A Backward Euler Doubling Feasibility Study Based on Thorium Series Cascade

K. Huang,[1, 1] Y. Li,[1, 2] B. Ganapol[3]

[1]*Jiaotong University, Department of Nuclear Engineering, Xi'an, Shaanxi, 710049*
P.R. China
[1]*hwang@hotmail.com,* [2]*yunzhao@mail.xjtu.edu.cn*
[3]*University of Arizona, Department of Aerospace and Mechanical Engineering, Tucson, Arizona, 85721 USA*
ganapol@cowboy.ame.arizona.edu

INTRODUCTION

The solution to the Bateman equations of radioactive decay are central to burnup calculations for operational nuclear reactors. There are several excellent methods of solving the equations of radioactive decay, with the CRAM[1] algorithm as one of the best, at least for constant flux burnup. In this work, we attempt to investigate the feasibility of a relatively straight forward algorithm based simply on finite difference and doubling and apply the resulting benchmark to a recently developed decay chain formulation.

THEORY

We begin with the coupled first order ODEs

$$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{A}\boldsymbol{y}(t);\ \ \boldsymbol{y}(0) = \boldsymbol{y}_0 \cdot \qquad (1a)$$

of Bateman equations for *m* isotopes. The solution for the isotopic densities can be expressed analytically as

$$\boldsymbol{y}(t) = e^{\boldsymbol{A}t}\boldsymbol{y}_0 \qquad (1b)$$

in terms of the matrix exponential function. While it seems that such a simple expression should be easily evaluated, generally, this is not the case[2]. The evaluation of a matrix exponential function is one of the most challenging (and important) of all numerical methods[3]. If the matrix $\boldsymbol{A}$ is diagonalized, then, complex arithemetic may be required since the eigenvalues can be complex. In addition, the real parts may be overly large or small causing under/over flows. For large *m*, diagonalization may require extended precision and thus be quite time consuming. A Taylor series evaluation sometimes requires many terms and therefore is also time consuming. For these reasons, alternative numerical solutions to Eq(1a), other than matrix exponentials, are preferred. Here, we will derive and demonstrate arguably the simplest of all possible methods—the Backwards Euler Doubling Finite Difference (BEDFD).

Backward Euler Finite Differences (BEFD)

Assuming an interval [$t_{j\text{-}1}$, t_j] of width h_j with known initial condition (IC) $\boldsymbol{y}_{j-1}$, then from Eq(1b), we find

$$\boldsymbol{y}_j = \boldsymbol{R}\left(h_j\right)\boldsymbol{y}_{j-1}, \qquad (2)$$

where $\boldsymbol{R}$ the response matrix for interval h_j

$$\boldsymbol{R}\left(h_j\right) \equiv e^{\boldsymbol{A}h_j}.$$

$\boldsymbol{y}_{j-1}$ and $\boldsymbol{y}_j$ are on the left and right interval edges respectively. Let h_j be further partitioned into a set of 2^n (refined) sub-intervals each of width

$$h_{j,n} \equiv \frac{h_j}{2^n},$$

for a sequence *n* of integers, $n = 0,1,\ldots,N$. Thus, for each *n*, there are a total of 2^n sub-intervals. Over sub-interval *k*, within $h_{j,n}$ therefore,

$$\boldsymbol{y}_{j,k} = \boldsymbol{R}\left(h_{j,n}\right)\boldsymbol{y}_{j,k-1}, \qquad (3)$$

with $k = 1,2,\ldots,2^n$. $\boldsymbol{y}_{j,k-1}$ and $\boldsymbol{y}_{j,k}$ are on the left and right edges of sub-interval *k* respectively. At this point, we rewrite Eq(3) as

$$\boldsymbol{R}\left(h_{j,n}\right) = \left[e^{-\boldsymbol{A}h_{j,n}}\right]^{-1},$$

and approximate the matrix inverse exponential by the first two terms of its Taylor series

$$\boldsymbol{R}\left(h_{j,n}\right) \simeq \left[\boldsymbol{I} - h_{j,n}\boldsymbol{A}\right]^{-1}, \qquad (4)$$

which is the backward Euler finite difference (BEFD) approximation we apply in the following.

Doubling (BEDFD)

The concept of doubling with convergence acceleraton is stated in the following two steps:

a. Starting from the left edge $t_{j\text{-}1}$ with a converged IC, $\boldsymbol{y}_{j,k=0} = \boldsymbol{y}_{c,j-1}$ (to be defined below) for a fixed $n > 0$, double the response $\boldsymbol{R}_i\left(h_{j,n}\right)$ of the first interval k =1 to give the response for two such intervals, k =1 and 2 of width $2h_{j,n}$. Thus,

$$\begin{aligned}\boldsymbol{y}_{j,1} &= \boldsymbol{R}_i\left(h_{j,n}\right)\boldsymbol{y}_{j,0}\\ \boldsymbol{y}_{j,2} &= \boldsymbol{R}_i\left(h_{j,n}\right)\boldsymbol{y}_{j,1} = \boldsymbol{R}_i\left(h_{j,n}\right)\boldsymbol{R}_i\left(h_{j,n}\right)\boldsymbol{y}_{j,0}\end{aligned}$$

to give the combined response over two intervals

$$\boldsymbol{R}_i\left(2h_{j,n}\right) \equiv \boldsymbol{R}_i\left(h_{j,n}\right)^2.$$

Now doubling to four intervals gives

$$\boldsymbol{R}_i\left(2^2 h_{j,n}\right) \equiv \boldsymbol{R}_i\left(2h_{j,n}\right)\boldsymbol{R}_i\left(2h_{j,n}\right) = \boldsymbol{R}_i\left(h_{j,n}\right)^{2^2}.$$

Continued doubling until the original width h_j is covered gives the full width response for interval j of

$$\boldsymbol{R}_i\left(2^n h_{j,n}\right) \equiv \boldsymbol{R}_i\left(2^{n-1} h_{j,n}\right)\boldsymbol{R}_i\left(2^{n-1} h_{j,n}\right) = \boldsymbol{R}_i\left(h_{j,n}\right)^{2^n}. \tag{5a}$$

Note that this procedure requires n doublings over the entire interval h_j rather than 2^n applications of individual responses $\boldsymbol{R}_i\left(h_{j,n}\right)$, which can be a considerable savings when m is large in both time and accumulation of propagation error. The approximation to $\boldsymbol{y}_j$ starting the next interval $[t_{j\text{-}1}, t_j]$ is therefore

$$\boldsymbol{y}_j = \boldsymbol{R}_i\left(2^n h_{j,n}\right)\boldsymbol{y}_{c,j-1}. \tag{5b}$$

b. At each interval boundary, we have established a sequence of approximations to $\boldsymbol{y}_j$, called $y_j(n)$, to be accelerated to convergence by either the Wynn-epsilon or Richardson extrapolation[4]. The first to converge to the desired accuracy between the original and extrapolated sequences gives the converged $\boldsymbol{y}_{cj}$ to start the next interval. In this way, we minimize propagation error with a highly precise IC starting each time interval.

Implementation

Maximum efficiency is achieved when the edit intervals are uniform since the response need only be found once for the first k-sub-intervals and composited with the converged IC for the remaining intervals unless a higher n for convergence of $\boldsymbol{y}_j$ is necessary.

As already mentioned, the advantage of doubling over simple finite difference is now clear in that rather than cycle through 2^n sub-intervals for width h_j, only n doubled sub-intervals, where convergence acceleration gives the solution at the interval end point.

DEMONSTARTION 1: A STIFF ODE

The first demonstration comes from Eq(1.21) of Ref 5, where

$$\boldsymbol{A} \equiv \begin{bmatrix} -k_1 & 0 & k_2 \\ 0 & -k_4 & k_3 \\ k_1 & k_4 & -\left(k_2 + k_3\right) \end{bmatrix} \qquad \boldsymbol{y}_0 \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{6}$$

with

$$\begin{aligned}&k_1 = 8.4303270x10^{-10},\ k_3 = 2.4603642x10^{10}\\ &k_2 = 2.9002673x10^{11},\ k_4 = 8.7600580x10^{-6},\end{aligned}$$

is a simulation of proton transfer in an H-H bond. We consider this problem since, as shown in Figs. 1a,b, the third component (in Fig. 1b) is many orders of magnitudes smaller than the first two (Fig. 1a) and therefore could become completely lost if the default error option in MATLAB's *ode15s* is used[5]. The doubling results in Figs 1a,b are for a relative error of 10^{-10} without need of adjustment of the doubling implementation described above.

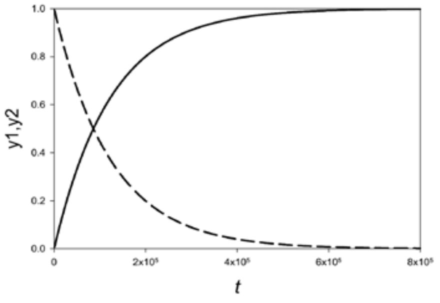

Fig. 1a. Components 1 and 2.

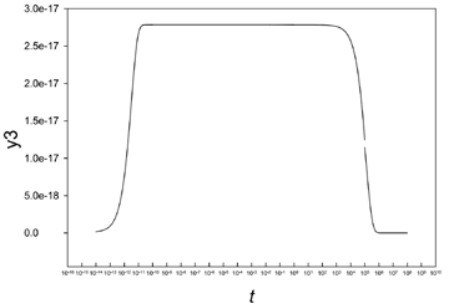

Fig. 1b. Component 3

By matrix diagonalization, the analytical solution to Eq(6) is

$$y(t) = Te^{\lambda t}T^{-1}y_0 \qquad (7)$$

providing a check on the doubling method.

In spite of stiff nature of this problem, the doubling implementation and the analytical solution in Double Precision (DP) arithmetic agree completely to the 10- digits quoted and nearly to machine DP as shown in Table 1a. For t near zero ~10^{-11}, the story is different as the first component of the analytical solution fails as shown in Table 1b and is confirmed by Quadruple Precision (QP) arithmetic.

Table 1a. Relative Error from Exact

t\RE	*y1*	*y2*	*y3*
6.00E+04	6.8942E-12	1.3031E-13	1.3366E-13
1.20E+05	6.9674E-12	2.6083E-13	2.6464E-13
1.80E+05	7.0337E-12	3.9113E-13	3.9582E-13
2.40E+05	7.0938E-12	5.2140E-13	5.2773E-13
3.00E+05	7.1492E-12	6.5240E-13	6.6078E-13

Table 1b. Comparison of doubling and exact solutions at short time.

Doubling

t	*y1*	*y2*	*y3*
2.00E-11	1.358830154E-16	1.000000000E+00	2.779087235E-17
4.00E-11	2.973363061E-16	1.000000000E+00	2.784228352E-17
6.00E-11	4.588369001E-16	1.000000000E+00	2.784237863E-17
8.00E-11	6.203375815E-16	1.000000000E+00	2.784237880E-17
1.00E-10	7.818382631E-16	1.000000000E+00	2.784237880E-17

Exact

t	*y1*	*y2*	*y3*
2.00E-11	1.110223025E-16	1.000000000E+00	2.779087235E-17
4.00E-11	3.330669074E-16	1.000000000E+00	2.784228352E-17
6.00E-11	4.440892099E-16	1.000000000E+00	2.784237863E-17
8.00E-11	6.661338148E-16	1.000000000E+00	2.784237880E-17
1.00E-10	7.771561172E-16	1.000000000E+00	2.784237880E-17

From these first results, it is most likely safe to assume that the BEDFD has potential in treating constant coefficient ODEs, in particular those associated with nuclear reactor fuel burnup.

DEMONSTARTION 2: THORIUM SERIES CASCADE

In this demonstration, we apply BEDFD to the Thorium Series decay chain of 12 elements shown in Fig. 2.

The Bateman equations for the Thorium Series for j=1,…,12, are

$$\frac{dN_j(t)}{dt} = \left(1-\delta_{j,11}\right)\left(1-(1-\beta_1)\delta_{j,10}\right)\lambda_{j-1}N_{j-1}(t) + \\ +\delta_{j,11}\beta_2\lambda_{j-2}N_{j-2}(t) - \lambda_j N_j(t) + \delta_{j,12}\lambda_{j-2}N_{j-2}(t), \qquad (8)$$

with $\lambda_0 = \lambda_{12} \equiv 0$ and $N_j(t)$ for $j < 0$ and

$$A \equiv \begin{bmatrix} -\lambda_1 & 0 & \dots & & & & & & & & \dots & 0 \\ \lambda_1 & -\lambda_2 & 0 & \dots & & & & & & & & \dots \\ 0 & \lambda_2 & -\lambda_3 & 0 & \dots & & & & & & & \\ \dots & 0 & \lambda_3 & -\lambda_4 & 0 & \dots & & & & & & \\ & \dots & 0 & \lambda_4 & -\lambda_5 & 0 & \dots & & & & & \\ & & \dots & 0 & \lambda_5 & -\lambda_6 & 0 & \dots & & & & \\ & & & \dots & 0 & \lambda_6 & -\lambda_7 & 0 & \dots & & & \\ & & & & \dots & 0 & \lambda_7 & -\lambda_8 & 0 & \dots & & \\ & & & & & \dots & 0 & \lambda_8 & -\lambda_9 & 0 & \dots & \\ & & & & & & \dots & 0 & \beta_1\lambda_9 & -\lambda_{10} & 0 & \dots \\ \dots & & & & & & & 0 & \beta_2\lambda_9 & 0 & -\lambda_{11} & 0 \\ 0 & \dots & & & & & & \dots & 0 & \lambda_{10} & \lambda_{11} & 0 \end{bmatrix}. \qquad (9a)$$

Initial conditions are

$$N(0) = N_0 \equiv \left\{N_0\delta_{j,1};\, j = 1,\dots,12\right\}\cdot \qquad (9b)$$

Therefore, we must evaluate

$$\frac{dN(t)}{dt} = AN(t);\;\; N(0) = N_0, \qquad (9c)$$

where

$$N(t) \equiv \left[N_1(t) \quad N_2(t) \quad \dots \quad N_{12}(t)\right]^T \qquad (9d)$$

and assuming N_0 to be one mole ($6.023x10^{23}$) of Thorium nuclei.

Table 2a gives nuclide half-lives in units of per day and branching ratios. The results are assumed precise as quoted to avoid unintended error. We now apply BEDFD as outlined above to this relatively stiff problem.

Table 2a Decay Constants for Thorium Series (per day)

m	*λ(da^{-1})*
1	1.351625d-13
2	3.302667d-04
3	2.661685d+00
4	9.934261d-04
5	1.908497d-01
6	1.077121d+03
7	4.130201d+05
8	1.563490d+00
9	1.648442d+01
10	2.002940d+11
11	269348d+02

Branching Ratios

β_1=0.6406q0

β_2=0.3594q0

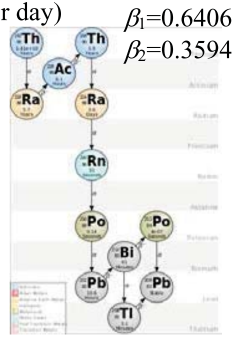

Fig. 2 Thorium Series

Figures 3a,b show the evolution of the 12 nuclides over 10000 years and 3.65 days. While not shown, the sum over all nuclides is exactly N_0 to at least 9- places for all times. All seems in order as ^{232}Th barely decays as ^{208}Pb builds from zero over a long period. There are also some fast decays that make several nuclide densities nearly asymptotically equal.

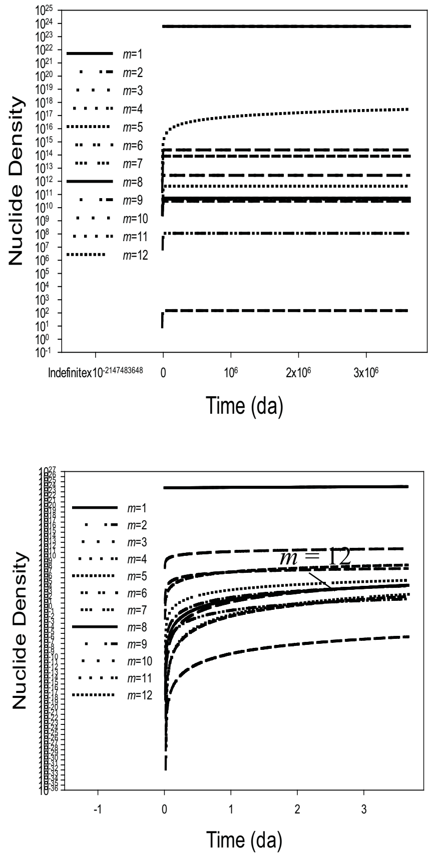

Fig. 3a,b. Evolution of the Thorium Series nuclides to 10000 *yrs* and to 3.65 *days*.

A word on the use of extended precision is in order. Generally, methods developers prefer to use DP to QP, which provides a convergence check of DP. This was the application presented here. However, QP has the advantage when extreme accuracy is desired to achieve the solution in fewer doublings. The reduction in doublings, can therefore give a time advantage over the increased computational cost of QP, which therefore should be considered a viable option.

Tables 3 provide a Thorium Series benchmark comparison at 1 year. The DP values in Table 3a should be accurate to better than one digit in the last place and agree with the analytical solution to all digits. One of the primary motivations for the benchmark was to compare to the newly developed Transmutation Trajectory Analysis (TTA) method[6], which achieves high accuracy by defining all the depletion decay chains and analytically solving each. All digits are in agreement. The next task will be to develop a benchmark for hundreds of isotopes to see if the high precision of the doubling method maintains.

Table 3a. Benchmark for Thorium Series at 1 year.

t\m	**1**	**2**	**3**	**4**
3.65E+01	6.023000000E+23	2.953567638E+12	3.627339589E+08	1.726783940E+10
7.30E+01	6.023000000E+23	5.871744544E+12	7.248715743E+08	6.867693006E+10
1.10E+02	6.023000000E+23	8.754954782E+12	1.082669924E+09	1.525986406E+11
1.46E+02	6.023000000E+23	1.160361734E+13	1.436181002E+09	2.674636650E+11
1.82E+02	6.023000000E+23	1.441814617E+13	1.785456180E+09	4.117635120E+11
2.19E+02	6.023000000E+23	1.719895028E+13	2.130546215E+09	5.840482804E+11
2.56E+02	6.023000000E+23	1.994643378E+13	2.471501255E+09	7.829245141E+11
2.92E+02	6.023000000E+23	2.266099592E+13	2.808370846E+09	1.007053135E+12
3.28E+02	6.023000000E+23	2.534303118E+13	3.141203942E+09	1.255147447E+12
3.65E+02	6.023000000E+23	2.799292931E+13	3.470048909E+09	1.525971220E+12

t\m	**5**	**6**	**7**	**8**
3.65E+01	6.770295695E+07	1.199525307E+04	3.128258655E+01	7.939993206E+06
7.30E+01	3.102328978E+08	5.496713713E+04	1.433495695E+02	3.717037959E+07
1.10E+02	7.231989444E+08	1.281377827E+05	3.341723329E+02	8.722319215E+07
1.46E+02	1.298389797E+09	2.300522692E+05	5.999565686E+02	1.570954782E+08
1.82E+02	2.027908499E+09	3.593113728E+05	9.370532208E+02	2.458226147E+08
2.19E+02	2.904164707E+09	5.145701970E+05	1.341954915E+03	3.524774301E+08
2.56E+02	3.919863474E+09	6.945361853E+05	1.811290774E+03	4.761688346E+08
2.92E+02	5.067994426E+09	8.979672033E+05	2.341821419E+03	6.160404980E+08
3.28E+02	6.341821336E+09	1.123669691E+06	2.930434147E+03	7.712695760E+08
3.65E+02	7.734872062E+09	1.370496881E+06	3.574138287E+03	9.410654817E+08

t\m	**9**	**10**	**11**	**12**
3.65E+01	7.501737681E+05	3.955073326E-05	1.359150694E+04	1.310116157E+08
7.30E+01	3.519211116E+06	1.855401855E-04	6.376713399E+04	1.313896772E+09
1.10E+02	8.263344020E+06	4.356608149E-04	1.497343094E+05	4.761869212E+09
1.46E+02	1.488744027E+07	7.848970516E-04	2.697691038E+05	1.163451205E+10
1.82E+02	2.330000268E+07	1.228424973E-03	4.222134591E+05	2.303525827E+10
2.19E+02	3.341308648E+07	1.761607946E-03	6.054736919E+05	4.001356400E+10
2.56E+02	4.514216931E+07	2.379989774E-03	8.180181372E+05	6.356700654E+10
2.92E+02	5.840602584E+07	3.079288088E-03	1.058374878E+06	9.464330570E+10
3.28E+02	7.312660696E+07	3.855387976E-03	1.325129554E+06	1.341422709E+11
3.65E+02	8.922892327E+07	4.704335839E-03	1.616923250E+06	1.829176771E+11

CONCLUSIONS

A simple Backward Euler Doubling Finite Difference (BEDFD) method for solving first order ordinary differential equations is developed with enhanced precision through the convergence acceleration. Numerical results of two analytically solvable test problems clearly demonstrated that this newly proposed method is robust and accurate. Therefore, this method could be promising in solving larger depletion problems—our next effort in our comparisons with TTA.

REFERENCES

1. M. Pusa and J. Leppänen, “Computing the matrix exponential in burnup calculations,” *Nucl. Sci. Eng.*, **164**, 2, 140–150 (2010).

2. N. J. Higham, *Functions of Matrices*, SIAM, (2008).

3. C. Moler, C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM review,* **45** (1) 3{49. doi: 10.1137/ S00361445024180, (2003).

4. A. Sidi, *Practical Extrapolations Methods*, Cambridge University Press, Cambridge, (2003).

5. L.F. Shampine, et. al., *Solving ODEs with MATLAB*, Cambridge University Press, (2003).

6. Kai Huang, submitted, (2015).