M Sutton, Daniel J Kelly, and Brian N Aviles. Analysis of tally correlation in large light water reactors. In *Proceedings of the International Conference on the Physics of Reactors (PHYSOR-2014)*, 2014.

[25] Nicholas J Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1179–1193, 2005.

[26] Arieh Iserles, Hans Z Munthe-Kaas, Syvert P Nørsett, and Antonella Zanna. Lie-group methods. *Acta Numerica 2000*, 9:215–365, 2000.

[27] Aarno E Isotalo and PA Aarnio. Higher order methods for burnup calculations with bateman solutions. *Annals of Nuclear Energy*, 38(9):1987–1995, 2011.

[28] Aarno E Isotalo, Gregory G Davidson, Tara M Pandya, William A Wieselquist, and Seth R Johnson. Flux renormalization in constant power burnup calculations. *Annals of Nuclear Energy*, 96:148–157, 2016.

[29] Aarno E Isotalo, Jaakko Leppänen, and Jan Dufek. Preventing xenon oscillations in Monte Carlo burnup calculations by enforcing equilibrium xenon distribution. *Annals of Nuclear Energy*, 60:78–85, 2013.

[30] Aarno E Isotalo and Ville Sahlberg. Comparison of neutronics-depletion coupling schemes for burnup calculations. *Nuclear Science and Engineering*, 179(4):434–459, 2015.

[31] Colin Josey, Benoit Forget, and Kord Smith. High order methods for the integration of the Bateman equations and other problems of the form of $y' = f(y,t)y$. *Journal of Computational Physics*, Submitted.

[32] Colin Josey, Kord Smith, and Benoit Forget. A stochastic benchmark problem for depletion algorithms. In *Proceedings of the American Nuclear Society (ANS Winter Meeting 2017)*, Submitted.

[33] Dan Kotlyar and Eugene Shwageraus. Stochastic semi-implicit substep method for coupled depletion Monte-Carlo codes. *Annals of Nuclear Energy*, 92:52–60, 2016.

[34] Jaakko Leppänen. Serpent–a continuous-energy Monte Carlo reactor physics burnup calculation code. *VTT Technical Research Centre of Finland*, 4, 2013.

[35] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49, 2003.

[36] Hans Munthe-Kaas. Runge-Kutta methods on Lie groups. *BIT Numerical Mathematics*, 38(1):92–111, 1998.

[37] Olavi Nevanlinna and Aarne H Sipilä. A nonexistence theorem for explicit A-stable methods. *Mathematics of computation*, 28(128):1053–1056, 1974.

[38] Brynjulf Owren. Order conditions for commutator-free Lie group methods. *Journal of Physics A: Mathematical and General*, 39(19):5585, 2006.

[39] Maria Pusa. Higher-order Chebyshev rational approximation method and application to burnup equations. *Nuclear Science and Engineering*, 182(3), 2016.

[40] Miriam A. Rathbun and David P. Griesheimer. Effect of delayed energy release on power normalization in reactor depletion calculations. In *Proceedings of the American Nuclear Society (ANS Annual Meeting 2017)*, 2017.

[41] Joel Rhodes, Kord Smith, and Zhiwen Xu. CASMO-5 energy release per fission model. In *Proceedings of the International Conference on the Physics of Reactors (PHYSOR-2008)*, 2008.

[42] Paul K Romano and Benoit Forget. The OpenMC Monte Carlo particle transport code. *Annals of Nuclear Energy*, 51:274–281, 2013.

[43] Yousef Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 29(1):209–228, 1992.

[44] Lawrence F Shampine. Error estimation and control for ODEs. *Journal of Scientific Computing*, 25(1):3–16, 2005.

[45] Andrzej Suslowicz. Application of numerical Lie group integrators to parabolic PDEs. 2001.

[46] Mechthild Thalhammer. A fourth-order commutator-free exponential integrator for nonautonomous differential equations. *SIAM journal on numerical analysis*, 44(2):851–864, 2006.

[47] Neil E Todreas and Mujid S Kazimi. *Nuclear systems: thermal hydraulic fundamentals*, volume 1. CRC press, 2012.

[48] Paul J. Turinsky. *Handbook of Nuclear Engineering*, chapter Core Isotopic Depletion and Fuel Management, pages 1241–1312. Springer US, Boston, MA, 2010. doi:10.1007/978-0-387-98149-9_10.

[49] Wolfram Research, Inc. Mathematica 10.4. URL: https://www.wolfram.com.

[50] F Yang, G Wang, G Yu, K Wang, and Z Chen. Implementation of inline equilibrium xenon method in RMC code. In *Proceedings of the PHYSOR Conference*, pages 1–5. American Nuclear Society Sun Valley, ID, 2016.