Exponential Time Differencing Schemes for Fuel Depletion and Transport in Molten Salt Reactors: Theory and Implementation *

Zack Taylor,*,[a] Benjamin S. Collins,[a] and G. Ivan Maldonado[b]

[a] *Oak Ridge National Laboratories, Nuclear Energy and Fuel Cycle Division*
1 Bethel Valley Road, P.O. Box 2008, MS-6172, Oak Ridge, TN 37831-6172

[b] *University of Tennessee, Department of Nuclear Engineering*
1412 Circle Dr, Knoxville, Tennessee

*Email: taylorrz@ornl.gov

Number of pages: 53
Number of tables: 12
Number of figures: 19

Abstract

A numerical framework for modeling depletion and mass transport in liquid fueled molten salt reactions is presented based on exponential time differencing. The solution method involves using the finite volume method to transform the system of partial differential equations into a much larger system of ordinary differential equations. The key part of this method involved solving for the exponential of a matrix. We explore six different algorithms to compute the exponential in a series of progression problems which explore physical transport phenomena in molten salt reactors. This framework shows good results for solving linear parabolic partial differential equations with each of the six matrix exponential algorithms. For large problems, the series solvers such as Padé and Taylor have large run times, which can be mitigated by using the Krylov subspace.

Keywords — Molten Salt Reactors, Depletion, Burnup, Species Transport, Exponential Time Differencing

I. INTRODUCTION

Liquid fuel molten-salt reactors (MSRs) are a class of next-generation advanced nuclear reactors which not only show great promise, but also have a significant operational history. By design, MSRs operate in a much different way than traditional nuclear reactors. While conventional nuclear reactors employ fixed-geometry fuel elements, MSRs dissolve fuel in a molten salt that continuously flows throughout the reactor's primary loop. This allows for various chemical and isotopic species to transport and react in the loop. Moreover, during operation of any nuclear reactor, the material composition of the fuel is constantly changing. These changes come from transmutation due to neutron flux irradiation, nuclear decay, chemical reactions, and mass transport. Thus, understanding how the fuel composition changes is key to understanding many other physical processes which occur in nuclear reactors. Modeling these composition changes also provides insights into fuel safety and performance.

Modeling the composition changes in nuclear reactor fuel is referred to as *nuclear fuel depletion analysis*. The calculations presented herein model the atomic density of various isotopes in a nuclear reactor over long and short periods of time, referred to as *transients*. Modeling shorter transients is required in postulated accident scenarios and is necessary when tracking transient fission products like xenon and samarium. Long depletion steps are important for understanding fuel burnup and cycle length and for optimizing fuel performance. In traditional nuclear reactors, these calculations involve solving a large system of stiff first-order ordinary differential equations. The problem of accurately solving these equations has been resolved [1, 2, 3, 4, 5] by using modern matrix exponential methods. The existing depletion codes were developed to model the current fleet of reactors with fixed-geometry fuel. Consequently, advances in modern depletion codes have led to the ability to add external feed and removal of fuel materials, allowing for lumped depletion MSR analysis [2, 6, 7]. However, these codes do not characterize the underlying physical phenomena in advanced reactors with flowing fuel. There have been efforts to solve the problem of precursor drift in MSRs, but these analyses are limited to either steady-state analyses, or they employ numerical methods not well suited for full depletion calculations [8, 9].

The objective of this work is to redefine fuel depletion calculations in a nuclear reactor with flowing fuel, not only to set up the mathematical expressions to model the physical phenomena, but also to develop robust, accurate solutions to these equations. As part of this effort, the MSR

depletion code *libowski* has been developed based on a finite volume discretization of the transient species transport equation using an exponential time differencing scheme. In utilizing the species transport equation to track changes in isotopic composition, the nuclear/chemical reactions and mass transport that occur in MSRs can be accurately modeled. Many of the decay- and neutron-induced reactions occurring in nuclear reactors create a very stiff system of equations that must be solved. Traditional numerical integration techniques will not be able to model the lifetime of a reactor. Therefore, a method based on exponential time differencing is employed, requiring the computation of a matrix exponential [10, 11, 12]

II. BURNUP EQUATIONS

Traditional burnup calculations involve solving a system of first-order linear ordinary differential equations (ODEs) as shown in Eq. (1):

$$\frac{dn_i}{dt} = \sum_{j=1}^{N} \left(\lambda_{j\to i} + \phi \sum_{k=1}^{K} \gamma_{j\to i,k} \sigma_{k,j} \right) n_j(t) - \left(\lambda_i + \phi \sum_{k=1}^{K} \sigma_{k,i} \right) n_i(t), \tag{1}$$

where N is the number of nuclides in the system, and K is the number of neutron-induced reactions for a specific isotope. The first term on the right-hand side of Eq. (1) represents generation from decay of nuclide j and neutron-induced reactions. The second term includes losses from decay and transmutation reactions. Microscopic reaction rates and the neutron flux are collapsed into single group constant values over a time step. Spatially, the reactor core can be divided into depletion zones, or the reactor can be homogenized into a single point for lumped calculations.

Equation (1) is more commonly represented in matrix vector form:

$$\frac{d\boldsymbol{n}}{dt} = \boldsymbol{A}\boldsymbol{n}, \quad \boldsymbol{n}(t_0) = \boldsymbol{n}_0, \tag{2}$$

where $\boldsymbol{n}(t)$ is the nuclide concentration vector, $\boldsymbol{A}$ is the transition matrix, and $\boldsymbol{n}_0$ is the initial condition vector. Eq. (2) has the solution $\boldsymbol{n}(t) = e^{\boldsymbol{A}t}\boldsymbol{n}_0$, where $e^{\boldsymbol{A}t}$ is defined as follows [1, 13, 14]:

$$e^{\boldsymbol{A}} = \sum_{k=0}^{\infty} \frac{1}{k!} \boldsymbol{A}^k \tag{3}$$

The transition matrix contains the decay and transmutation coefficients:

$$a_{i,i} = -\left(\lambda_i + \phi \sum_{k=1}^{K} \sigma_{k,i}\right), \tag{4}$$

$$a_{i,j\neq i} = b_{j\to i}\lambda_j + \sum_{k=1}^{K} \gamma_{j\to i,k}\sigma_{k,j}\phi. \tag{5}$$

There are two major matrix properties which influence the accuracy of many popular numerical matrix exponential algorithms used to compute Eq. (3): matrix norm and the location of the eigenvalues on the imaginary plane. Series approximations such as Padé and Taylor are most accurate around the origin, meaning that the norm of the matrix must be small, they have no requirements for the eigenvalues [14]. How small the norm must be depends on the order of the approximation. When using the ℓ_1 norm, $\boldsymbol{A}t$ is known to be

$$||\boldsymbol{A}t||_1 = |t|\ ||\boldsymbol{A}||_1 \geq |t|\ \max|a_{i,j}|, \tag{6}$$

meaning that the norm must be greater than or equal to the absolute value of the maximum matrix element multiplied by the absolute value of time. Because burnup calculations are often taken over long time steps and include isotopes which can greatly increase the norm of the matrix, series approximations incur complications. Therefore, to use these approximations, the matrix exponential algorithm must be combined with scaling and squaring to reduce the norm to a suitable value.

Solutions based on the Cauchy integral formula do not have a requirement on the norm of the matrix, but they do have a requirement on the eigenvalues [3]. In particular, the eigenvalues of the transition matrix must fall in a region enclosed by the contour function. It is noted by Pusa that the eigenvalues of the transition matrix are clustered around the negative real axis [1]. This makes the CRAM algorithm well suited to solve the system given in Eq. (2). A number of papers discuss the accuracy of CRAM vs many commonly used matrix exponential methods for depletion calculations [1, 15]. Isotalo and Pusa demostrate that CRAM outperformed the methods tested.

Many matrix exponential algorithms rely on solving systems of linear equations, and the choice of linear solvers affects the algorithms' accuracy. The half lives and microscopic cross sections for nuclides can vary significantly, causing the magnitude coefficients in the transition matrix to vary from zero to 10^{21} [16]. For example, radioactive decay results in half lives that

range from 10^{-24} seconds to billions of years [17]. Many iterative solvers have difficulty dealing with the rounding errors introduced by the coefficients, and the resulting system will also have extremely small and large eigenvalues. Iterative solves that rely on Krylov subspace methods become disadvantageous for solving such systems because of the spectral properties of the matrix [16]. To achieve a high order of accuracy and stability, direct solvers are chosen over iterative ones. Such solvers include SuperLU or sparse Gaussian elimination with partial pivoting.

II.A. Burnup Equations in Molten Salt Reactors

Depletion calculations in MSRs can be better represented by the multicomponent chemical species transport equation. The species transport equation, shown in Eq. (7), is derived by a conservation of mass basis which accounts for change in time, convective and diffusion transport, and rate of generation from reaction [18].

$$\underbrace{\frac{\partial \rho_i}{\partial t}}_{\substack{\text{Change in}\\ \text{density}\\ \text{with time}}} + \underbrace{\nabla \cdot \rho_i(r,t)\boldsymbol{v}}_{\substack{\text{Transport}\\ \text{with fluid}\\ \text{velocity}}} + \underbrace{\nabla \cdot j_i(r,t)}_{\substack{\text{Transport}\\ \text{with mass}\\ \text{diffusion}}} = \underbrace{R_i(r,t)}_{\substack{\text{Generation}\\ \text{from}\\ \text{reaction}}} \tag{7}$$

Using Eq. (7) as the basis, depletion equations in MSRs can be derived by using Eq. (1) to represent the source terms from nuclear reactions in burnup calculations. Reaction rates for nuclear reactions must be converted from atomic density to mass density using Avogadro's constant and molar mass, $n_i = \rho_i N_A / M_i$. The MSR depletion equation represents the change in an isotope's density via fluid transport, mass diffusion, and nuclear reactions:

$$\begin{aligned} \frac{\partial \rho_i}{\partial t} + \nabla \cdot \rho_i \boldsymbol{v} + \nabla \cdot j_i = &\sum_{j=1}^{N} \frac{M_i}{M_j} \left(b_{j \to i} \lambda_j + \sum_{k=1}^{K} \gamma_{j \to i,k} \sigma_{k,j} \phi \right) \rho_j \\ &- \left(\lambda_i + \phi \sum_{k=1}^{K} \sigma_{k,i}(r) \right) \rho_i. \end{aligned} \tag{8}$$

It is important to note that $\boldsymbol{v}$ is the mass averaged velocity, and j_i is the diffusive flux which can account for both molecular and turbulent diffusion.

II.B. Spatial Discretization

The spatial dependence of Eq. (8) is approximated using a cell-centered 2D finite volume scheme. Each dimension is split, meaning that the 2D implementation consists of one dimension scheme in each direction [19]. A schematic for a single cell P is shown in Figure 1.

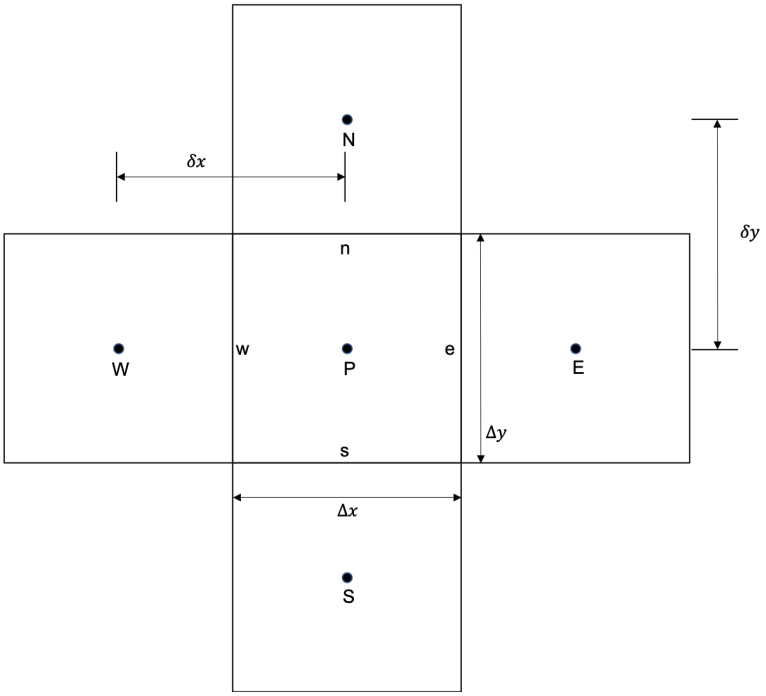

Fig. 1. 2D finite volume representation

II.B.1. Transport Terms

The diffusive flux is represented with Fick's law of diffusion for an ideal mixture:

$$j_{i,x} = -D_i \frac{d\rho_i}{dx}. \tag{9}$$

Each volume element is integrated in the x and y direction to produce a second-order spatial discretization of the diffusive flux. For the x direction, the diffusion flux is integrated over the cell's surface:

$$\frac{1}{V}\int_V \frac{\partial}{\partial x}\left(D_i \frac{\partial \rho_i}{\partial x}\right) dxdy = \frac{D_{e,i}}{\Delta x \Delta y}\left(\frac{\partial \rho_i}{\partial x}\right)_e \Delta y - \frac{D_{w,i}}{\Delta x \Delta y}\left(\frac{\partial \rho_i}{\partial x}\right)_w \Delta y. \tag{10}$$

The diffusion coefficients are averaged between each pair of points. After the derivative approximation is plugged into each surface, the method is the same, using a second-order central difference with cell centers as finite difference points.

The convective transport term is more difficult to deal with than diffusion. Diffusion has no primary direction of flow, it simply causes a species to evenly distribute through a medium through a concentration gradient. Convection, on the other hand, has a primary flow direction that is driven by a pressure gradient. One of the major drawbacks of using a second-order central differencing scheme is the inability to identify flow direction. In addition to the neglect in identifying the flow direction, the central differencing scheme will cause numerical instability problems for flows with high Péclet numbers [20]. To combat these numerical problems and to handle flow direction, a second-order upwind differencing scheme is used.

There are a number of classical upwind differencing schemes—such as first order, power law, and QUICK—each of which has a different order of accuracy and stability region. Second-order or higher convection schemes can lead to undershooting or overshooting, and applying boundary conditions can be problematic [20]. Because of the potential problems with higher order schemes, efforts have been made to derive a class of second-order total variation diminishing (TVD) schemes that avoid stability and oscillation issues. TVD schemes have the property of preserving monotonicity, meaning that they must not create local extrema, the value of an existing local minimum must be non-decreasing, and the value of a local maximum must be non-increasing [21]. One other consequence of a monotonicity preserving scheme is that the total variation of the solution should diminish or remain the same with time.

For flow in the x direction, the convection operator is integrated over the control volume to approximate the convective flux into the cell:

$$\frac{-1}{V}\int_V \frac{\partial}{\partial x}(\rho_i v)dxdy \approx \frac{-1}{\Delta x \Delta y}\left[v_e \rho_{e,i} - v_w \rho_{w,i}\right]\Delta y, \tag{11}$$

where v_e and v_w are the velocity components in the x direction normal to the cell's surface. The second-order convection flux approximation is represented in a general form for the east and west faces:

$$\begin{aligned} \rho_e &= \rho_P + \frac{1}{2}\psi(r_e)(\rho_E - \rho_P), \text{ and} \\ \rho_w &= \rho_W + \frac{1}{2}\psi(r_w)(\rho_P - \rho_W), \end{aligned} \tag{12}$$

where ψ is the flux limiter function, and r is the ratio of the upwind side gradient and the downwind side gradient [20]. The capital letters P, E and W represent the species concentrations in cells W, E and P. The ratio is defined as the upwind difference over the downwind difference. For positive flow, the gradient ratio for the east and west cell faces is as follows:

$$r_e = \left(\frac{\rho_P - \rho_W}{\rho_E - \rho_P}\right), \quad r_w = \left(\frac{\rho_W - \rho_{WW}}{\rho_P - \rho_W}\right), \tag{13}$$

where, WW is the concentration in the cell west of cell W. The generic form of Eq. (12) allows for the use of first-, second- or higher order convection schemes, depending on the flux limiter function. Many linear limiter functions exist which are more stable than central differencing and more accurate than first-order upwind, but these schemes are still vulnerable to unphysical oscillations which occur because no linear convection scheme greater than first order can be monotonic [19]. Nonlinear schemes solve this problem and have been heavily used in the computational fluid dynamics community. There are many flux limiter functions, but for the context of this work, only the Superbee and MUSCL functions are presented here in Eqs. (14) and (15):

$$\psi(r) = \max[0, \min(2r, 1), \min(r, 2)] \qquad \text{Superbee} \tag{14}$$

$$\psi(r) = \max\left[0, \min\left(2r, \frac{r+1}{2}, 2\right)\right] \qquad \text{MUSCL} \tag{15}$$

More often, these higher order convection flux methods are implemented in a manner known as *deferred corrections* to ensure that the coefficients in the matrix do not introduce negative concentrations [22]. The deferred correction method applies to the first-order upwind convection flux implicitly and the second-order portion explicitly. The second-order correction is calculated using species concentrations from the previous time step and is applied as a constant source term [20]. Because the second-order term is applied explicitly, small time steps can increase the

accuracy of the convective flux approximation, and the transition matrix must be updated after each iteration. For smooth functions, this formulation is second order, but it reverts back to first order for discontinuous functions.

II.B.2. Boundary Conditions

Dirichlet and Neumann boundary conditions are implemented using a mixture of numerical and direct methods. The first-order portion of the convective flux is directly calculated for a Dirichlet boundary, and the second order correction term requires information for a cell outside the boundary. A ghost cell is placed outside the domain, and its concentration is calculated by linear extrapolation. For cell P on the west boundary, the corresponding ghost cell will have the following concentration:

$$\rho_W = 2\rho_b - \rho_P, \tag{16}$$

where ρ_b is the Dirichlet boundary value. Neumann boundary conditions are applied in a similar manner by approximating the derivative at the boundary using a second-order central difference approximation at the boundary. For a the east boundary, this leads to the following:

$$\rho_W = \rho_P - \rho_b' \delta x, \tag{17}$$

where ρ_b' is the value of the Neumann boundary condition.

II.B.3. Volumetric Source Terms and Initial Conditions

Volumetric source terms and initial conditions must be applied in a way that preserves the spatial discretization technique. This means that initial conditions and volumetric sources must be integrated over the cell using the mean value theorem. Nuclear reaction rates are calculated in a way that preserves reaction rates. This is done by integrating over all neutron energies and the volume of the cell. Over a single burn-up calculation step, the scalar neutron flux and all cross sections are assumed to be constant. Using volume average operators and the multigroup approximation, the neutron flux and microscopic cross section are collapsed into single values for each depletion zone [23]:

$$\sigma_{k,j} = \frac{\int_V \int_0^\infty \sigma_{k,j}(r,E)\phi(r,E)dEdV}{\int_V \int_0^\infty \phi(r,E)dEdV}, and \tag{18}$$

$$\phi = \frac{1}{V}\int_V \int_0^\infty \phi(r,E)dEdV. \tag{19}$$

Source terms for nuclear reactions are not calculated inside libowski. Thus, for nuclear reactions, an external source must provide the cross sections, neutron flux, and decay constants, and they must already be calculated in the correct manner.

III. EXPONENTIAL TIME DIFFERENCING METHODS

Using the approximation methods in Section II, Eq. (8) is transformed from a coupled system of partial differential equations to an even larger system of coupled ordinary differential equations. In matrix vector form, this set of equations takes the following form:

$$\frac{d\boldsymbol{\rho}}{dt} = \boldsymbol{L}\boldsymbol{\rho} + \boldsymbol{N}(t,\boldsymbol{\rho}), \tag{20}$$

where the operators $\boldsymbol{L}$ and $\boldsymbol{N}$ denote the linear and nonlinear parts, respectively. The linear operator is a matrix, commonly referred to as the *transition matrix*, that contains all of the nuclear source terms, as well as the linearized convection and diffusion coefficients. Although the high-order flux limiter functions are nonlinear, they are implemented as constant source terms. These convection flux correction terms are calculated explicitly from the previous time step and added to the transition matrix. The nonlinear operator contains any nonlinear source terms that might arise when modeling transport in MSRs, although none were present in Eq. (8). These source terms may include chemical reactions and phase transition models, for example. Accounting for nonlinear source terms here allows for them to be added in future work.

Exponential time differencing (ETD) is a class of methods used for solving stiff systems of ordinary differential equations, in which the stiffness arises from the linear operator. These methods have been extensively used to solve nonlinear partial differential equations in a wide range of scientific fields [10, 11, 12, 24, 25, 26, 27]. Using ETD methods, the solution to Eq. (20) is known to be [10]

$$\boldsymbol{\rho}(t_n + \Delta t) = e^{\Delta t \boldsymbol{L}} \boldsymbol{\rho}(t_n) + e^{\Delta t \boldsymbol{L}} \int_0^{\Delta t} e^{-\boldsymbol{L}\tau} \boldsymbol{N}(t_n + \tau, \boldsymbol{\rho}(t_n + \tau)) d\tau. \tag{21}$$

This formalization is exact, and exponential time differencing methods work to approximate the integral of the nonlinear portion. Exponential time differencing methods have the property of being exact when $\boldsymbol{N}(\boldsymbol{\rho}, t) =$ constant [12]. The integral in Eq. (21) can be evaluated using traditional multistep methods or Runge-Kutta methods [10]. While constant source terms can be implemented in this framework, it should be noted that other methods have been developed to handle constant and nonlinear time-dependent source terms in reactor depletion calculations. For example, constant source terms in libowski are implemented using the dummy species method described by Isotalo et al [2]. This creates the same system shown above in Eq. (2),

$$\boldsymbol{\rho}(t_n + \Delta t) = e^{\Delta t \boldsymbol{L}} \boldsymbol{\rho}(t_n), \tag{22}$$

where matrix $\boldsymbol{L}$ constants all have the same depletion coefficients as transition matrix $\boldsymbol{A}$, but with the addition of coefficients from diffusion and convection. The size of matrix $\boldsymbol{L}$ is also much larger than $\boldsymbol{A}$, being on the order of the number of species times the number of finite volume cells.

IV. COMPUTING THE EXPONENTIAL OF A MATRIX

When obtaining solutions based on exponential time differencing, an exponential of a matrix must be computed. There are multiple computational methods for solving for the matrix exponential, many of which were developed specifically to evaluate $e^{\boldsymbol{A}t}$ or $e^{\boldsymbol{A}t}\boldsymbol{v}$: the former evaluates the matrix exponential directly, and the latter calculates the action of the exponential on a vector. Because of the way the system is presented in Eq. (22), only the action of the matrix on the vector is required. However, in some of the methods presented below, only direct evaluation of the matrix exponential is possible.

Computation of the exponential of a matrix is by far the most difficult part of the ETD methods. The basis for many of the methods is mathematically in depth, and the methods are also difficult to deploy in a manner that ensures fast, accurate computation of the matrix exponential. Numerous methods for computing the matrix exponential are discussed here, but they are not limited to those presented here.

IV.A. Series Approximations

Solvers of this nature often exploit the following relation when computing the matrix exponential:

$$e^{\boldsymbol{A}t} = \left(e^{\boldsymbol{A}t/m}\right)^m, \tag{23}$$

where m is a scalar that *scales* matrix $\boldsymbol{A}t$. This method is known as scaling and squaring, and its purpose is to reduce the norm of matrix $\boldsymbol{A}t$, especially in situations for $\boldsymbol{A}t$, when $t \to \infty$, as mentioned above. For series methods near the origin, the accuracy of the method diminishes as the matrix norm increases.

IV.A.1. Padé Approximation

The Padé approximation represents a function by expanding it as a ratio of two power series. A (p, q) Padé approximation for $e^{\boldsymbol{A}t}$ is defined by [14]

$$e^{\boldsymbol{A}t} \approx R_{p,q}(\boldsymbol{A}t) = \frac{N_{p,q}(\boldsymbol{A}t)}{D_{p,q}(\boldsymbol{A}t)}, \tag{24}$$

where

$$N_{p,q}(\boldsymbol{A}t) = \sum_{j=0}^{p} \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!}(\boldsymbol{A}t)^j, \tag{25}$$

$$D_{p,q}(\boldsymbol{A}t) = \sum_{j=0}^{q} \frac{(p+q-j)!q!}{(p+q)!j!(q-j)!}(-\boldsymbol{A}t)^j. \tag{26}$$

Padé methods are similar to those in the Taylor series, as they approximate a function using a series solution however, Padé series usually outperform Taylor series when the function contains poles. Series solutions methods such as Padé are also more accurate near the origin, so the matrix norm $||\boldsymbol{A}t||$ must be sufficiently small for the approximation to be accurate [1]. Yet another problem arises when $\boldsymbol{A}$ has a wide spread of eigenvalues, causing an ill-conditioned linear system [13, 14].

There are many ways to develop an algorithm for computing the matrix exponential using the Padé approximation [13, 28, 29]. The key in deriving an algorithm is to understand the error associated with the size of the matrix norm and to limit the computation time. Moler and Van

Loan derived an elegant, simple proof for determining values for p, q, and m given a matrix norm [14]. As noted by Higham [28], this derivation contains weaknesses. Moler and Van Loan assumed that the matrix norm needed to be less than one half ($||\boldsymbol{A}t|| < 1/2$), but Higham proved that this is not the case. Higham further showed that the required minimal matrix norm is different for each order of the Padé implementation. Above this norm, a higher order Padé approximation would be required, or matrix scaling would need to occur. One other weakness was the derivation of the error bound. It was designed to be easily commutable, which resulted in the error bound not being sharp [28]. When the error bound is not sharp, then it is possible to overscale the matrix, resulting in a loss of accuracy. Higham and Al-Mohy describe two algorithms to resolve the overscaling problem found in other work by Higham. Higham's work in 2009 [29] is an update to the algorithm described in the work from 2005 [28], which fixes the overscaling problem.

In combination with scaling and squaring, the Padé approximation is probably the most widely used method for computing the exponential of a matrix. In fact, the *expm* function in MATLAB is based on this approach [29]. Two methods based on the Padé approach are implemented in libowski—*Padé - Method 1* and *Padé - Method 2*. Both algorithms work to form a Padé approximation of type $p = q$ and are further denoted by m, combined with scaling and squaring,

$$e^{\boldsymbol{A}} = (e^{2^{-s}\boldsymbol{A}})^{2^s} \approx r_m(2^{-s}\boldsymbol{A})^{2^s}, \tag{27}$$

where s is the scaling and squaring parameter, $m = 2^s$, and $\boldsymbol{A}$ is $\boldsymbol{A}t$ and r_m is the Pad/'e approximation of order m. The scaling parameter s is chosen so that the exponential is computed with a backward error bounded by the unit roundoff. Method 1 is based on the algorithm developed by Higham [28], in which the backwards error is based on $||\boldsymbol{A}||$. Method 2 is based on Higham's later work [29], which reduces the Method 1 problem of overscaling by tightening the backwards error based on $||\boldsymbol{A}^k||^{1/k}$:

$$||\boldsymbol{A}^k||^{1/k} \leq ||\boldsymbol{A}||, \quad k = 1 : \infty. \tag{28}$$

IV.A.2. Taylor Series

Formally, matrix exponential is defined using an infinite Taylor series:

$$e^{\boldsymbol{A}t} = \sum_{k=0}^{\infty} \frac{1}{k!}(\boldsymbol{A}t)^k. \tag{29}$$

Therefore, a straightforward way to calculate the matrix exponential is to use its formal definition. However, this method is not commonly used in application for either the matrix or the scalar case. The number of terms required to achieve convergence can be large and can produce computational inefficiency. This method also suffers from numerical roundoff errors from cancellation for large values of k [14].

While not commonly used in practice, Al-Mohy and Higham developed an algorithm for computing the action of the matrix exponential of matrix $A \in \mathbb{C}^{n\times n}$ on matrix $B \in \mathbb{C}^{n\times n_0}$, where $n_0 << n$ based on a truncated Taylor series [30]:

$$e^{\boldsymbol{A}}\boldsymbol{B} = (e^{2^{-s}\boldsymbol{A}})^{2^s}\boldsymbol{B} \approx T_m(2^{-s}\boldsymbol{A})^{2^s}\boldsymbol{B}, \tag{30}$$

where T_m is a truncated Taylor series of order m. Unlike the Padé methods presented above, the Taylor series does not require linear solves.

The Taylor method presented here was developed in a manner similar to that used for the Padé methods, as it is derived using a backward error analysis based on $||\boldsymbol{A}^k||^{1/k}$, keeping in mind the computational cost. Unlike Padé Method 2, the Taylor method uses two key preprocessing steps to reduce the norm of $\boldsymbol{A}$. These steps include shifting and optional balancing. Shifting is implemented in libowski however, optional balancing is not. This algorithm is implemented in libwski with the suggested parameters found in reference ([30]).

IV.B. Rational Approximations

While the Padé approximation is indeed a rational function approximation, it differs from the methods presented in this section. The rational functions presented here are represented in partial fraction decomposition form. These types of methods are algorithms that were developed by transforming the matrix exponential into the complex plane using Cauchy's integral formula:

$$e^{\boldsymbol{A}t} = \frac{1}{2\pi i}\int_{\Gamma} e^{z}(z\boldsymbol{I} - \boldsymbol{A}t)^{-1}dz, \tag{31}$$

where $\boldsymbol{A}t$ is analytic inside the closed contour Γ that winds once around the eigenvalues of $\boldsymbol{A}t$ [3] [31]. In practice, the action of the matrix exponential on vector $\boldsymbol{v}$ must be evaluated, which leads to evaluating the following:

$$e^{\boldsymbol{A}t}\boldsymbol{v} = \frac{1}{2\pi i}\int_{\Gamma} e^{z}(z\boldsymbol{I} - \boldsymbol{A}t)^{-1}\boldsymbol{v}dz. \tag{32}$$

Three solvers based on rational approximations were implemented in libowski: *hyperbolic*, *parabolic* and *CRAM*. The following sections briefly discuss the theories behind these solvers.

IV.B.1. Quadrature Contours

This method involves evaluating the contour integral by choosing an analytic function $\phi(\theta)$ that maps the real line onto the contour. Because the function $e^{\phi(\theta)}$ decreases exponentially as $|\theta| \to \infty$, the approximation can be truncated to a finite number of quadrature points. When the spectrum of the transition matrix falls on the left-hand side of the complex plane close to the real axis, the contour Γ denotes a Hankel-like contour that winds from $-\infty - 0i$ on the lower half-plane and $-\infty + 0i$ on the upper half-plane [31]. This allows for definition of a general contour function that will enclose the eigenvalues on the left-hand side of the complex plane around the negative real axis.

Trefethen et al. note three contour functions to Γ, two of which are presented here [31]. The simplest contour function is a parabola defined by

$$\phi = N[0.1309 - 0.1194\theta^2 + 0.2500i\theta], \tag{33}$$

which has a convergence rate of $\mathcal{O}(2.85^{-N})$. Accuracy of about 14 or more digits can be achieved with N = 32. The approximation of e^z on the complex plane is shown in Figure 2. The second contour function is that of a hyperbola, which is defined by

$$\phi = 2.246N[1 - \sin(1.1721 - 0.3443i\theta)], \tag{34}$$

which has a convergence rate of $\mathcal{O}(3.20^{-N})$, with an accuracy of about 16 or more digits with N = 32. The same approximation for e^z on the complex plane is shown in Figure 3. Figures 2 and 3 show high levels of accuracy not only on the negative real axis, but also for a wide region of the left-hand side of the complex plane.

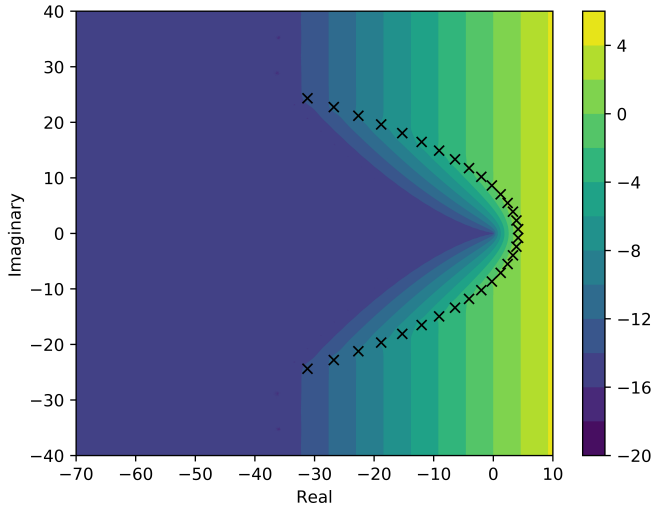

Fig. 2. $\log_{10}|r(z)-e^z|$ for N = 32, where the contour is defined by a parabola, Eq. (33); quadrature points are denoted with x marks

Applying these approximations to real valued scalars or matrices requires half the amount of computational cost, because the poles of a rational function with real valued coefficients form conjugate pairs [3]. For matrix $\boldsymbol{A}t$, the solution becomes

$$e^{\boldsymbol{A}t}\boldsymbol{v} \approx r(\boldsymbol{A}t)\boldsymbol{v} = 2Re\left(\sum_{k=1}^{N/2} c_k(\boldsymbol{A}t - z_k\boldsymbol{I})^{-1}\boldsymbol{v}\right), \tag{35}$$

requiring $N/2$ solves of the linear system $\boldsymbol{x} = (\boldsymbol{A}t - z_k\boldsymbol{I})^{-1}c_k\boldsymbol{v}$. It is important to note that these linear systems are independent of one another and can be solved in parallel.

IV.B.2. Best Rational Approximation

A different approach is to choose a function $r(z)$ that is the best approximation of the exponential function on the negative real axis, thus bypassing the need for a contour function [17] [31]. This method is known as the *Chebychev Rational Approximation Method* (CRAM) and

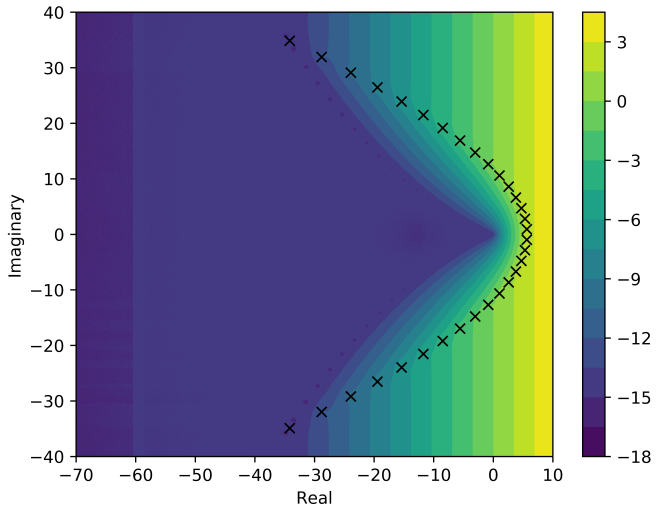

Fig. 3. $\log_{10}|r(z) - e^z|$ for N = 32, where the contour is defined by a hyperbola, Eq. (34); quadrature points are denoted with x marks

has a similar form as the previous rational approximation . The convergence rate for CRAM is of the order $\mathcal{O}(9.28903^{-N})$, which is remarkably faster than those previously shown. With $N = 16$ quadrature points, CRAM gives about 15 or more digits of accuracy. Thus, the same order of accuracy can be achieved with half the number of quadrature points than in the rational approximations defined by contour functions.

The difficulty with using the CRAM approximation is in finding the coefficient for the rational approximation. For CRAM of orders 14 and 16, the rational coefficient can be found in Pusa ([3]) for up to 20 digits. Figure 4 shows the accuracy of CRAM to the function e^z on the complex plane. Because the rational function was built to ensure the best approximation on the negative real axis, the accuracy of CRAM is in a more narrow range of the real axis. For a real values matrix, the CRAM algorithm leads to the following solution:

$$e^{\boldsymbol{A}t}\boldsymbol{v} \approx r(\boldsymbol{A}t)\boldsymbol{v} = c_0\boldsymbol{v} + 2Re\left(\sum_{k=1}^{N/2} c_k(\boldsymbol{A}t - z_k\boldsymbol{I})^{-1}\boldsymbol{v}\right). \tag{36}$$

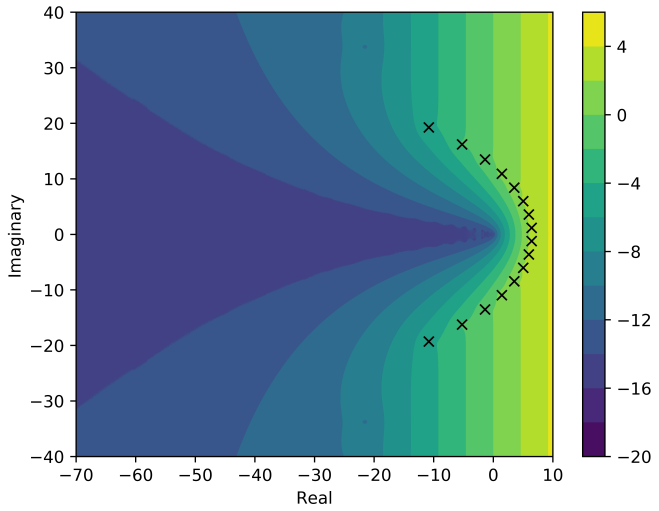

Fig. 4. $\log_{10}|r(z) - e^z|$ for CRAM with N = 16; quadrature points are denoted with x marks

IV.B.3. Accuracy of Rational Approximations

While the matrix exponential can be formally defined by Eq. (29), when examining the accuracy of rational approximations presented, it is more useful to define the matrix exponential in terms of the transition matrix's eigenvalues. Because the contour Γ must wind around the eigenvalues of $\boldsymbol{A}t$, if $\boldsymbol{A}$ has eigenvalues with non-trivial imaginary parts, then these eigenvalues will scale as a function of t. If these eigenvalues scale to a portion of the complex plane which fall outside the contour, then the accuracy of these methods can be compromised. As stated above, for traditional burnup matrices, the eigenvalues are clustered around the negative real axis [1]. Therefore, the accuracy of CRAM and the other two rational approximations are not a function of time t [32]. For burnup calculations in MSRs defined by Eq. (8), the eigenvalues are not necessarily clustered around the negative real axis, as they have non-trivial imaginary parts. Numerical experiments were used to find that these imaginary parts were introduced by the addition of convection terms in the transition matrix. In addition, these imaginary parts were found to be correlated with the ratio of the velocity to the spatial discretization, as well as the boundary conditions.

While rational approximations such as CRAM have shown exemplary results in burnup calculations, the relative accuracy of CRAM diminishes when the nuclide concentration diminishes

significantly over the time step. For a nuclide concentration $n_i(t) << n_i(0)$, the error estimate for CRAM follows [5]:

$$\frac{\delta \hat{n}_i(t)}{n_i(t)} = \hat{\varepsilon}_{k,k} \frac{n_i(0)}{n_i(t)}. \tag{37}$$

This means that for CRAM of order k, a concentration smaller than $\hat{\varepsilon}_{k,k} n_i(0)$ is not captured by the rational approximation. Additionally, the accuracy of CRAM is diminished when performing calculations on fresh fuel vs depleted fuel. For fresh fuel, the concentrations of nuclides depends on transitions corresponding to only a few nuclides in the initial condition. If CRAM introduces large errors in approximating the matrix elements of a few transitions which influence the production of a large number of nuclides, then the relative error in calculating nuclides concentrations increases. For used fuel depletion, these errors are averaged out to provide an overall more accurate calculation. While this error was derived for transition matrices built from solving tradition burnup calculations in the form of Eq. (2), it is assumed that the same holds true for MSR calculations in the form of Eq. (8). To increase the accuracy of CRAM methods for depletion calculations, a sub-stepping approach was introduced into ORIGEN [5]. This sub-stepping method is also used in this work in the implementation of the CRAM, hyperbolic and parabolic algorithms.

The absolute error in computing rational approximations in the form of Eq. (35) do not follow the same error as CRAM. In the scalar case of CRAM, if the absolute error is plotted on the negative real axis as $x \to -\infty$, then the absolute error asymptotically approaches $\hat{\varepsilon}_{k,k}$. This is because the exponential function tends to zero, while CRAM stabilizes at $\hat{\varepsilon}_{k,k}$. As $x \to 0$, the absolute error oscillates between $-\hat{\varepsilon}_{k,k}$ and $\hat{\varepsilon}_{k,k}$ [32]. While the errors for rational functions based on quadrature contours do not necessarily follow this same behavior, the sub-stepping method was implemented and is later shown to also increase their accuracy.

Sub-stepping is implemented by scaling the time step size and evaluating the solution from the previous step, as shown in Algorithm 1 below, where m is the number of substeps to be taken:

Algorithm 1 Sub-stepping

1: $\boldsymbol{v}_0 = \boldsymbol{\rho}_0$
2: $t = t/(m+1)$
3: **for** $j = 0, 1, \ldots m$ **do**
4: $\quad \boldsymbol{\rho}_{m+1} = r(\boldsymbol{A}t)\boldsymbol{v}_0$
5: $\quad \boldsymbol{v}_0 = \boldsymbol{\rho}_{m+1}$
6: **end for**

IV.C. Krylov Subspace Approximation

Krylov subspace approximations are a class of popular methods used in sparse matrix algorithms. The idea of Krylov subspace methods is to project the sparse $n \times n$ $\boldsymbol{A}$ matrix into a lower dimensional subspace. The new lower dimension projection is of size $m \times m$, where $m < n$. Because the matrix is of lower dimension, calculating its matrix exponential is much faster. It is important to note that Krylov subspace methods can only be used as an operation on a vector: direct calculation of the matrix exponential is not possible [33].

The objective is to approximate the matrix exponential as a polynomial of order $m-1$: this takes the following form:

$$e^{\boldsymbol{A}}\boldsymbol{v} \approx p_{m-1}(\boldsymbol{A})\boldsymbol{v}. \tag{38}$$

Because the polynomial interpolates the exponential function in the Hermite sense at the eigenvalues, the eigenvalues must be sufficiently close to one another for the approximation to be accurate [1]. This approximation is an element of the Krylov subspace, which is defined by

$$K_m = \text{span}\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}^2\boldsymbol{v}, ..., \boldsymbol{A}^{m-1}\boldsymbol{v}\}. \tag{39}$$

For a general nonsymmetric matrix, the Arnoldi algorithm can be used to build the Krylov space [33] [34]. Algorithm 2 constructs an orthonormal basis—$\boldsymbol{V}_m = [\boldsymbol{v}_1, \boldsymbol{v}_2, ...\boldsymbol{v}_m]$—of the Krylov subspace, as well as an $m \times m$ upper Hessenberg matrix:

Algorithm 2 Arnoldi

```
Compute v_1 = v/||v||_2
for j = 1, 2, ... m do
    Compute w = A v_j
    for i = 1, 2, ... j do
        Compute h_{i,j} = (w, v_i)
        Compute w = w − h_{i,j} v_i
    end for
    Compute h_{j+1,j} = ||w||_2 and v_{j+1} = w/h_{j+1,j}
end for
```

The Arnoldi algorithm produces the following relation:

$$\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_m\boldsymbol{H}_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T, \tag{40}$$

where $\boldsymbol{H}_m = \boldsymbol{V}_m^T \boldsymbol{A} \boldsymbol{V}_m$, and $\boldsymbol{e}_m$ is the unit vector of dimension m. The Hessenberg matrix $\boldsymbol{H}_m$ represents the projection of $\boldsymbol{A}$ onto the Krylov subspace. The approximation in Krylov space is known to be

$$e^{\boldsymbol{A}} \boldsymbol{v} \approx \beta \boldsymbol{V}_m e^{\boldsymbol{H}_m} \boldsymbol{e}_1, \tag{41}$$

where $\beta = ||\boldsymbol{v}||_2$ [34]. The computation of $e^{\boldsymbol{H}_m}$ becomes much easier because $\boldsymbol{H}_m$ is dense and smaller than $\boldsymbol{A}$. After the Krylov approximation is made, a typical method for solving the matrix exponential is used on $e^{\boldsymbol{H}_m}$. The quality of this approximation is exact when $n = m$. This is because at step m, $h_{m+1,m} = 0$, and Eq. (40) becomes:

$$\boldsymbol{A} \boldsymbol{V}_m = \boldsymbol{V}_m \boldsymbol{H}_m. \tag{42}$$

The Arnoldi process will be exact after m steps, when m is greater than or equal to the degree of the minimal polynomial in Eq. (38). At this point, Eq. (38) is exact, but this is unlikely to occur until $m = n$ [33] [34].

V. RESULTS

A number of mass transport tests were conducted to access each of the presented matrix exponential algorithms. The first problem set explored the spatial convergence of the convection and diffusion differential operators in typical reaction-convection-diffusion problems. Secondly, the six neutron precursors were modeled for short transients. This problem induced a transition matrix with large imaginary parts to introduce sub-stepping in reducing the numerical error. Lastly, a depletion problem was run with a small number of nuclides. Whereas most of the problems would have analytical solutions, a select few would not. For these select problems, the reference solutions are further discussed in the specific section.

All problems showed an error based on a reference solution. For the following results, the relative errors are defined as:

$$E_\infty = \max\left(\hat{\boldsymbol{u}} - \boldsymbol{u}\right) \qquad E_1 = \frac{1}{N} \sum_{i=1}^{N} \hat{u}_i - u_i,$$

where N is the number of elements in the solution domain. Sometimes it is more meaningful to

show an absolute error instead of a relative error. The results explicitly state whether a relative or absolute difference is used. Runtime is also reported for some tests and is reported as the wall time for calling the *solve* function. This includes the time to build the matrix, run the solution algorithm, and unpacking the solution. For problems with multiple time steps, the matrix was rebuilt before each time step to update the deferred correction source term. While these run times are reported with no standard deviation, some changes are to be expected when running problems multiple times or on different machines.

As discussed above, sub-stepping can increase the accuracy of Cauchy-based solvers. Unless otherwise noted, for all results shown, no substeps were used for the CRAM, parabolic, or hyperbolic solvers. For some of the reaction-diffusion-convection problems, sub-stepping did not play a role in increasing the accuracy of the solution, but it did increase the runtime. Exceptions to this are discussed further in the results.

Cauchy solvers CRAM, Parabolic and Hyperbolic have the ability to run with different orders. In the case of CRAM, these coefficients for different orders need to be precomputed, but the Parabolic and Hyperbolic solvers can have their coefficients computed on the fly. For the results shown in this report, the default order for CRAM is $N = 16$ and for Parabolic and Hyperbolic, the order is $N = 32$.

In some of the tests each of the matrix exponential algorithms achieves the same error to a number of significant figures. For these cases, the reported error represents each of the six matrix exponential methods. Runtimes for each of the algorithms are different, and is still shown in these cases.

V.A. Spatial Convergence Study

V.A.1. Diffusion

The first diffusion problem consisted of a 2D system shown as:

$$\frac{\partial U}{\partial t} = k\frac{\partial^2 U}{\partial x^2} + k\frac{\partial^2 U}{\partial y^2}, \tag{43}$$

on the domain $x \in [0, 1]$, $y \in [0, 1]$, subject to periodic boundary conditions and initial condition:

$$U(x, y, 0) = \sin(2\pi x)\sin(2\pi y) \tag{44}$$

with solution:

$$U(x, y, t) = e^{-t} \sin(2\pi x) \sin(2\pi y), \tag{45}$$

with $k = 1/(8\pi^2)$. The problem was run for a total time of 2.0 seconds, with the number of cells in the x and y directions being the same [10, 20, 40, 80]. Errors and convergence rates for each of the solvers were the same and are presented in Table I.

TABLE I
Convergence rate for diffusion problem 1 using absolute error

Cells	E_∞ Rate	E_1 Rate	E_∞ Error	E_1 Error
100	-	-	4.39e-03	2.20e-04
400	2.02	2.02	1.08e-03	4.53e-04
1,600	1.97	2.01	2.76e-04	1.13e-04
6,400	1.99	2.00	6.94e-05	2.82e-05

These results show good convergence rates for the E_∞ and E_1 error functions, showing second order convergence for the diffusion operator. While each of the solvers maintained the same error, the runtimes were drastically different. Runtimes for all of the solvers are shown in Figure 5.

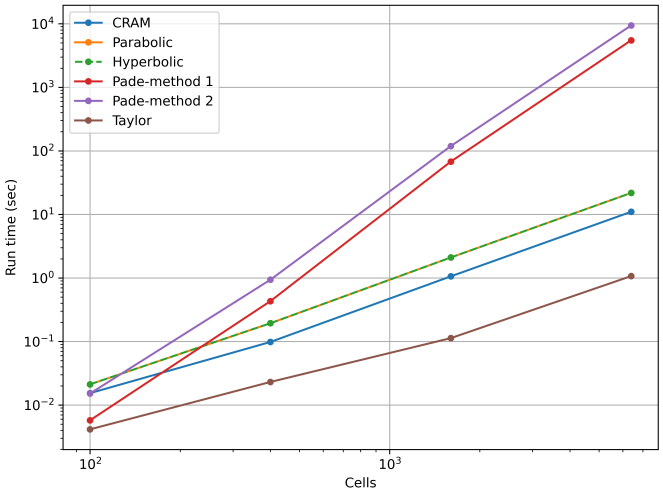

Fig. 5. Runtime performance for diffusion problem

As seen in Figure 5, each of the Cauchy solvers showed a monomial relation between the problem size and run time. The Taylor solver also showed this relation, but with a slightly convex

shape. The parabolic and hyperbolic solvers have almost the same solve time. This is because each solver must compute the solution to 16 linear systems. The CRAM solver requires half the number of linear solves, making it about twice as fast. For this example, the Taylor solver was the fastest but it was only slightly faster than either of the Cauchy solvers. The Padé solvers show poor scaling with an increased number of spatial cells.

Due to the relatively long solve times, particularly for the Padé solvers, the Krylov subspace approximation was used to analyze the error and runtime. For a spatial resolution of 160 cells in both the x and y direction, results for various subspace dimensions M are shown in Table II.

TABLE II
Error and run times for different Krylov subspace dimensions

Solver	M	E_∞ Error	E_1 Error	Run time (sec)
Padé-method 1	5	1.74e-05	7.05e-06	1.13e-02
-	10	1.74e-05	7.05e-06	1.47e-02
-	25	1.74e-05	7.05e-06	3.31e-02
-	50	1.74e-05	7.05e-06	9.27e-02
-	100	1.74e-05	7.05e-06	3.33e-01
-	150	1.74e-05	7.05e-06	7.30e-01
-	200	1.74e-05	7.05e-06	1.33e+00
Padé-method 2	5	1.74e-05	7.05e-06	9.15e-03
-	10	1.74e-05	7.05e-06	1.18e-02
-	25	1.74e-05	7.05e-06	2.96e-02
-	50	1.74e-05	7.05e-06	8.95e-02
-	100	1.74e-05	7.05e-06	3.28e-01
-	150	1.74e-05	7.05e-06	8.20e-01
-	200	1.74e-05	7.05e-06	1.55e+00

Interestingly, in Table II, the error associated with reducing the overall dimension of the problem did not change, even though the runtime drastically decreased leading to the conclusion that the dimension of the true space is much smaller.

V.A.2. Reaction-diffusion

In the second example, a 1D reaction-diffusion problem was modeled. This problem was taken from work performed by Chou et al. ([11]). The partial differential equations (PDEs) are as follows:

$$\begin{aligned}\frac{\partial U}{\partial t} &= d\frac{\partial^2 U}{\partial x^2} - aU + V,\\ \frac{\partial V}{\partial t} &= d\frac{\partial^2 V}{\partial x^2} - bV,\end{aligned} \tag{46}$$

on the domain $x \in [0, \pi/2]$. The system is subject to the following boundary conditions:

$$\frac{\partial U}{\partial x}(0,t) = 0, \quad \frac{\partial V}{\partial x}(0,t) = 0, \quad U(\frac{\pi}{2},t) = 0, \quad V(\frac{\pi}{2},t) = 0, \tag{47}$$

with the following initial condition:

$$U(x,0) = 2\cos(x), \quad V(x,0) = (a-b)\cos(x). \tag{48}$$

The exact solution is given by

$$\begin{aligned}U(x,t) &= \left(e^{-(a+d)t} + e^{-(b+d)t}\right)\cos(x),\\ V(x,t) &= (a-b)e^{-(b+d)t}\cos(x).\end{aligned} \tag{49}$$

The same three test problems that were conducted in the work by Chou et al. ([11]) were also conducted in libowski. These test results correspond to changes in coefficients $[a, b, d]$, which produced a diffusion-dominated system [0.1, 0.01, 1.0], a reaction-dominated system [2.0, 1.0, 0.001], and a stiff reaction system [100, 1.0, 0.001]. Each test case was run with one time step to $t = 1$, so $\Delta t = 1$. The number of cells in the x direction was varied from 10 to 320. Results for the spatial convergence for the CRAM solver for the diffusion-dominated, reaction-dominated and stiff reaction–dominated cases are shown in Tables III, IV, and V. Each solver produced the same error at up to five decimal places for this set of n values.

While there is little difference between the errors from these solvers for this first problem, there is a notable difference in runtimes for the Padé and Taylor solvers. With $n = 1000$, runtimes for each solver are shown in Table VI.

Table VI shows that CRAM was the fastest solver, followed by the parabolic and hyperbolic

TABLE III
Convergence rate for diffusion–dominated problem using absolute error

n	E_∞ Rate	E_1 Rate	E_∞ Error	E_1 Error
20	2.00	2.00	1.87e-04	1.19e-04
40	2.00	2.00	4.69e-05	2.99e-05
80	2.00	2.00	1.17e-05	7.46e-06
160	2.00	2.00	2.93e-06	1.87e-06
320	2.00	2.00	7.33e-07	4.67e-07

TABLE IV
Convergence rate for reaction–dominated problem using absolute error

n	E_∞ Rate	E_1 Rate	E_∞ Error	E_1 Error
20	2.00	2.00	2.23e-04	1.42e-04
40	2.00	2.00	5.58e-05	3.55e-05
80	2.00	2.00	1.40e-05	8.88e-06
160	2.00	2.00	3.49e-06	2.22e-06
320	2.00	2.00	8.72e-07	5.55e-07

TABLE V
Convergence rate for stiff reaction–dominated problem using absolute error

n	E_∞ Rate	E_1 Rate	E_∞ Error	E_1 Error
20	2.00	2.00	9.42e-03	6.00e-03
40	2.00	2.00	2.36e-03	1.50e-03
80	2.00	2.00	5.89e-04	3.75e-04
160	2.00	2.00	1.47e-04	9.38e-05
320	2.00	2.00	3.68e-05	2.34e-05

TABLE VI
Runtime (sec) for reaction-diffusion problem with 1,000 mesh cells

	Problem Domination		
Solver	Diffusion	Reaction	Stiff Reaction
CRAM	1.98e-02	2.54e-02	2.87e-02
Parabolic	3.01e-02	3.08e-02	3.07e-02
Hyperbolic	2.97e-02	3.08e-02	2.88e-02
Padé method-1	9.90e+01	5.84e+01	6.02e+01
Padé method-2	1.37e+02	7.77e+01	8.07e+01
Taylor	2.36e+02	1.33e-01	1.55e-01

solvers. Both of the Padé solvers had drastically much longer run times for the diffusion-dominated cases, with Method 2 being the longest. Interestingly, the Taylor solver showed dramatically different runtimes based on the physical process dominating the problem. In the diffusion-dominated cases, the Taylor solver took the longest, but in the reaction-dominated cases, the runtime was on the order of the CRAM, parabolic, and hyperbolic solvers.

The Krylov subspace approximation was applied to each of the sub-problems with 1,000

cells in the x direction. The Krylov dimensions were varied from 2 to 100, and the results are shown in Figure 6. Unlike the previous diffusion example, a much larger Krylov dimension was required to reach a converged error for the diffusion dominated case. The reaction dominated cases showed faster convergence rates as a function of Krylov dimension. Each solver showed the same error behavior, but with slightly different converged errors due to numerical accuracy in the algorithms. Each solver also showed similar behavior, with runtime scaling as a function of the Krylov dimension; however, each axis was scaled differently.

V.A.3. Convection

Convection was tested using PDEs of the form

$$\frac{\partial U}{\partial t} = -v\frac{\partial U}{\partial x}, \tag{50}$$

where v is velocity. Two tests were shown to demonstrate the problem on numerical diffusion and to show the second-order accuracy of the TVD scheme. First, Eq. (50) was applied to a system on the domain $x \in [0, 100]$, $t \in [0, 20]$ subject to the following boundary conditions:

$$U(0, t) = 1.0, \quad \frac{\partial U}{\partial x}(100, t) = 0.0, \tag{51}$$

and the initial condition:

$$U(x, 0) = 0.0. \tag{52}$$

Results at $t = 20$ for a first-order upwind differencing scheme are shown in Figure 7 for a first-order backward differencing formula (BDF1) and the ETD scheme. This figure depicts the temporal accuracy of ETD schemes when compared to traditional time integration methods. In just a single time step, ETD achieved the same level of accuracy that BDF1 achieved as $\Delta t \to 0$.

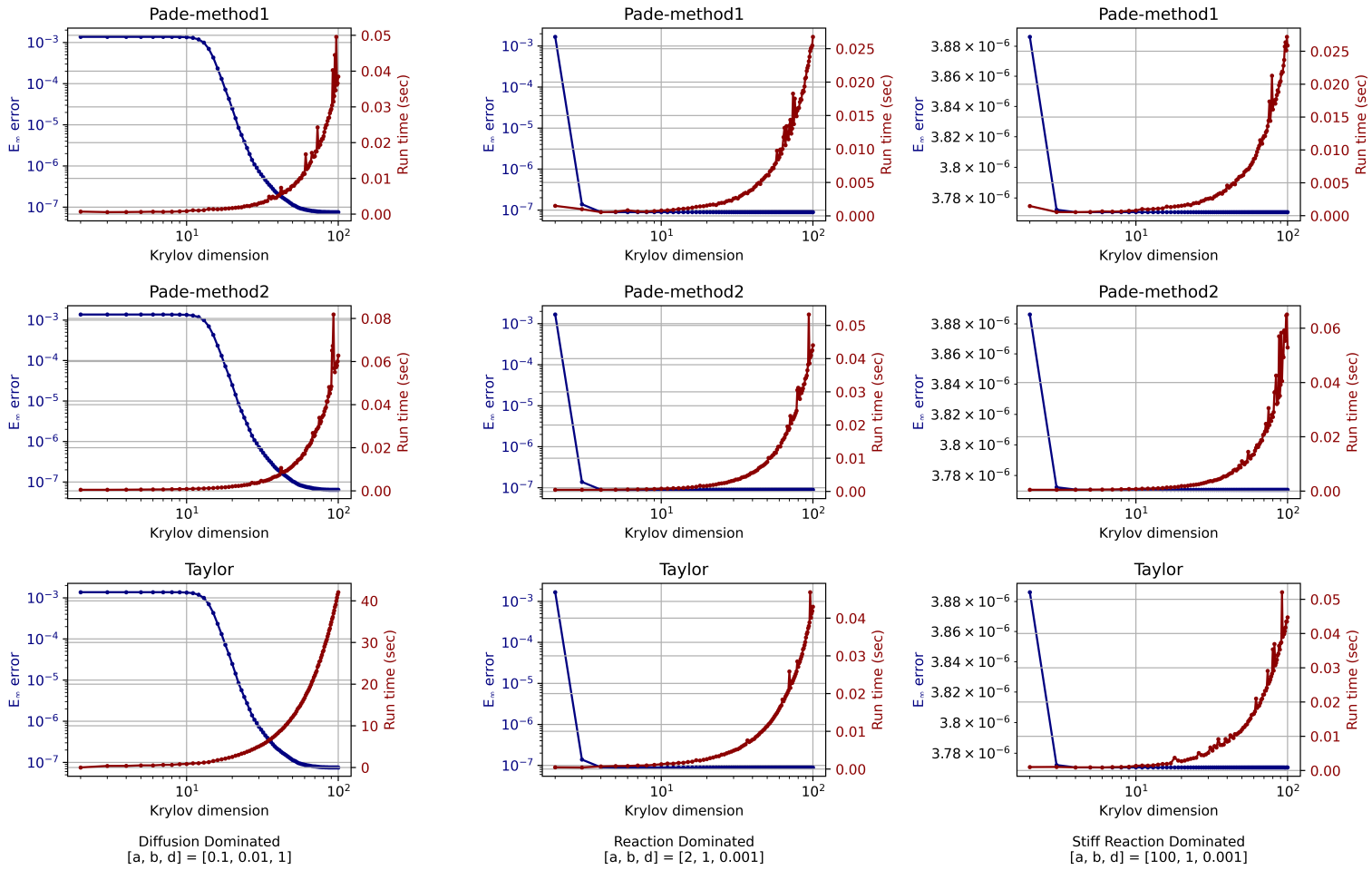

Fig. 6. Results for the Krylov subspace approximation for each sub-problem

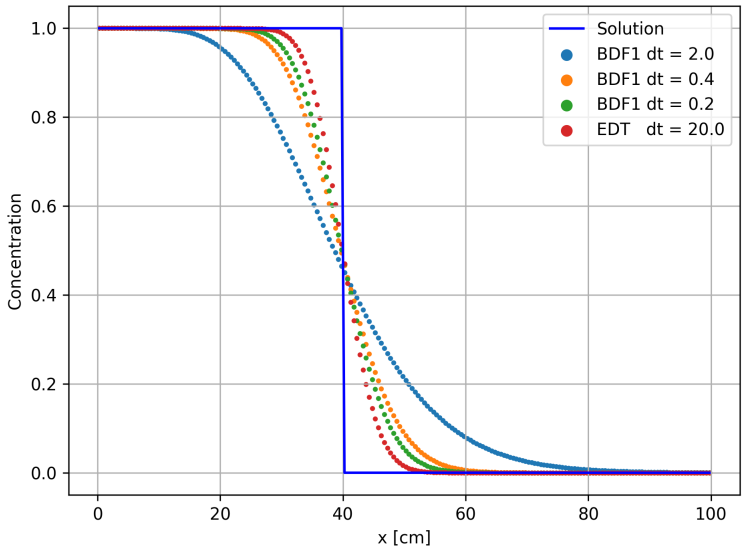

Fig. 7. First-order upwind differencing, dx = 0.5, v = 2

Second-order upwind flux approximations are discussed above, and while there are a number of flux limiter functions to choose from, only the Superbee and MUSCL functions are presented in this work due to their popularity. For the same problem depicted in Figure 7, the second-order TVD scheme with Superbee and MUSCL limiters, along with the first-order upwind scheme, are shown in Figure 8. Because of the deferred corrections implementation, many time steps were needed to accurately add the second-order flux correction. While this method was not second order for discontinuous functions, it did increase the accuracy of handling discontinuous functions.

To show the convergence rate for the TVD scheme, the problem depicted by Eq. (50) is solved using a smooth initial condition:

$$U(x,0) = e^{-((x-30)/10)^2} \tag{53}$$

with periodic boundary conditions on the domain $x \in [0, 100]$, $t \in [0, 2]$. Because of the finite volume discretization, the initial condition must be implemented using the mean value theorem (MVT). Results for EDT are shown in Tables VII, VIII and IX using absolute error. These results are shown at t = 2, with dt = 0.1.

Table VII shows that the first-order upwind flux has a convergence rate of about one for E_∞ and E_1. The Superbee limiter shows a better convergence rate than first-order upwind, but it

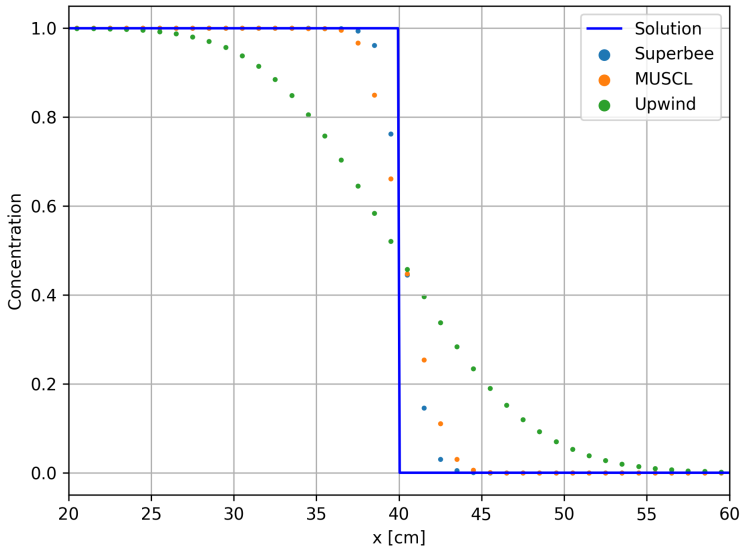

Fig. 8. Comparison of flux limiters, dx = 1.0, dt = 0.2, v = 2

shows a convergence rate a little lower than 2. Table IX shows that the MUSCL limiter had the best convergence rate, which is at or above 2 for the E_1 error.

TABLE VII
Convergence rate for smooth convection problem first-order upwind using absolute error

n	E_∞ Rate	E_1 Rate	E_∞ Error	E_1 Error
20	1.00	0.95	1.41e-01	2.92e-02
40	0.66	0.83	8.87e-02	1.64e-02
80	0.90	0.97	4.76e-02	8.36e-03
160	0.96	0.98	2.44e-02	4.24e-03
320	0.98	0.99	1.23e-02	2.13e-03

TABLE VIII
Convergence rate for smooth convection problem superbee limiter using absolute error

n	E_∞ Rate	E_1 Rate	E_∞ Error	E_1 Error
20	2.10	2.06	5.69e-02	1.17e-02
40	0.88	1.50	3.10e-02	4.13e-03
80	1.15	1.64	1.40e-02	1.33e-03
160	0.75	1.71	8.27e-03	4.07e-04
320	1.24	1.82	3.50e-03	1.15e-04

Runtimes for the smooth initial condition are shown in Figure 9, shown as an average of the flux limiter functions. These problems show an opposite trend to the diffusion problems, the Cauchy solvers show higher runtimes until the number of cells reach a large value. The Padé

TABLE IX
Convergence rate for smooth convection problem MUSCL limiter using absolute error

n	E_∞ Rate	E_1 Rate	E_∞ Error	E_1 Error
20	1.85	1.97	6.82e-02	1.25e-02
40	0.99	2.00	3.44e-02	3.13e-03
80	1.54	2.10	1.18e-02	7.30e-04
160	1.47	2.05	4.28e-03	1.76e-04
320	1.57	2.05	1.44e-03	4.26e-05

solvers begin at a low runtime but scale poorly as a function of matrix size. The Taylor solver shows the best results as far as runtime, being that it is the lowest for all values of dx.

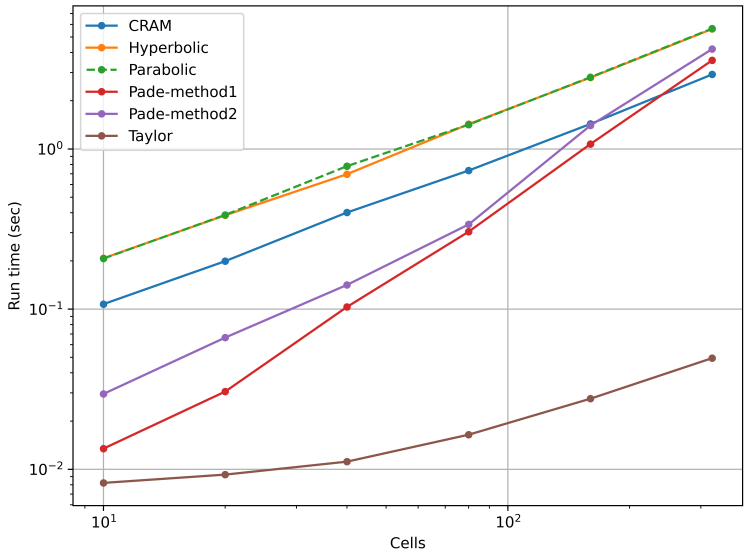

Fig. 9. Runtimes for smooth convection problem, dt = 0.01, v = 2

V.B. Neutron Precursors

This problem examines a convection-driven flow with the 6 neutron precursor groups, as shown in Eqs. (54) and (55):

$$\frac{\partial C_i}{\partial t} = -v_x \frac{\partial C_i}{\partial x} - v_y \frac{\partial C_i}{\partial y} + \beta_i \Psi(x,y) - \lambda_i C_i, \tag{54}$$

$$\Psi(x,y) = \begin{cases} \psi_0 \sin\left(\frac{\pi x}{50}\right) \sin\left(\frac{\pi y}{100}\right), x \in [0,50], y \leq 100 \\ 0 \text{ , otherwise,} \end{cases} \tag{55}$$

where $i \in [1,6]$, $x \in [0,50]$, $y \in [0,400]$, $t \in [0,60]$, $v_x = 0$, and $v_y = 25$, subject to the following boundary conditions:

$$C_i(x,0) = C_i(x,400), \quad \frac{dC_i}{dx}(0,y) = 0, \quad \frac{dC_i}{dx}(50,y) = 0. \tag{56}$$

Coefficients for the system are shown in Table X [35]. Each precursor had the same initial condition of zero, and the source term for each precursor was scaled in the x and y directions by the sine function. The spatial domain was modeled to mimic an MSR with a core region extending from $y \in [0,100]$ and $x \in [0,50]$, with a core exterior loop modeled from $y \in [100,400]$.

TABLE X
Parameters for Neutron Precursors

Group	λ	β
1	0.0127	0.0006
2	0.0317	0.00364
3	0.115	0.00349
4	0.311	0.00628
5	1.4	0.00179
6	3.87	0.0007

While there is no analytic solution for this example problem, a reference solution was generated in Matlab using the symbolic tool box to solve for the matrix exponential. To make the analytic matrix exponential easier to solve in Matlab, the first order upwind scheme was used for the convection operator. A small, spatially discretized problem was set up with 5 cells in the x direction and 20 in the y direction. The transition matrix that was built was then exported to Matlab, and the matrix exponential was solved at various times and saved as the reference solution. While the spatial accuracy was not tested in this case, this method will access the accuracy of each matrix exponential algorithm. To further simplify the problem, the first-order upwind difference scheme was applied to the convective flux.

One important feature for many of these solvers was the location of the eigenvalues for the transition matrix. The spectrum was calculated using the Matlab symbolic tool box and is plotted in Figure 10 various values of dt. Figure 10 show 6 elliptical rings, one at each dt, each one

representing a precursor group. For a given time step size, the eigenvalues shifted along lines with slopes that are the ratio of their real and imaginary parts [32]. For a system in which the eigenvalues are located in a region where solutions based on Cauchy's integral break down, these eigenvalues can be shifted into a region where the solutions hold. These eigenvalues can be shifted either by using sub-stepping or by reducing the time step size. Figure 10 shows how changing the time step size can shrink the real and imaginary parts of the eigen spectrum.

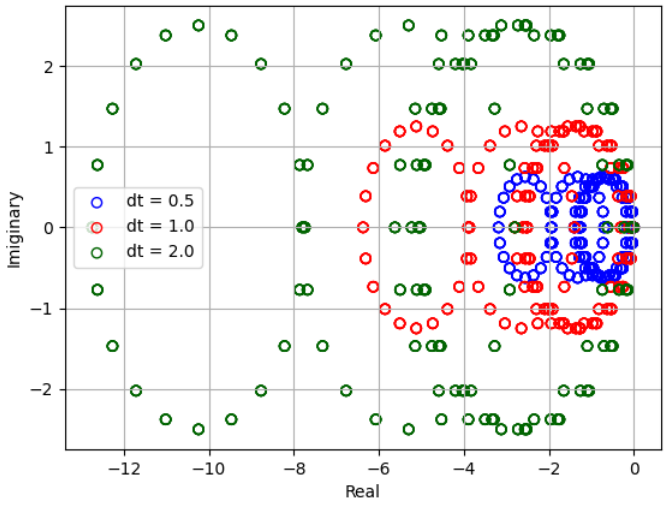

Fig. 10. Spectrum for the Neutron Precursors

To understand how sub-stepping can work to improve the accuracy of a Cauchy-based solver, the neutron precursors problem was computed at 10-second time step intervals. The location of the eigenvalues is a function of the ratio v_x/dx in the transition matrix; therefore, at the prescribed discretization, this ratio was manipulated by changing the flow velocity. For each Cauchy solver, the eigenvalues at two different flow velocities, 25 and 50, are shown in Figure 11. As shown in Figure 11, as the ratio increased, so did the spread of the eigenvalues on the real and imaginary axis. This led to a limitation of the velocity to discretization size for convection problems when using Cauchy solvers. One solution, which is also shown in Figure 11, is to use sub-stepping to reduce the time step size, thus confining the eigenvalues. As the number of substeps is increased, the eigenvalues become confined in a region in which the contour encloses the spectrum, theoretically increasing the solver's accuracy. Each plot in Figure 11 shows how the the spectrum was confined

using 0, 2 and 6 substeps. As the number of substeps increased, the spectrum shrank into the confines of the contour.

The relative E_∞ error for each solver is shown in Figure 12 for low and high velocity cases, with each of the Cauchy solvers shown with six substeps. These results indicate that regardless of the flow velocity, the series solvers maintained about the same error. This is because the matrix l_1 norm was not greatly increased by this change in flow velocity. The Padé-method 1 performed the worst of the six solvers, whereas Taylor performed the best, achieving a remarkably accurate approximation to the transition matrix. It is also interesting to note how the Cauchy solver error decayed, while the series solver errors tended to increase with time.

Errors relative to the Matlab symbolic tool box for each Cauchy solver as a function of substeps for both velocities of 25 and 50 are shown in Figure 13. Each Cauchy solver started out with a large error with zero substeps for both cases, with the error for v = 50 being larger than v = 25. This is to be expected because the eigenvalues had a larger imaginary component which pushed them past the contour. For each case, every Cauchy solver increased in accuracy as the number of substeps increased. Another interesting note is the rate at which each of the solvers errors converge as a function of time. In the high velocity case, the errors tend to linearly decay, especially with a low number of substeps. For the low velocity case, the errors tend to exponentially decay. Looking at Figure 11, the eigenvalues for the Parabolic solver fall outside of the quadrature points with zero substeps. This coincides with the error for the Parabolic being large with zero substeps.

Sub-stepping increases the accuracy of the Cauchy solvers, but the manner in which it is implemented has a negative effect on runtime. The runtime performance for each solver is shown in Table XI. As expected, runtime for the Cauchy solvers increased linearly with the number of substeps. CRAM was the fastest solver for zero substeps, with Taylor being close behind.

V.C. Simple MSR Depletion

A simplified version of equation 8 is presented to show depletion with flowing fuel:

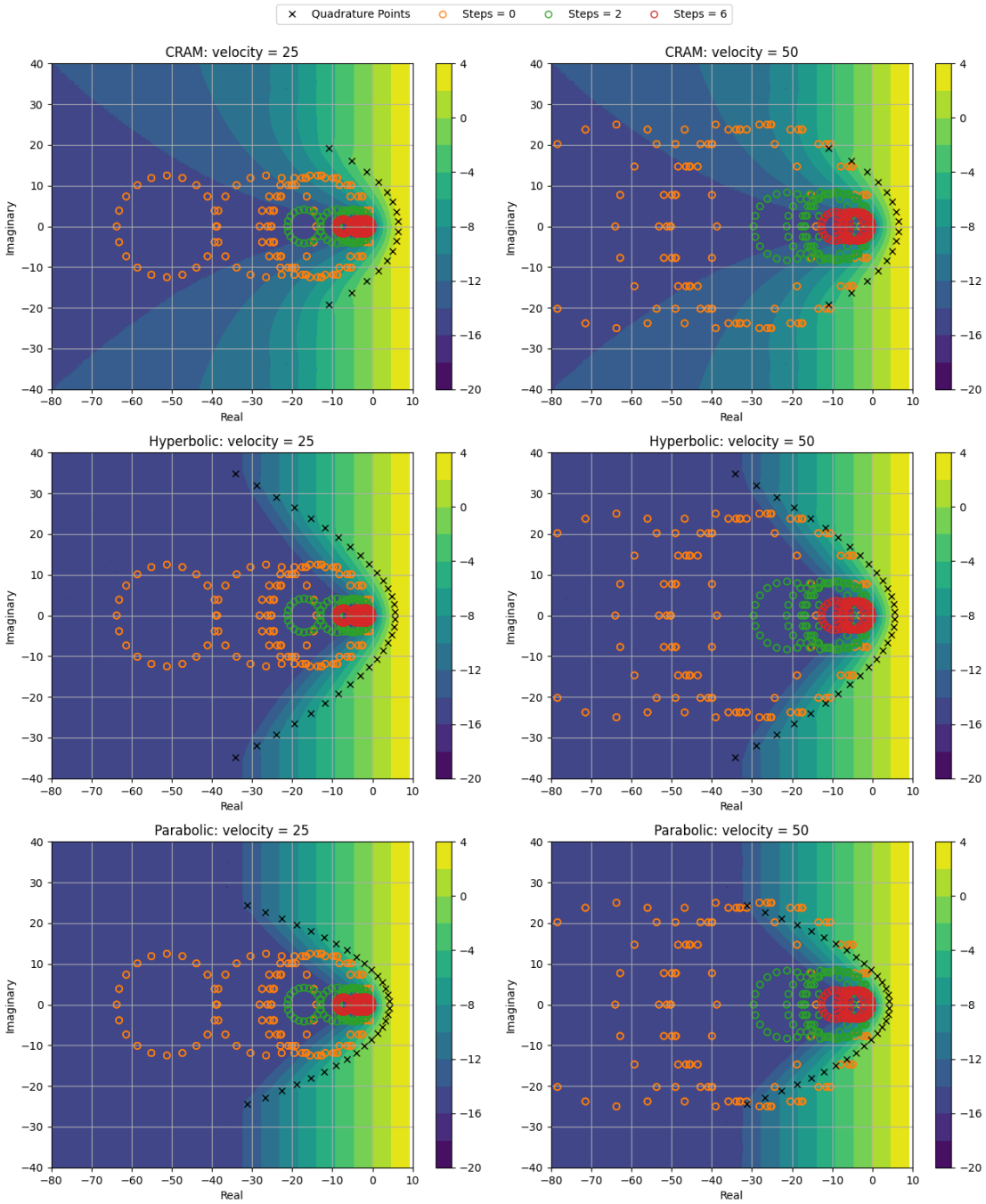

Fig. 11. Eigenvalues of the neutron precursor problem superimposed on Cauchy scalar error plots

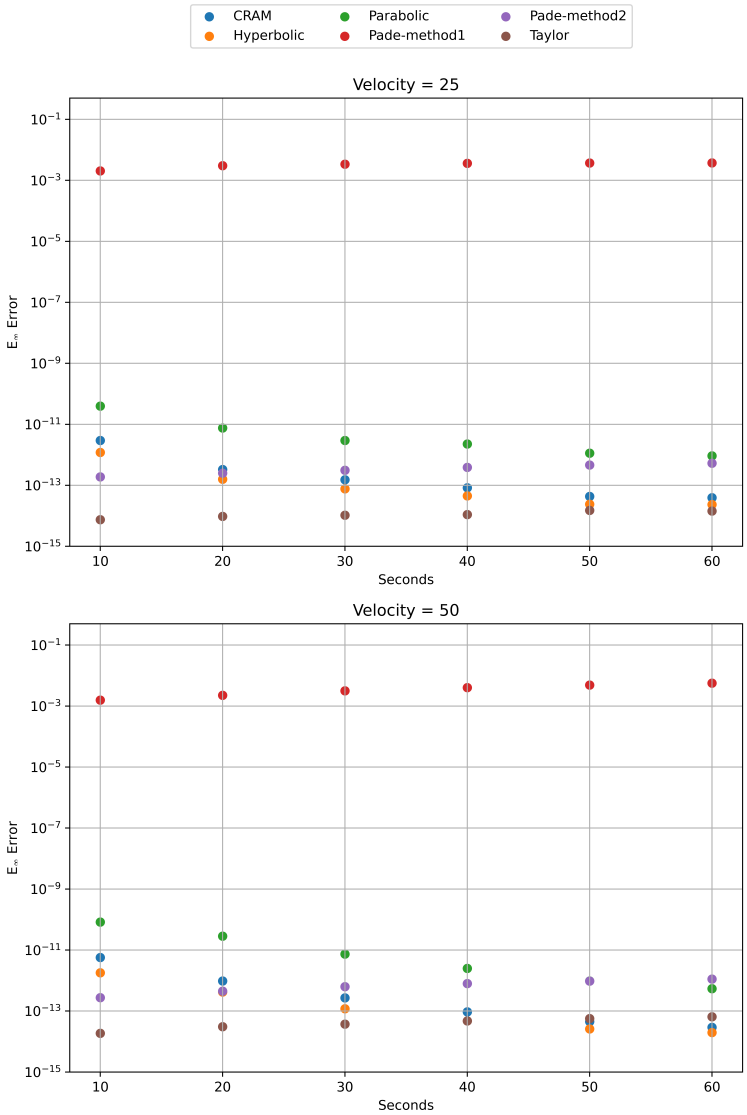

Fig. 12. Relative E_∞ error for neutron precursors

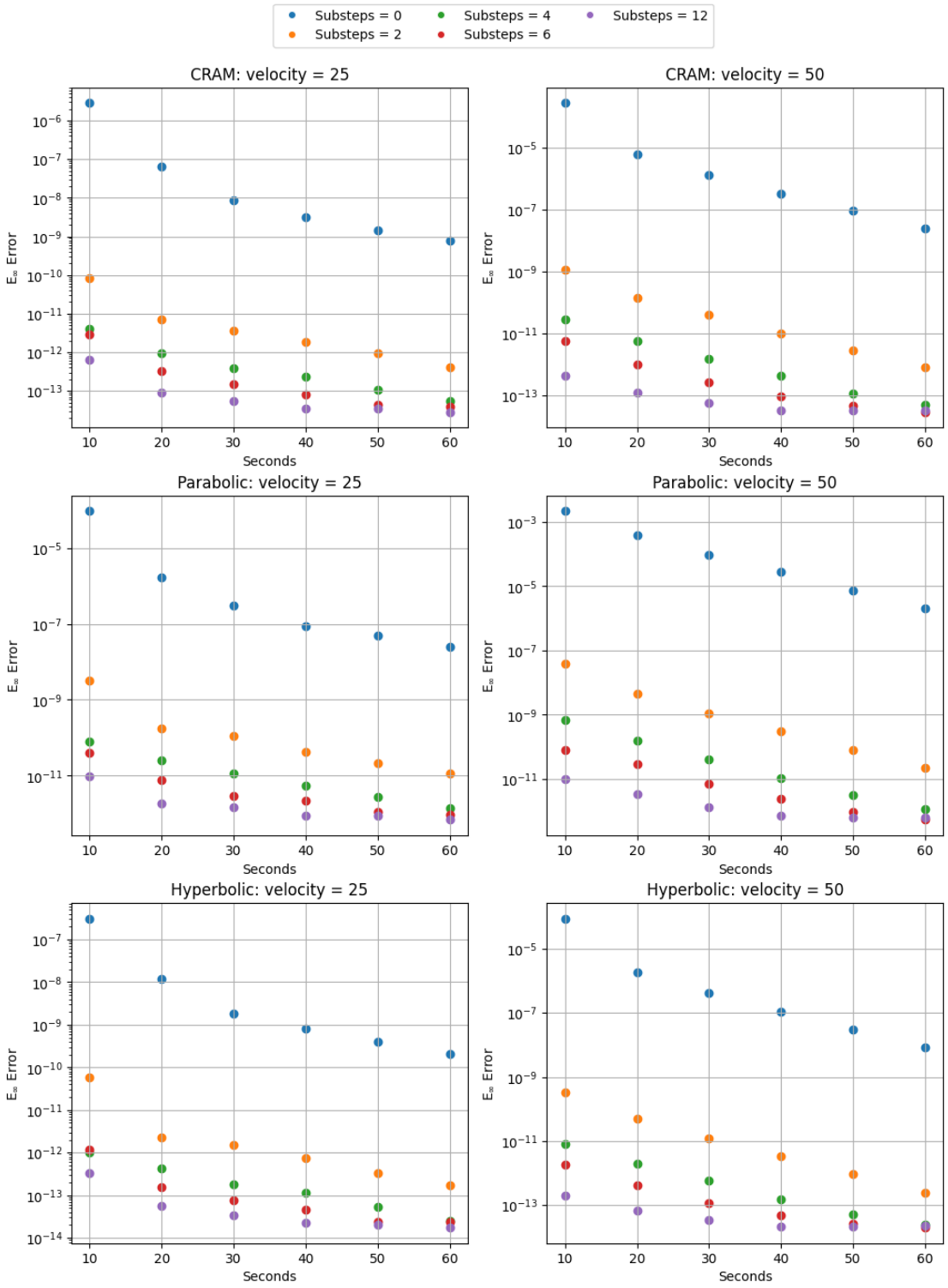

Fig. 13. Neutron precursor errors as a function of substeps for Cauchy solvers

TABLE XI
Runtime in seconds for neutron precursor problem

	Substeps				
	0	2	4	6	12
CRAM	0.715	2.16	3.59	5.01	9.33
Parabolic	1.42	4.24	7.07	10.0	18.5
Hyperbolic	1.42	4.26	7.09	9.98	18.5
Padé-method 1	1.84	NA	NA	NA	NA
Padé-method 2	3.38	NA	NA	NA	NA
Taylor	0.735	NA	NA	NA	NA

$$\frac{\partial \rho_i}{\partial t} + v_y \frac{\partial \rho_i}{\partial y} = \sum_{j=1}^{N} \frac{M_i}{M_j} \left(b_{j \to i} \lambda_j + \sum_{k=1}^{K} \gamma_{j \to i, k} \sigma_{k,j} \phi \right) \rho_j - \left(\lambda_i + \phi \sum_{k=1}^{K} \sigma_{k,i}(r) \right) \rho_i. \tag{57}$$

The spatial domain is $x \in [0, 0.6858]$, $y \in [0, 5.761]$ with $v_y = 0.25$. The total depletion time is 100 days at 20 day time step intervals. The spatial discretization is kept small (3 cells in the x-direction and 9 in y-direction) so that a reference solution can be computed. A small section of 103 isotopes taken from Reference [36] Appendix B was selected, making the total number of equations 2,781. Reaction rates were generated using an ORIGEN library for light-water reactor fuel. After the transition matrix was generated in libowski, it was exported into Matlab to generate the analytical matrix exponential at each of the time step intervals. Again, the first order upwind convection scheme is used to make this solution easier to compute. The initial condition is taken from Reference [37] for the MSRE salt and converted to mass density. The neutron flux follows a sinusoidal shape function in the lower part of the reactor, the 3x3 cell region. The upper 3x6 region does not have a neutron flux. Periodic boundary conditions are added at the top and bottom of the spatial domain to mimic a flow loop.

Eigenvalues for the problem are shown in Figures 14 and 15 for both base eigenvalues of the transition matrix and ones adjusted for the 20 day time step size. Figure 14 shows an elliptic ring of nine eigenvalue clusters which are close to the negative real axis and when combined with Figure 15 shows that the base eigenvalues behave well for Cauchy solvers. Each one of these nine eigenvalue clusters it self contains many sets of eigenvalues. When the base line eigenvalues are

plotted in Figure 14, the are zoomed out so that they appear as a single point. When adjusted for the 20 day time step size, the eigenvalue clusters are pushed far on the negative real axis and on the imaginary axis as well. This seemly does not effect the accuracy of Cauchy solvers based on the placement of the eigenvalues based on Figure 15. The adjusted eigenvalues in the left column in Figure 15 only show the cluster at the origin of the axis.

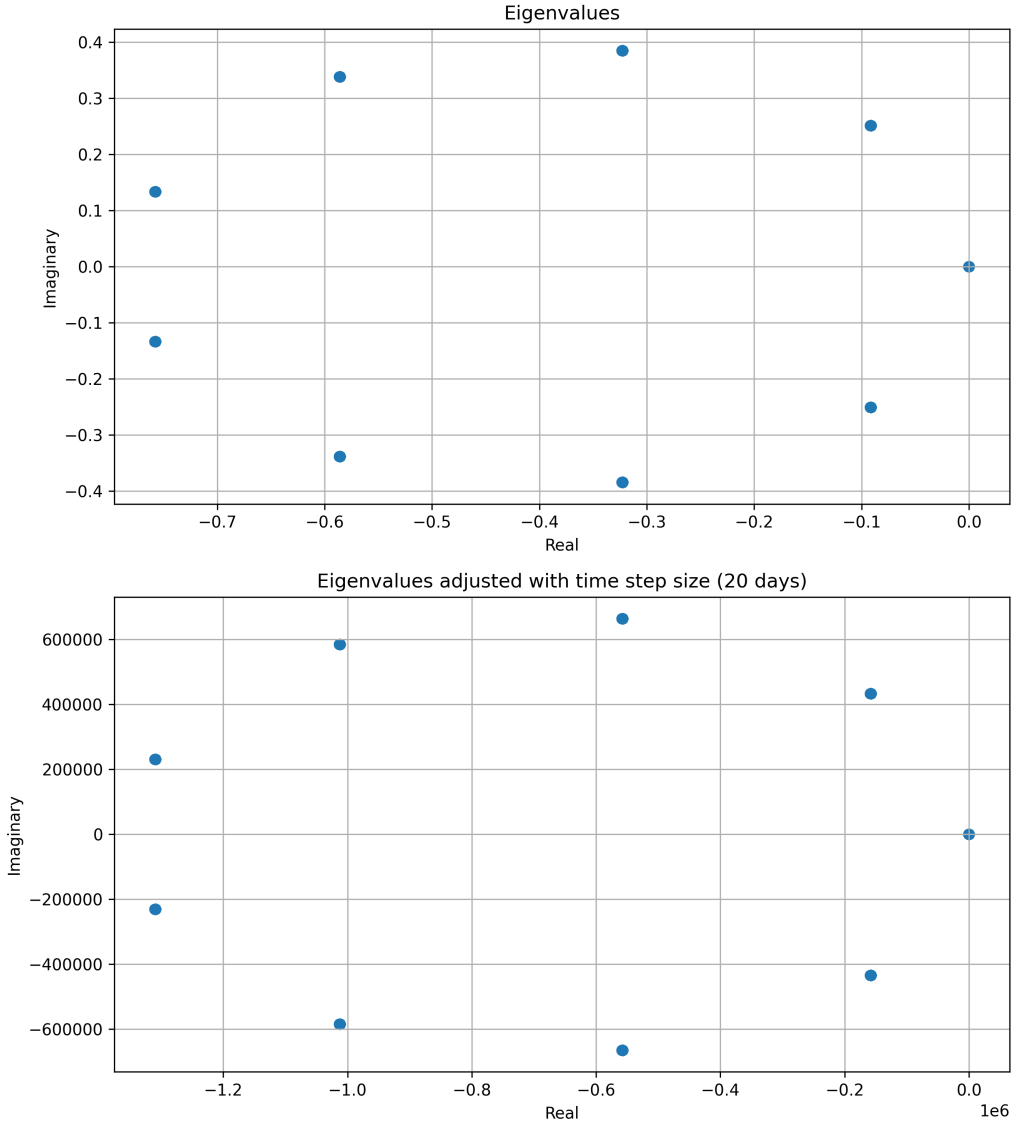

Fig. 14. Base eigenvalues for MSR depletion and adjusted with time step size

Error for each of the solvers is shown in Figure 16. The Taylor, Hyperbolic, CRAM and Padé-method 1 solvers all begin with low errors which gradually increase. The Parabolic solver

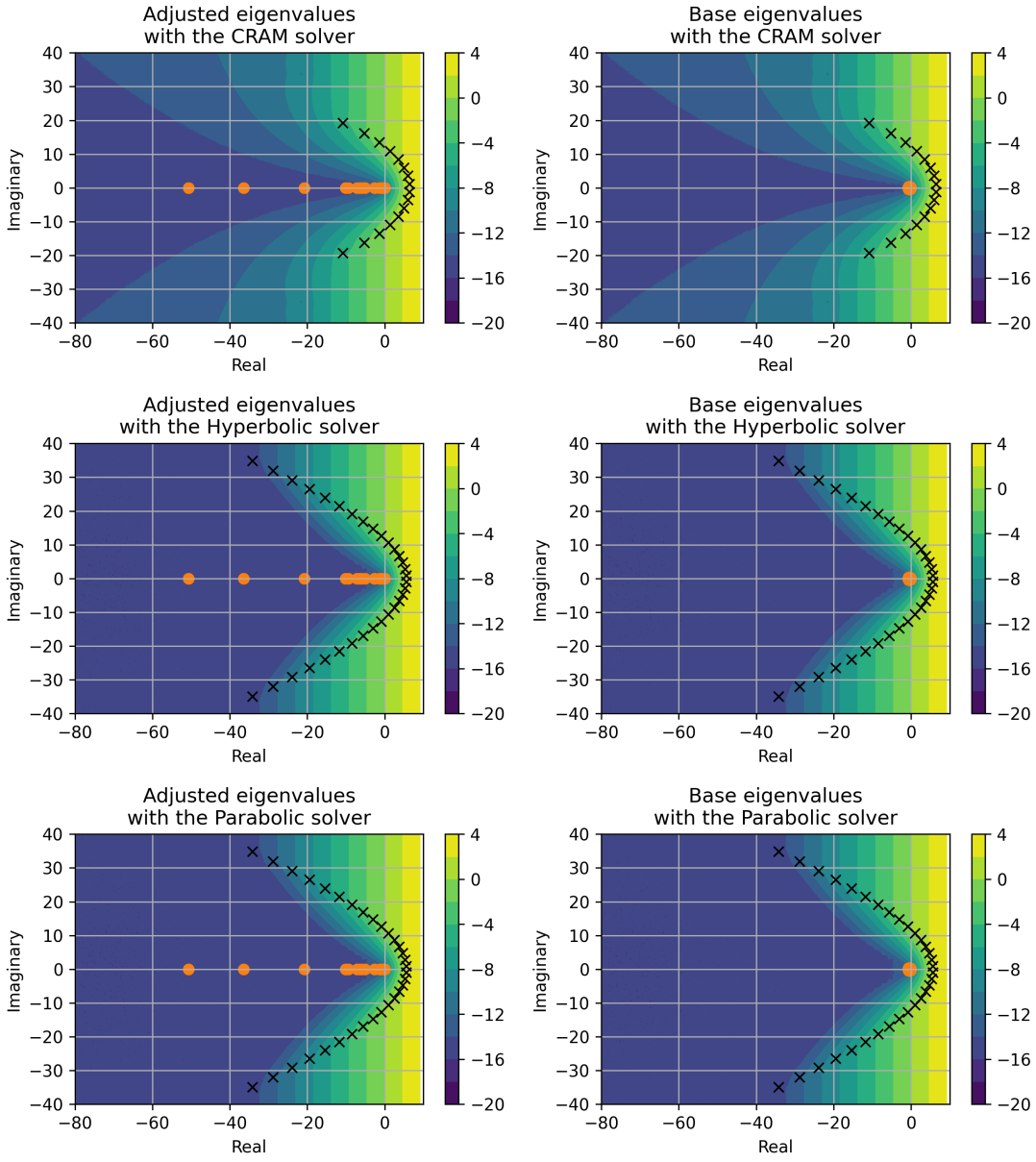

Fig. 15. Base eigenvalues for MSR depletion and adjusted with time step size superimposed on Cauchy scalar error plot

begins with a high error, which decreases rapidly until following the other solvers. Padé-method 2 follows the same trend as the other solvers but has a larger overall error. Substepping can be used to increase the accuracy of the Cauchy solvers. The error as a function of substep for the first time step at 20 days is shown for each of the Cauchy solvers in Figure 17. As the number of substeps increases, the error for CRAM and Hyperbolic reach a constant level. The Parabolic solver has the worst error of the three Cauchy solvers and does not reach the constant value over the substep range. The reduction in error for Cauchy solvers with substepping can come from three possible avenues. The first is moving the eigenvalues into a region where the solver will be more accurate. The eigenvalues for this problem do create an epileptic ring on the left hand side of the imaginary plane, but the imaginary values of these eigenvalues are already in a region far enough down the negative real axis, that their error contributions should not be seen. The second is that the nuclide concentrations are small enough over the time step range that they might not be accurately captured using the Cauchy solvers. This might be the case for some of the nuclides due to the relatively small concentrations. The last has to do with the generation pathways for fission products. In fresh fuel calculations the generation rates for all nuclides depends on the matrix coefficients of a few transitions. If the error of calculating the matrix exponential is large for these few transitions, then the overall error in the solution can be greater, than in depleted fuel calculations. Error is largest in the first step for Cauchy solvers then seems to diminish as the fuel is depleted, this reduction in error could possibly come from both the second and third points. More analysis will need to be conducted to understand this behavior.

Runtimes for each solver is shown in Table XII for the total solve time of the problem, this includes the five 20 day steps to reach 100 day depletion. With zero substeps the CRAM solver has the lowest solve time, followed by the Parabolic and Hyperbolic solvers, Padé-method 1 runs slightly longer, followed by method 2 and the Taylor solver.

Because of the long runtimes for these solvers, it would be useful to apply the Krylov method to reduce this runtime. In section IV.C it was stated that the eigenvalues of the transition matrix would need to be sufficiently close for the Krylov subspace to be accurate. As the depletion time increases, the spread of the eigenvalues does as well, pushing them further away from one another. So, for the Krylov approximation to be accurate, the time step size must be sufficiently small. To test the accuracy of this method, a single depletion step of 20 days is calculated using different

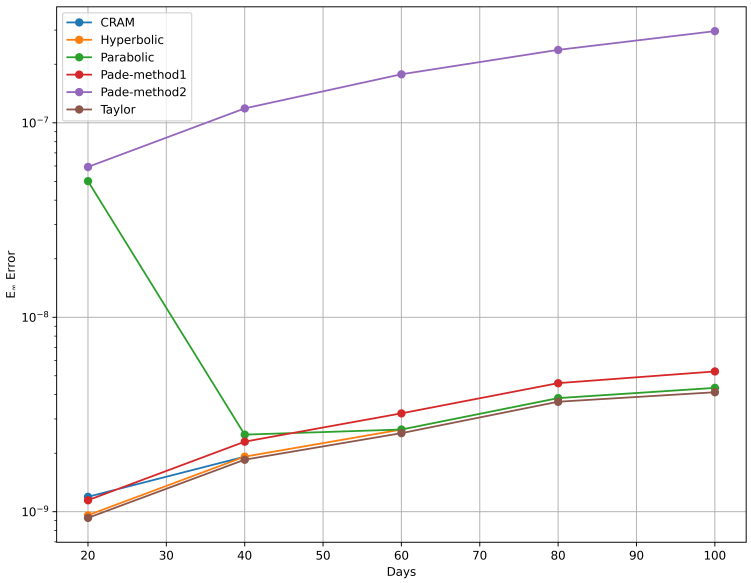

Fig. 16. MSR 2D depletion error. Cauchy solvers are shown with 4 substeps

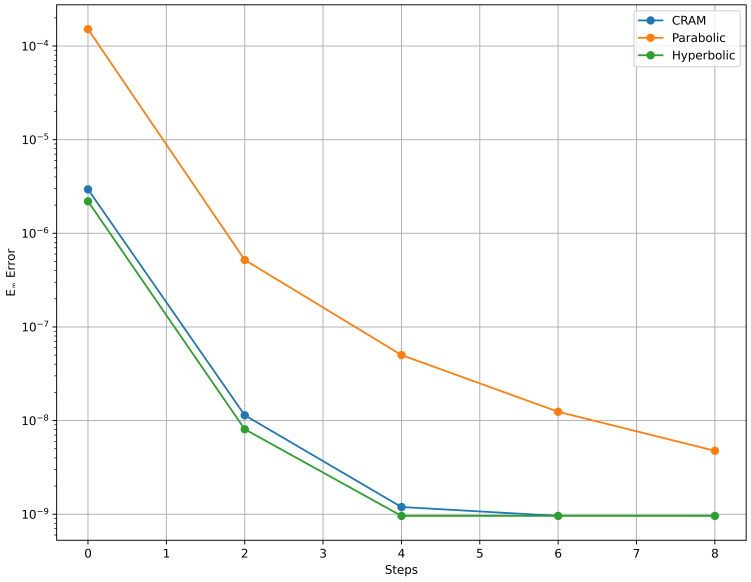

Fig. 17. MSR 2D depletion error at the first time step of 20 days as a function of Cauchy substeps

TABLE XII
Runtime in seconds for MSR 2D depltion problem

	Substeps					
	0	2	4	6	8	10
CRAM	3.65e+02	1.08e+03	1.79e+03	2.48e+03	3.17e+03	3.94e+03
Parabolic	6.47e+02	1.92e+03	3.18e+03	4.43e+03	5.70e+03	7.00e+03
Hyperbolic	6.54e+02	1.96e+03	3.23e+03	4.49e+03	5.77e+03	7.21e+03
Padé-method 1	7.36e+02	NA	NA	NA	NA	NA
Padé-method 2	1.48e+03	NA	NA	NA	NA	NA
Taylor	2.21e+04	NA	NA	NA	NA	NA

dt values to approach it. These values range from a single 20 day time step, down to a time step size of 0.5 days. The results are shown in Figure 18 for a range of Krylov dimensions. Although it appears that the error starts low, increases, then begins to drop again, the solutions for the low Krylov dimensions of 5 and 10 do not converge to a reasonable solution. The concentrations of nuclides are zero or nearly zero, resulting in an error of about 1. With a single time step of 20 days, the error does not decrease to a suitable value over the Krylov dimensions. As the time step size is decreased, the Krylov dimension required for suitable error decreases. Because of the overall size of the reduced order system is much smaller than the original system of 2,781 equations, the problem can run much faster. Runtimes for each solver using the Krylov subspace is shown in Figure 19. These results show great promise in reducing the runtime of this problem while maintaining great accuracy. For example, with a Krylov dimension of 100 at a dt of 0.5 days, the solution has a relative accuracy on the order of 10^{-8} with just a 2 second runtime for the Cauchy solvers. Runtimes for all solvers is also reduced but the Taylor solver still takes the longest. There is slight variation for the Cauchy solvers as a function of Krylov dimension, but the over all trend is up.

VI. CONCLUSION

This paper describes the use of exponential time differencing to solve a large system of parabolic PDEs arising in mass transport and depletion problems in MSRs. First, the mass transport equations were defined to model nuclear depletion in MSRs by combining volumetric nuclear reaction source terms with the species transport equation. Next, this equation was discretized on a finite volume mesh using a second-order diffusive approximation and a variable order TVD scheme for convection. Application of these algebraic approximations to the transport equation

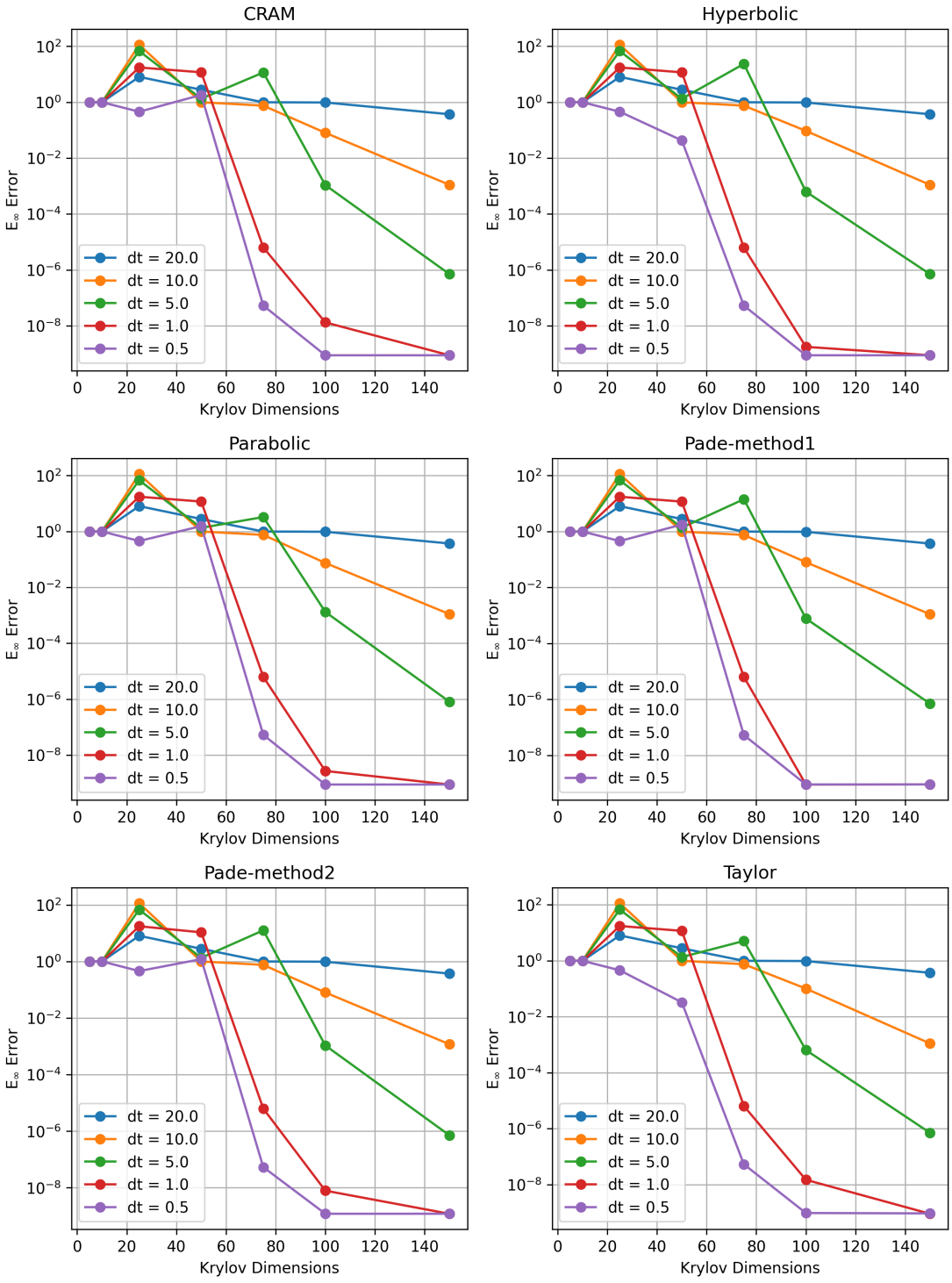

Fig. 18. MSR 2D depletion error with Krylov Dimension and time step size

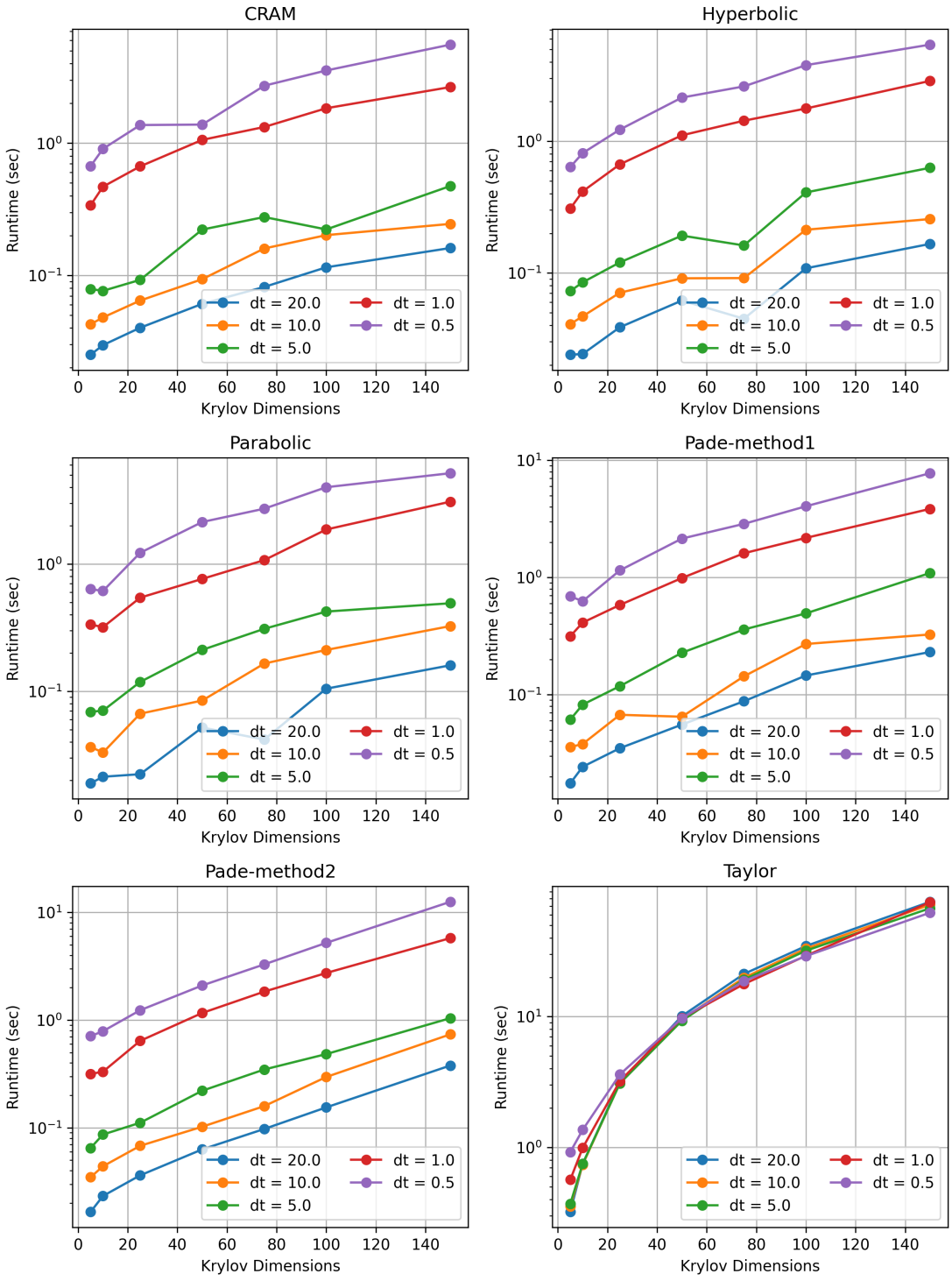

Fig. 19. MSR 2D depletion runtimes with Krylov subspace

transformed the system of PDEs to in a larger system of ODEs. Because of the vast differences in nuclear reaction rates, this system of linear ODEs is very stiff, and the time step lengths required for depletion analysis can be large. Therefore, exponential time differencing is used to integrate the ODEs. A major component of exponential time differencing involves computing the action of the exponential of a matrix on a vector. This computation can be difficult and time consuming, depending on the numerical algorithm. A number of these algorithms are presented here, three involving series approximations, and three involving transforming the computation into the complex plane using Cauchy's integral formula. A number of mass transport tests were conducted to assess the accuracy of the spatial operators, linear source terms, matrix exponential algorithms, and run times of these algorithms. These tests show that the numerical framework developed can be used to solve mass transport problems in MSRs. The framework presented in this paper shows great promise in developing future work for solving these type of problems.

Numerical properties of the transition matrix play a large part in the accuracy of matrix exponential algorithms. Series approximations are most accurate about the origin, indicating that the matrix norm must be sufficiently small. These approximations combine the scaling and squaring property to reduce the matrix norm enough for them to be accurate. However, for large transition matrices, these algorithms can have long run times, so a Krylov subspace is often used in their computation. Solutions based on Cauchy's integral formula do not have a requirement on the matrix norm, but they do have a requirement on placement of eigenvalues. Each of these Cauchy solvers requires that the eigenvalues be located on the left-hand side of the complex plane. Additionally, the imaginary parts of the eigenvalues must be located in a region in which the contour function encloses them. In the case of CRAM, this eigenvalue requirement is more strict. While scaling and squaring are not explicitly implemented in these Cauchy methods, the consequence is in the manner of substeps. Sub-stepping works by moving the eigenvalue spectrum into a region which increases the solver's accuracy for convection diffusion problems. Accuracy can also be improved in general depletion problems by the use of sub-stepping and by reducing the rate at which nuclide concentrations change over a time interval.

Results from the spatial convergence test show excellent convergence rates in both diffusion problems for each matrix exponential solver. Convection problems at higher than first order, as well as the MUSCL limiter, show second-order convergence rates for smooth solutions. In the case

of step changes in Dirichlet boundary conditions, both the Superbee and MUSCL limiter functions limit the amount of numerical diffusion seen in the first-order upwind difference scheme. Results from the neutron precursor problem show how shrinking the magnitude of the imaginary parts increases the accuracy of Cauchy solvers. For larger problems, like the one shown in the MSR depletion example, the Krylov subspace can show good results with decreasing the overall runtime while maintaining reasonable solution accuracy.

Solving mass transport and depletion problems in molten salt reactors has proved to be a complex mathematical problem. These problems involve a large number of isotopes or chemical species, with as many as 2,200 isotopes in large analyses, or up to 90 in small analyses. As the number of cells grows in the system, the number of unknowns increases significantly. When high-fidelity simulation is needed, then these methods must be used in a parallel structure. Future work will involve modeling a larger number of isotopes for reactor analysis, as well as adding mass transport models for phase transition and surface chemical reactions.

ACKNOWLEDGMENTS

This work was funded by the Department of Energy-Office of Nuclear Energy's Nuclear Energy Advanced Modeling and Simulation (NEAMS) program.

REFERENCES

[1] M. Pusa and J. Leppänen, "Computing the Matrix Exponential in Burnup Calculations," *Nuclear Science and Engineering*, **164**, *2*, 140 (2010); 10.13182/NSE09-14.

[2] A. Isotalo and W. Wieselquist, "A method for including external feed in depletion calculations with CRAM and implementation into ORIGEN," *Annals of Nuclear Energy*, **85**, 68 (2015); https://doi.org/10.1016/j.anucene.2015.04.037.

[3] M. Pusa, "Rational Approximations to the Matrix Exponential in Burnup Calculations," *Nuclear Science and Engineering*, **169**, *2*, 155 (2011); 10.13182/NSE10-81.

[4] A. Yamamoto, M. Tatsumi, and N. Sugimura, "Numerical Solution of Stiff Burnup Equation with Short Half Lived Nuclides by the Krylov Subspace Method," *Journal of Nuclear Science and Technology - J NUCL SCI TECHNOL*, **44**, 147 (2007); 10.1080/18811248.2007.9711268.

[5] A. Isotalo and M. Pusa, "Improving the Accuracy of the Chebyshev Rational Approximation Method Using Substeps," *Nuclear Science and Engineering*, **183** (2016); 10.13182/NSE15-67.

[6] B. R. Betzler, J. J. Powers, and A. Worrall, "Molten salt reactor neutronics and fuel cycle modeling and simulation with SCALE," *Annals of Nuclear Energy*, **101**, 489 (2017); https://doi.org/10.1016/j.anucene.2016.11.040.

[7] S. Xia, J. Chen, W. Guo, D. Cui, J. Han, J. Wu, and X. Cai, "Development of a Molten Salt Reactor specific depletion code MODEC," *Annals of Nuclear Energy*, **124**, 88 (2019); https://doi.org/10.1016/j.anucene.2018.09.032.

[8] M. Cheng and D. Zhi-Min, "Development of a three dimension multi-physics code for molten salt fast reactor," *Nuclear Science and Techniques*, **25** (2014); 10.13538/j.1001-8042/nst.25.010601.

[9] C. Wan, T. Hu, and L. Cao, "Multi-physics numerical analysis of the fuel-addition transients in the liquid-fuel molten salt reactor," *Annals of Nuclear Energy*, **144**, 107514 (2020); https://doi.org/10.1016/j.anucene.2020.107514.

[10] S. Cox and P. Matthews, "Exponential Time Differencing for Stiff Systems," *Journal of Computational Physics*, **176**, *2*, 430 (2002); https://doi.org/10.1006/jcph.2002.6995.

[11] C.-S. Chou, Y.-T. Zhang, R. Zhao, and Q. Nie, "Numerical methods for stiff reaction-diffusion systems," *Discrete and Continuous Dynamical Systems - Series B*, **7** (2007); 10.3934/dcdsb.2007.7.515.

[12] H. Ashi, L. Cummings, and P. Matthews, "Comparison of methods for evaluating functions of a matrix exponential," *Applied Numerical Mathematics*, **59**, *3*, 468 (2009); https://doi.org/10.1016/j.apnum.2008.03.039., selected Papers from NUMDIFF-11.

[13] R. Sidje, "Expokit: A Software Package for Computing Matrix Exponentials," *ACM Trans. Math. Softw.*, **24**, 130 (1998); 10.1145/285861.285868.

[14] C. Moler and C. Van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later," *SIAM Review*, **45**, *1*, 3 (2003); 10.1137/S00361445024180.

[15] A. Isotalo and P. Aarnio, "Comparison of depletion algorithms for large systems of nuclides," *Annals of Nuclear Energy*, **38**, *2*, 261 (2011); https://doi.org/10.1016/j.anucene.2010.10.019.

[16] M. Pusa and J. Leppänen, "Solving Linear Systems with Sparse Gaussian Elimination in the Chebyshev Rational Approximation Method," *Nuclear Science and Engineering*, **175**, *3*, 250 (2013); 10.13182/NSE12-52.

[17] M. Pusa, "Numerical methods for nuclear fuel burnup calculations: Dissertation," PhD Thesis, Aalto University, Finland (2013).

[18] R. B. Bird, W. E. Stewart, and E. N. Lightfoot, *Transport Phenomena*, John Wiley and Sons, Inc., New York (2006).

[19] N. Waterson and H. Deconinck, "Design principles for bounded higher-order convection schemes – a unified approach," *Journal of Computational Physics*, **224**, *1*, 182 (2007); https://doi.org/10.1016/j.jcp.2007.01.021., special Issue Dedicated to Professor Piet Wesseling on the occasion of his retirement from Delft University of Technology.

[20] H. K. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics The Finite Volume Method*, Pearson Prentice Hall (2007).

[21] P. K. Sweby, "High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws," *SIAM Journal on Numerical Analysis*, **21**, *5*, 995 (1984)URL http://www.jstor.org/stable/2156939.

[22] P. Khosla and S. Rubin, "A diagonally dominant second-order accurate implicit scheme," *Computers & Fluids*, **2**, *2*, 207 (1974); https://doi.org/10.1016/0045-7930(74)90014-0.

[23] A. Isotalo, "Computational Methods for Burnup Calculations with Monte Carlo Neutronics," PhD Thesis, Aalto University, Finland (2013).

[24] A. Bratsos and A. Khaliq, "An exponential time differencing method of lines for Burgersâ-Fisher and coupled Burgers equations," *Journal of Computational and Applied Mathematics*, **356**, 182 (2019); https://doi.org/10.1016/j.cam.2019.01.028.

[25] G. Beylkin, J. M. Keiser, and L. Vozovoi, "A New Class of Time Discretization Schemes for the Solution of Nonlinear PDEs," *Journal of Computational Physics*, **147**, *2*, 362 (1998); https://doi.org/10.1006/jcph.1998.6093.

[26] A. Bratsos and A. Q. Khaliq, "A conservative Exponential Time Differencing method for the nonlinear cubic Schrödinger equation," *International Journal of Computer Mathematics*, **94**, 1 (2015); 10.1080/00207160.2015.1101458.

[27] Q. Du and W. Zhu, "Analysis and Applications of the Exponential Time Differencing Schemes and Their Contour Integration Modifications," *BIT Numerical Mathematics*, **45**, 307 (2005); 10.1007/s10543-005-7141-8.

[28] N. J. Higham, "The Scaling and Squaring Method for the Matrix Exponential Revisited," *SIAM Journal on Matrix Analysis and Applications*, **26**, *4*, 1179 (2005); 10.1137/04061101X.

[29] A. Al-Mohy and N. Higham, "A New Scaling and Squaring Algorithm for the Matrix Exponential," *SIAM Journal on Matrix Analysis and Applications*, **31** (2009); 10.1137/09074721X.

[30] A. H. Al-Mohy and N. J. Higham, "Computing the Action of the Matrix Exponential, with an Application to Exponential Integrators," *SIAM Journal on Scientific Computing*, **33**, *2*, 488 (2011); 10.1137/100788860.

[31] L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer, "Talbot quadratures and rational approximations," *BIT Numerical Mathematics*, **46**, *3*, 653 (2006); 10.1007/s10543-006-0077-9.

[32] M. Pusa, "Accuracy considerations for Chebyshev rational approximation method (CRAM) in Burnup calculations," *Proceedings*, 973–984 (2013)International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, M&C 2013, M&C 2013 ; Conference date: 05-05-2013 Through 09-05-2013.

[33] Y. Saad, "Analysis of Some Krylov Subspace Approximations to the Matrix Exponential Operator," *SIAM Journal on Numerical Analysis*, **29**, *1*, 209 (1992)URL http://www.jstor.org/stable/2158085.

[34] E. Gallopoulos and Y. Saad, "On the parallel solution of parabolic equations," *ICS '89*, 17–28 (1989).

[35] K. Ott and R. Neuhold, *Introductory nuclear reactor dynamics*, American Nuclear Society (1985)URL https://books.google.com/books?id=3yVPAQAAIAAJ.

[36] R. Z. Taylor, "Libowski: A Numerical Framework for Solving Depletion and Mass Transport in Molten Salt Reactors," PhD Thesis, University of Tennessee (2021)URL https://trace.tennessee.edu/utk_graddiss/6662.

[37] M. Fratoni, D. Shen, G. Ilas, and J. Powers, "Molten Salt Reactor Experiment Benchmark Evaluation," Project 16-10240, University of California Berkeley and Oak Ridge National Laboratory (2020).