Annals of Nuclear Energy 87 (2016) 637–647

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

Improvements to the Transmutation Trajectory Analysis of depletion evaluation

Kai Huang[a], Hongchun Wu[a], Liangzhi Cao[a], Yunzhao Li[a,*], Wei Shen[b]

[a] *School of Nuclear Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China*
[b] *Canadian Nuclear Safety Commission Headquarters, 280 Slater Street, Ottawa, ON K1P 5S9, Canada*

ARTICLE INFO

Article history:
Received 1 February 2015
Received in revised form 13 October 2015
Accepted 14 October 2015
Available online 9 November 2015

Keywords:
Depletion evaluation
Transmutation Trajectory Analysis
Bateman equation

ABSTRACT

To simulate the nuclide evolution process in a nuclear reactor core, the Transmutation Trajectory Analysis (TTA) method solves the depletion equations by decomposing the depletion system into a number of linear chains and then solving each one analytically. In this paper, two improvements are proposed for TTA to obtain better efficiency. Firstly, the pseudo node evaluation for linear chain cutoff check has been removed. Instead, a time-averaged nuclide number density is employed as the chain termination criteria, which in theory can improve the computational efficiency by a factor of two. Secondly, a new recursive formula has been derived to replace the legacy direct solution formula for solving the linear chains. Numerical tests have been carried out based on a typical PWR fuel cycle to demonstrate that these improvements enable the TTA method to solve decay problems efficiently and accurately.

1. Introduction

Nuclear energy release is accompanied by nuclides depletion process. The variety of nuclides vary in their toxicities, decay properties and microscopic cross sections. Therefore, predicting the time evolution of nuclide inventories is of vital importance in nuclear applications. Actually, the depletion process is coupled together with both neutronics and thermal–hydraulics processes. Within a small enough time step, however, these three processes can be treated independently. Although the depletion process is quite complicated, there are governing equations called depletion equations describing the variation rate of nuclide number density which equal the difference of rate of production and consumption. These equations can be written as a set of first order ordinary differential equations assuming constant microscopic reaction rates (Stamm'ler and Abbate, 1983) for each depletion region during small time steps. The coefficient matrix of the depletion equations, or depletion matrix, describes the nuclide transformation relations. For example, each of the diagonal entries stands for the disappearance of the corresponding nuclide, while each of the off-diagonal entries represents the contribution between two different nuclides.

There are two major types of methods which can be used for solving the depletion equations with constant coefficients: matrix exponential methods and linear chain methods. In general, computation of matrix exponential in itself constitutes a rich field of study (Moler and Van Loan, 2003). Several methods have been introduced to solving depletion equations, such as the Taylor expansion and truncation method with secular equilibrium assumed for short-lived nuclides (Croff, 1980; Hermann and Westfall, 1998), and Chebyshev rational approximation method (CRAM) (Pusa and Leppänen, 2010; Pusa, 2011). Taylor expansion and truncation method is based on the definition of the matrix exponential in Taylor series form. CRAM is related to the rational approximation of exponential function (Trefethen et al., 2006). Other than these matrix approaches, linear chain methods offer a direct solution method. With little computational overhead, this approach explicitly models nuclide chains and enumerates all important ones. A more recent general analytical solution by Cetnar, referred to as Transmutation Trajectory Analysis method (TTA) (Cetnar, 2006) has remedied the identical vanishing coefficients issue. When this solution is used, the only approximation left is the termination of linear chains of low importance. The term linear chain here refers to a simple depletion process in which each nuclide has at most only one predecessor and at most only one daughter nuclide. Each of these linear chains is governed by analytically solvable bi-diagonal depletion equations, or bi-diagonal Bateman equations. The solution of the original depletion equations can be obtained as a superposition of those analytical solutions.

The depletion problems could be classified into two different types, namely the decay problems and the burnup problems (Isotalo, 2013), based on the absence or presence of neutron flux.

* Corresponding author. Tel.: +86 029 82663285.
E-mail address: yunzhao@mail.xjtu.edu.cn (Y. Li).

http://dx.doi.org/10.1016/j.anucene.2015.10.013
0306-4549/© 2015 Elsevier Ltd. All rights reserved.

For a pure decay system, the disassembly of the directed graph representing the depletion matrix called linearization would be exact since a finite number of linear chains with finite length will be equal to the original depletion equations according to the superposition principal and the absence of closed transformation cycles. The latter condition is insured, since a nuclide could not go through a series of energy emitting decay processes and return to itself precisely with an unchanged mass. However, for a burnup system with neutron-induced nuclear reactions involved, there might be closed transformation cycles, thus a finite or even infinite number of linear chains with infinite length are necessary for maintaining the equivalence between the original depletion equations and the series of linear chains. Practically, not all of these linear chains are important enough to be considered however, because the number of nuclides transferred through the linear chain decreases rapidly along the chain. Consequently, a cutoff check is carried out to determine the effective end of each linear chain growing process. Considering that the determination of linear chain solutions contributes most to the computational cost, the efficiency could be improved by reducing the number and the length of calculated linear chains or finding more efficient analytic formulas to resolve the nuclide number densities for each of these linear chains. Such an efficiency improvement would accumulate considerable saving on computational effort in applications that demand large numbers of depletion calculations.

In this paper, two improvements concerning efficiency of TTA are proposed. Firstly, a time-averaged nuclide number density is employed to simplify the linear chain cutoff check, which is usually done by calculating the number density of a stable node appended at the end of the chain. In this paper, this artificial stable node is referred to as a pseudo node. Secondly, in terms of the analytic solution, a recursive formula, which is mathematically equivalent with the legacy direct formula, is derived and implemented to reduce computational effort spent on solving linear chains. Numerical tests have been carried out based on a typical PWR fuel cycle. It has been demonstrated that these improvements could make the TTA method run faster by a factor of about 8 while maintaining the same computational accuracy. While it has been shown that the burnup problems could be solved elegantly by CRAM (Pusa and Leppänen, 2010; Pusa, 2011; Isotalo and Aarnio, 2011), the TTA method has better performance for solving decay problems, and could be viewed as a complement with respect to CRAM.

The paper is organized as the following. Section 2 derives the theory and formulas of the two improvements in detail. Numerical results and discussions of three selected test cases based on a PWR fuel cycle are given in Section 3. Finally, conclusions are drawn in Section 4.

2. Theoretical model

The TTA method finds the final solution by summing up contributions from all linear chains. The basic building element of linear chain is called node, and each root node corresponds to a nuclide with non-zero initial number density. The level of a node is defined as its distance from the root node. These concepts are illustrated in Fig. 1, in which only nuclide *a* is assigned a non-zero initial number density for the sake of simplicity. The linear chain searching process is in a form of three levels of iterations. The outermost iteration selects the nuclides that have non-zero initial nuclide number densities as root nodes. The second level iteration is composed of two innermost iterations: one for growing the linear chain along the transfer relationships until being terminated; and the other for identifying the restart node, which is defined as the node that has unexplored successors and is closest to the last node. Between the two innermost iterations, the nodes after restart node of the linear chain are taken into account to the contributions of the solution, since they belong to the newly explored transmutation path. Termination of linear chain in the growing process happens when the last node has no successors or the importance of the linear chain falls below a certain value (cutoff criterion). The second level iteration ends when the restart node cannot be found. The pseudo code of the linear chain searching process is provided in Appendix A under Algorithm 1.

The governing equations for each linear chain are the bi-diagonal Bateman equations:

$$\frac{dN_1(t)}{dt} = -\lambda_1 N_1(t) \tag{1a}$$

$$\frac{dN_{i+1}(t)}{dt} = \lambda_{i+1,i}N_i(t) - \lambda_{i+1}N_{i+1}(t) \quad (1 \leqslant i < n) \tag{1b}$$

where $N_i(t)$ refers to the number density (cm^{-3}) of the *i*th node, λ_i is the vanishing coefficient of *i*th node nuclide (s^{-1}), while $\lambda_{i+1,i}$ stands for the transfer coefficient from the *i*th node nuclide to $(i+1)$th node nuclide, defined as:

$$\begin{aligned} \lambda_i &= \lambda_i^{\text{decay}} + \sigma_{a,i}\phi \\ \lambda_{i+1,i} &= b_{i+1,i}\lambda_i^{\text{decay}} + \sigma_{i+1,i}\phi \end{aligned} \tag{2}$$

λ_i^{decay} is the decay constant, $\sigma_{a,i}$ is the single group microscopic absorption cross section, $b_{i+1,i}$ and $\sigma_{i+1,i}$ are the branching ratio and the single group microscopic cross section that produces $(i+1)$th node nuclide respectively, ϕ is the neutron flux.

2.1. Improvement on the cutoff check

To avoid linear chains of infinite length, or unnecessary linear chain nodes, a criterion has to be employed to stop the searching process for each linear chain. The essential measurement is the linear chain passage (Cetnar, 2006; Isotalo and Aarnio, 2011), which is defined as the number density that goes through the last transfer relation:

$$P_n = \int_0^{t_f} \lambda_{n+1,n} \cdot N_n(t)dt \tag{3}$$

where the linear chain is cut between the *n*th and the $(n+1)$th nodes, and $\lambda_{n+1,n}$ is the corresponding transfer coefficient. It offers an estimate of the remaining number densities that will be neglected. Thus, the linear chain should be terminated once P_n falls below the cutoff criterion determined by:

$$\varepsilon_{\text{cutoff}} = \text{cutoff} \cdot N_{\text{total}}(0) \tag{4}$$

where $N_{\text{total}}(0)$ represents the initial total nuclide number density.

The independent pseudo node approach, which is implemented initially, appends a pseudo node with $\lambda_{\text{pseudo}} = 0.0$ and $\lambda_{\text{pseudo},n} = 1.0$ after the *n*th node of the linear chain. Then, it can be shown that P_n equals $\lambda_{n+1,n}N_{\text{pseudo}}(t_f)$. Since most nodes are not stable, and require pseudo node calculations, the total computing effort is nearly doubled.

An alternative approach is to determine linear chain passage from a time-averaged nuclide number density:

$$\overline{N}_n(t) = \frac{1}{t}\int_0^t N_n(\tau)d\tau \tag{5}$$

As will be discussed latter, $N_n(t)$ could be expanded as the sum of terms taking the form $t^k e^{-\lambda_j t}$. The integrals of these terms could be pre-calculated and stored for all possible combinations of λ_j and *k* up to a very limited number of values (3 is enough in most cases). Compared to the independent pseudo node approach, the computation solving the pseudo node is saved, and additional integration is required. The time saving effect dominates in practice, because the integration is merely based on weighted summation of pre-calculated values, therefore the computational time is approximately

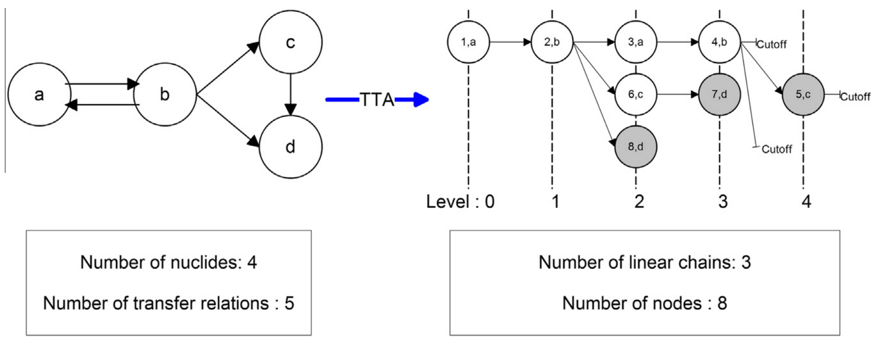

Fig. 1. Illustration of linear chains and nodes.

halved. Furthermore, the time-averaged nuclide densities can also be used occasionally to calculate the number of particular type reactions occurred during a time step (Hermann and Westfall, 1998).

2.2. Direct formulas of the analytic solution

To solve the bi-diagonal Bateman equations analytically and efficiently, there are several proposed analytical formulas (Cetnar, 2006; Bateman, 1910; Rubinson, 1949; Hamawi, 1971; Miles, 1981; Dreher, 2013). Mathematically, there exists a unique analytical solution, but how to present it or in which form it is implemented into a code varies. Each different form affects computational efficiency. Traditionally, a direct formula is usually employed. Due to the fact that every node except the root node is rooted on a predecessor node, a recursive formula that makes use of stored results is likely to be the most efficient. But unlike any previously proposed recursive formulas (Hamawi, 1971; Miles, 1981), the one developed here only depends on one sub-chain (the original chain less the last node) solution.

With the following conditions added to Eq. (1):

$$\lambda_{i+1,i} = \lambda_i \quad (i \geqslant 1) \tag{6a}$$

$$N_i(0) = 0 \quad (i \geqslant 2) \tag{6b}$$

Bateman first derived a direct formula of the solution by means of Laplace transformation (Bateman, 1910). The Bateman formula can be presented as following, provided that the first i vanishing coefficients are distinct from each other:

$$\begin{aligned} N_i(t) &= \frac{N_1(0)}{\lambda_i}\sum_{j=1}^{i}\lambda_j\alpha_{i,j}e^{-\lambda_j t} \\ \alpha_{i,j} &= \prod_{\substack{k=1\\k\neq j}}^{i}\frac{\lambda_k}{\lambda_k-\lambda_j} \end{aligned} \tag{7}$$

For arbitrarily distributed vanishing coefficients, Cetnar obtained a direct formula of the solution by eliminating the infinities in the Bateman formula though limit operations (Cetnar, 2006):

$$\begin{aligned} N_i(t) &= \frac{N_1(0)}{\lambda_i}\sum_{j=1}^{a_i}\tilde{\lambda}_j\alpha_{i,j}e^{-\tilde{\lambda}_j t}\cdot\sum_{k=0}^{b_{i,j}}\frac{(\tilde{\lambda}_j t)^k}{k!}\cdot\Omega_{i,j,b_{i,j}-k} \\ \alpha_{i,j} &= \prod_{\substack{p=1\\p\neq j}}^{a_i}\left(\frac{\tilde{\lambda}_p}{\tilde{\lambda}_p-\tilde{\lambda}_j}\right)^{m_{i,p}} \\ \Omega_{i,j,k} &= \sum_{h_1=0}^{k}\cdots\sum_{h_{j-1}=0}^{k}\sum_{h_{j+1}=0}^{k}\cdots\sum_{h_{a_i}=0}^{k} \\ &\quad\times\prod_{\substack{p=1\\p\neq j}}^{a_i}\binom{h_p+b_{i,p}}{b_{i,p}}\left(\frac{\tilde{\lambda}_j}{\tilde{\lambda}_j-\tilde{\lambda}_p}\right)^{h_p}\delta\left(k,\sum_{\substack{p=1\\p\neq j}}^{a_i}h_p\right) \end{aligned} \tag{8}$$

where the first i vanishing coefficients have only a_i distinct values of $\tilde{\lambda}_j$ with each of them having multiplicity of $m_{i,j}$, and $b_{i,j} = m_{i,j} - 1$. These distinct vanishing coefficients are arranged in ascending order according to their position of their first occurrence.

2.3. Recursive formula of the analytic solution

According to the theory of differential equations (Leonard, 1996), the solution of the ith node takes the following form:

$$N_i(t) = \sum_{j=1}^{a_i}\sum_{k=0}^{b_{i,j}}\gamma_{i,j,k}t^k e^{-\tilde{\lambda}_j t} \tag{9}$$

For the first node described by Eq. (1a), the associated variables are as following:

$$\begin{aligned} a_1 &= 1 \quad b_{1,1} = 0 \\ \tilde{\lambda}_1 &= \lambda_1 \quad \gamma_{1,1,0} = N_1(0) \end{aligned} \tag{10}$$

For the other nodes described by Eq. (1b), substituting the solution form Eq. (9) yields the following equation:

$$\begin{aligned} &\sum_{j=1}^{a_{i+1}}\sum_{k=0}^{b_{i+1,j}}\left(kt^{k-1}-\tilde{\lambda}_j t^k\right)\gamma_{i+1,j,k}e^{-\tilde{\lambda}_j t} \\ &= \lambda_{i+1,i}\sum_{j=1}^{a_i}\sum_{k=0}^{b_{i,j}}\gamma_{i,j,k}t^k e^{-\tilde{\lambda}_j t} - \lambda_{i+1}\sum_{j=1}^{a_{i+1}}\sum_{k=0}^{b_{i+1,j}}\gamma_{i+1,j,k}t^k e^{-\tilde{\lambda}_j t} \end{aligned} \tag{11}$$

Since terms $t^k e^{-\tilde{\lambda}_j t}$ with different k and $\tilde{\lambda}_j$ values are linearly independent. The above equation could be decomposed into a series of linear algebraic equations giving out the relations between coefficients $\gamma_{i+1,*,*}$ and $\gamma_{i,*,*}$ (the asterisk sign indicates arbitrary values).

For those $j \in \{1,2,\ldots,a_i\}$ that satisfy $\tilde{\lambda}_j \neq \lambda_{i+1}$, the identity $b_{i+1,j} = b_{i,j}$ holds, and the following relations can be obtained from Eq. (11):

$$\begin{aligned} &(\lambda_{i+1}-\tilde{\lambda}_j)\gamma_{i+1,j,b_{i+1,j}} = \lambda_{i+1,i}\gamma_{i,j,b_{i,j}} \\ &(\lambda_{i+1}-\tilde{\lambda}_j)\gamma_{i+1,j,k} + (k+1)\gamma_{i+1,j,k+1} = \lambda_{i+1,i}\gamma_{i,j,k} \quad (0 \leqslant k < b_{i+1,j}) \end{aligned} \tag{12}$$

If there exists $\hat{j} \in \{1,2,\ldots,a_i\}$ satisfying $\tilde{\lambda}_{\hat{j}} = \lambda_{i+1}$, then $a_{i+1} = a_i$ and $b_{i+1,\hat{j}} = b_{i,\hat{j}} + 1$ are valid. The relations below are then decomposed from Eq. (11):

$$k\gamma_{i+1,\hat{j},k} = \lambda_{i+1,i}\gamma_{i,\hat{j},k-1} \quad (0 < k \leqslant b_{i+1,\hat{j}}) \tag{13}$$

Setting $t = 0$ yields $\sum_{j=1}^{a_{i+1}}\gamma_{i+1,j,0} = N_{i+1}(0)$. The last free variable $\gamma_{i+1,\hat{j},0}$ is settled eventually:

$$\gamma_{i+1,\hat{j},0} = N_{i+1}(0) - \sum_{\substack{j=1 \\ j \neq \hat{j}}}^{a_{i+1}} \gamma_{i+1,j,0} \tag{14}$$

If λ_{i+1} is distinct from all previously encountered vanishing coefficients, the following identities hold true: (1) $a_{i+1} = a_i + 1$, (2) $b_{i+1,a_{i+1}} = 0$, (3) $\widetilde{\lambda}_{a_{i+1}} = \lambda_{i+1}$. According to initial number density condition of $(i+1)$th node, the following relation must be satisfied:

$$\gamma_{i+1,a_{i+1},0} = N_{i+1}(0) - \sum_{j=1}^{a_i} \gamma_{i+1,j,0} \tag{15}$$

Consequently, for all j where $\widetilde{\lambda}_j \neq \lambda_{i+1}$:

$$\begin{aligned} & b_{i+1,j} = b_{i,j} \\ & \gamma_{i+1,j,b_{i+1,j}} = \frac{\lambda_{i+1,i}\gamma_{i,j,b_{i,j}}}{\lambda_{i+1} - \widetilde{\lambda}_j} \\ & \gamma_{i+1,j,k} = \frac{\lambda_{i+1,i}\gamma_{i,j,k} - (k+1)\gamma_{i+1,j,k+1}}{\lambda_{i+1} - \widetilde{\lambda}_j} \quad (0 \leqslant k < b_{i+1,j}) \end{aligned} \tag{16}$$

If λ_{i+1} coincides with one of the previously encountered vanishing coefficients, that is $\exists \hat{j} \in \{1, 2, \ldots, a_i\}, \widetilde{\lambda}_{\hat{j}} = \lambda_{i+1}$, then:

$$\begin{aligned} & a_{i+1} = a_i \quad b_{i+1,\hat{j}} = b_{i,\hat{j}} + 1 \\ & \gamma_{i+1,\hat{j},k} = \frac{\lambda_{i+1,i}\gamma_{i,\hat{j},k-1}}{k} \quad (0 < k \leqslant b_{i+1,\hat{j}}) \\ & \gamma_{i+1,\hat{j},0} = N_{i+1}(0) - \sum_{\substack{j=1 \\ j \neq \hat{j}}}^{a_{i+1}} \gamma_{i+1,j,0} \end{aligned} \tag{17}$$

Otherwise λ_{i+1} is a new vanishing coefficient, which means $\forall \hat{j} \in \{1, 2, \ldots, a_i\}, \widetilde{\lambda}_{\hat{j}} \neq \lambda_{i+1}$, then:

$$\begin{aligned} & a_{i+1} = a_i + 1 \quad b_{i+1,a_{i+1}} = 0 \\ & \widetilde{\lambda}_{a_{i+1}} = \lambda_{i+1} \quad \gamma_{i+1,a_{i+1},0} = N_{i+1}(0) - \sum_{j=1}^{a_i} \gamma_{i+1,j,0} \end{aligned} \tag{18}$$

The recursive formula is composed of Eq. (10) and Eqs. (16)–(18). It does not require the conditions of Eq. (6) to be satisfied, and could be even generalized to solve triangular Bateman equations in a once-through manner as presented in Appendix B.

2.4. Equivalence proof between the direct and recursive formulas

The analytic solution to bi-diagonal Bateman equations for each linear chain can be constructed from either the direct formula in Section 2.2, or recursive formula in Section 2.3. In order to prove the equivalence between them for a linear chain that satisfies the conditions of Eq. (6), it is verified that the two formulas produce identical solutions for the first node, and it is shown that solutions to the ith and $(i+1)$th nodes obtained by direct formula comply with the recursive formula. For the sake of notational simplicity, $\Lambda_i, \hat{\alpha}_{i,j}, H_{i,j,k}$ (defined as a Cartesian product of sets), $P_{i,j,k}(\boldsymbol{h})$ and $\hat{\Omega}_{i,j,k}$ are defined as:

$$\begin{aligned} & \Lambda_i = \prod_{j=1}^{i} \lambda_j = \prod_{j=1}^{a_i} \widetilde{\lambda}_j^{b_{i,j}+1} \\ & \hat{\alpha}_{i,j} = \prod_{\substack{p=1 \\ p \neq j}}^{a_i} \left(\frac{1}{\widetilde{\lambda}_p - \widetilde{\lambda}_j} \right)^{b_{i,p}+1} \\ & H_{i,j,k} = \prod_{p=1}^{j-1} \{0, 1, \ldots, k\} \times \{0\} \times \prod_{p=j+1}^{a_i} \{0, 1, \ldots, k\} \\ & P_{i,j,k}(\boldsymbol{h}) = \prod_{\substack{p=1 \\ p \neq j}}^{a_i} \binom{h_p + b_{i,p}}{b_{i,p}} \left(\frac{1}{\widetilde{\lambda}_j - \widetilde{\lambda}_p} \right)^{h_p} \delta\left(k, \sum_{p=1}^{a_i} h_p \right) \\ & \hat{\Omega}_{i,j,k} = \sum_{\boldsymbol{h} \in H_{i,j,k}} P_{i,j,k}(\boldsymbol{h}) \end{aligned} \tag{19}$$

Let $\Gamma_{i,j,k}$ denotes the coefficient associated with term $t^k e^{-\widetilde{\lambda}_j t}$ in the solution to ith node produced by Cetnar formula. Then $\Gamma_{i,j,k}$ could be expressed as:

$$\Gamma_{i,j,k} = \frac{N_1(0)}{k!\lambda_i} \Lambda_i \hat{\alpha}_{i,j} \hat{\Omega}_{i,j,b_{i,j}-k} \tag{20}$$

First, it is trivial to verify that $\Gamma_{1,1,0} = \gamma_{1,1,0} = N_1(0)$, which means that the two formulas produce identical solutions for the first node.

The following four identities are relevant to the inductive part of proof:

(1) For $j \in \{1, 2, \ldots, a_i\}$, it is obvious that the following equation holds:

$$\hat{\Omega}_{i,j,0} = 1 \tag{21}$$

(2) For $\hat{j} \in \{1, 2, \ldots, a_i\}$ satisfying $\widetilde{\lambda}_{\hat{j}} = \lambda_{i+1}$, it is also obvious that the following equation holds:

$$\hat{\Omega}_{i+1,\hat{j},k} = \hat{\Omega}_{i,\hat{j},k} \tag{22}$$

(3) For $i \geqslant 2$, the initial condition in Eq. (6b) leads to the following:

$$\sum_{j=1}^{a_i} \hat{\alpha}_{i,j} \hat{\Omega}_{i,j,b_{i,j}} = 0 \tag{23}$$

(4) For $j \in \{1, 2, \ldots, a_i\}$ such that $\widetilde{\lambda}_j \neq \lambda_{i+1}$:

$$\hat{\Omega}_{i+1,j,k} = \hat{\Omega}_{i,j,k} - \frac{\hat{\Omega}_{i+1,j,k-1}}{\lambda_{i+1} - \widetilde{\lambda}_j} \tag{24}$$

Eq. (24) is verified under two different conditions:

If $\exists \hat{j} \in \{1, 2, \ldots, a_i\}, \widetilde{\lambda}_{\hat{j}} = \lambda_{i+1}$, then $a_{i+1} = a_i, b_{i+1,\hat{j}} = b_{i,\hat{j}} + 1$. For those j that $j \neq \hat{j}$:

$$\begin{aligned} \hat{\Omega}_{i+1,j,k} - \hat{\Omega}_{i,j,k} &= \sum_{\substack{\boldsymbol{h} \in H_{i+1,j,k} \\ h_{\hat{j}} \neq 0}} P_{i+1,j,k}(\boldsymbol{h}) + \sum_{\substack{\boldsymbol{h} \in H_{i+1,j,k} \\ h_{\hat{j}} = 0}} P_{i+1,j,k}(\boldsymbol{h}) \\ & - \left(\sum_{\substack{\boldsymbol{h} \in H_{i,j,k} \\ h_{\hat{j}} = 0}} P_{i,j,k}(\boldsymbol{h}) + \sum_{\substack{\boldsymbol{h} \in H_{i,j,k} \\ h_{\hat{j}} \neq 0}} P_{i,j,k}(\boldsymbol{h}) \right) \\ &= \sum_{\substack{\boldsymbol{h} \in H_{i+1,j,k} \\ h_{\hat{j}} \neq 0}} \left[\binom{h_{\hat{j}} + b_{i+1,\hat{j}}}{b_{i+1,\hat{j}}} - \binom{h_{\hat{j}} + b_{i,\hat{j}}}{b_{i,\hat{j}}} \right] \\ & \times \left(\frac{1}{\widetilde{\lambda}_j - \widetilde{\lambda}_{\hat{j}}} \right)^{h_{\hat{j}}} P_{i,j,k-h_{\hat{j}}}(\boldsymbol{h} - h_{\hat{j}}\boldsymbol{e}_{\hat{j}}) = \sum_{\substack{\boldsymbol{h} \in H_{i+1,j,k} \\ h_{\hat{j}} \neq 0}} \binom{(h_{\hat{j}} - 1) + b_{i+1,\hat{j}}}{b_{i+1,\hat{j}}} \\ & \times \left(\frac{1}{\widetilde{\lambda}_j - \widetilde{\lambda}_{\hat{j}}} \right)^{h_{\hat{j}}} P_{i,j,k-h_{\hat{j}}}(\boldsymbol{h} - h_{\hat{j}}\boldsymbol{e}_{\hat{j}}) = \left(\frac{1}{\widetilde{\lambda}_j - \widetilde{\lambda}_{\hat{j}}} \right) \sum_{\boldsymbol{h} \in H_{i+1,j,k-1}} P_{i+1,j,k-1}(\boldsymbol{h}) \\ &= \frac{\hat{\Omega}_{i+1,j,k-1}}{\widetilde{\lambda}_j - \widetilde{\lambda}_{\hat{j}}} = -\frac{\hat{\Omega}_{i+1,j,k-1}}{\lambda_{i+1} - \widetilde{\lambda}_j} \end{aligned} \tag{25}$$

Otherwise, $a_{i+1} = a_i + 1, \widetilde{\lambda}_{a_{i+1}} = \lambda_{i+1}$. For all $j \in \{1, 2, \ldots, a_i\}$:

$$\begin{aligned}\hat{\Omega}_{i+1,j,k}-\hat{\Omega}_{i,j,k}&=\sum_{\boldsymbol{h}\in H_{i+1,j,k}}P_{i+1,j,k}(\boldsymbol{h})-\sum_{\boldsymbol{h}\in H_{i,j,k}}P_{i,j,k}(\boldsymbol{h})=\sum_{\substack{\boldsymbol{h}\in H_{i+1,j,k}\\ h_{a_{i+1}}\neq 0}}P_{i+1,j,k}(\boldsymbol{h})\\&=\sum_{\boldsymbol{h}\in H_{i+1,j,k-1}}\frac{1}{\widetilde{\lambda}_j-\widetilde{\lambda}_{a_{i+1}}}P_{i+1,j,k-1}(\boldsymbol{h})=\frac{\hat{\Omega}_{i+1,j,k-1}}{\widetilde{\lambda}_j-\widetilde{\lambda}_{a_{i+1}}}=-\frac{\hat{\Omega}_{i+1,j,k-1}}{\lambda_{i+1}-\widetilde{\lambda}_j}\end{aligned}\tag{26}$$

Secondly, it is verified, as shown below, that coefficients appear in the direct formula also satisfy Eqs. (16)–(18).

(1) For Eq. (16) recall that $j\in\{1,2,\ldots,a_i\}$ and $\widetilde{\lambda}_j\neq\lambda_{i+1}$, apply Eq. (21):

$$\begin{aligned}\Gamma_{i+1,j,b_{i+1,j}}&=\frac{N_1(0)}{b_{i+1,j}!\lambda_{i+1}}\Lambda_{i+1}\hat{\alpha}_{i+1,j}\hat{\Omega}_{i+1,j,0}\\&=\frac{N_1(0)}{b_{i,j}!\lambda_{i+1}}\lambda_{i+1}\Lambda_i\frac{\hat{\alpha}_{i,j}}{\lambda_{i+1}-\widetilde{\lambda}_j}\hat{\Omega}_{i,j,0}\\&=\frac{\lambda_i}{\lambda_{i+1}-\widetilde{\lambda}_j}\frac{N_1(0)}{b_{i,j}!\lambda_i}\Lambda_i\hat{\alpha}_{i,j}\hat{\Omega}_{i,j,0}=\frac{\lambda_i\Gamma_{i,j,b_{i,j}}}{\lambda_{i+1}-\widetilde{\lambda}_j}\end{aligned}\tag{27}$$

Making use of $b_{i+1,j}=b_{i,j}$ and Eq. (24):

$$\begin{aligned}\Gamma_{i+1,j,k}&=\frac{N_1(0)}{k!\lambda_{i+1}}\Lambda_{i+1}\hat{\alpha}_{i+1,j}\left(\hat{\Omega}_{i,j,b_{i,j}-k}-\frac{\hat{\Omega}_{i+1,j,b_{i,j}-(k+1)}}{\lambda_{i+1}-\widetilde{\lambda}_j}\right)\\&=\frac{\lambda_i}{\lambda_{i+1}-\widetilde{\lambda}_j}\frac{N_1(0)}{k!\lambda_i}\Lambda_i\hat{\alpha}_{i,j}\hat{\Omega}_{i,j,b_{i,j}-k}\\&\quad-\frac{k+1}{\lambda_{i+1}-\widetilde{\lambda}_j}\frac{N_1(0)}{(k+1)!\lambda_{i+1}}\Lambda_{i+1}\hat{\alpha}_{i+1,j}\hat{\Omega}_{i+1,j,b_{i,j}-(k+1)}\\&=\frac{\lambda_i\Gamma_{i,j,k}-(k+1)\Gamma_{i+1,j,k+1}}{\lambda_{i+1}-\widetilde{\lambda}_j}\quad(0\leqslant k<b_{i+1,j})\end{aligned}\tag{28}$$

(2) For Eq. (17), if $\exists\hat{j}\in\{1,2,\ldots,a_i\},\widetilde{\lambda}_{\hat{j}}=\lambda_{i+1}$, consequently $a_{i+1}=a_i, b_{i+1,\hat{j}}=b_{i,\hat{j}}+1$. Using Eq. (22) and Eq. (23) in the following identities:

$$\begin{aligned}\Gamma_{i+1,\hat{j},k}&=\frac{N_1(0)}{k!\lambda_{i+1}}\Lambda_{i+1}\hat{\alpha}_{i+1,\hat{j}}\hat{\Omega}_{i+1,\hat{j},b_{i+1,\hat{j}}-k}\\&=\frac{\lambda_i}{k}\frac{N_1(0)}{(k-1)!\lambda_i}\Lambda_i\hat{\alpha}_{i,\hat{j}}\hat{\Omega}_{i,\hat{j},b_{i,\hat{j}}-(k-1)}\\&=\frac{\lambda_i\Gamma_{i,\hat{j},k-1}}{k}\quad(0<k\leqslant b_{i+1,\hat{j}})\end{aligned}\tag{29}$$

$$\begin{aligned}\Gamma_{i+1,\hat{j},0}&=\frac{N_1(0)}{\lambda_{i+1}}\Lambda_{i+1}\hat{\alpha}_{i+1,\hat{j}}\hat{\Omega}_{i+1,\hat{j},b_{i+1,\hat{j}}}\\&=\frac{N_1(0)}{\lambda_{i+1}}\Lambda_{i+1}\left(-\sum_{\substack{j=1\\ j\neq\hat{j}}}^{a_{i+1}}\hat{\alpha}_{i+1,j}\hat{\Omega}_{i+1,j,b_{i+1,j}}\right)\\&=-\sum_{\substack{j=1\\ k\neq\hat{j}}}^{a_{i+1}}\Gamma_{i+1,j,0}\end{aligned}\tag{30}$$

(3) For Eq. (18), if $\forall\hat{j}\in\{1,2,\ldots,a_i\},\widetilde{\lambda}_{\hat{j}}\neq\lambda_{i+1}$, the following identity could be obtained similarly as the above equation:

$$\Gamma_{i+1,a_{i+1},0}=-\sum_{j=1}^{a_i}\Gamma_{i+1,j,0}\tag{31}$$

2.5. Analysis of the improved efficiency

Various factors could affect the actual computational efficiency, such as the numbers of different kinds of arithmetic operation and cache efficiency, an inclusive efficiency analysis will be quite complex and even dependent on machine. Instead, a preliminary analysis based on the numbers of multiplications and divisions is carried out. The cost of exponential evaluations is not considered, because the required number of evaluations is limited to several times of the number of nuclides as indicated above, which is negligible with respect to the numbers of multiplications and divisions in practice.

For the former practice of cutoff check, a level $k+1$ pseudo node is calculated after adding a level k actual node. It can be predicted that nearly half of the time consumption could be saved with newly proposed cutoff check implemented.

For the analysis of different analytic formulas. Let x and y denote the costs of multiplication and division respectively. To find the solution of a level k node, the computational effort required by the Bateman formula (Eq. (7)) could be expressed as $3kx+k(k-1)y$, and the Cetnar formula (Eq. (8)) is generally less efficient than Bateman formula, while the recursive formula needs at most $(3k-2)x+(k-1)y$. The cost of direct formula is more expensive, because the recursive formula has the advantage of making use of predecessor node solution. For a particular TTA solution, define the following quantities:

$$\begin{aligned}J_1&=\frac{\sum_{k=1}^{L_{max}}kI_k}{I_t}\\J_2&=\sqrt{\frac{\sum_{k=1}^{L_{max}}k^2I_k}{I_t}}\end{aligned}\tag{32}$$

where I_k stands for the number of level k nodes, I_t is the total number of nodes.

Let $f_1(x,y)$ and $f_2(x,y)$ express the overall costs of multiplications and divisions of direct and recursive formulas respectively:

$$f_1(x,y)\approx\sum_{k=1}^{L_{max}}(3kx+k(k-1)y)I_k=I_t\left(3J_1x+(J_2^2-J_1)y\right)\tag{33}$$

$$\begin{aligned}f_2(x,y)&\approx\sum_{k=1}^{L_{max}}((3k-2)x+(k-1)y)I_k\\&=I_t((3J_1-2)x+(J_1-1)y)\end{aligned}\tag{34}$$

For example, test case 1 in the following section with cutoff being 10^{-20} results in $J_1\approx 9.4623$ and $J_2\approx 9.8373$. Set $y=4x\neq 0$, where the value of 4 comes from simple tests. The speedup factor of recursive formula against the Bateman formula is estimated as the ratio f_1/f_2, which is approximately 6.27. When compared to Cetnar formula, the estimate will be slightly greater according to a previous study (Isotalo and Aarnio, 2011).

3. Numerical results

Based on the above theory, a numerical code has been developed. For the purpose of verifying the methods and codes, numerical results are provided in this section.

The ORIGEN2 libraries (Croff, 1980) are chosen as the data sources of depletion equations. There are 1307 nuclides and 1421 transfer relations in the decay library. The cross section library is subject to PWR type, with 466 nuclides having defined cross sections. And there are 8 fissile nuclides and 810 fission products. All calculations are performed on a single thread 3.2 GHz E6700 processor.

Three test cases are defined, and they are sketched in Fig. 2. In case 1, the fresh composition of ^{235}U, ^{238}U and ^{16}O with fraction ratio 0.034:0.966:2.0 is irradiated for 50 days by a uniform neutron flux of $3\times 10^{14}(\mathrm{cm}^{-2}\cdot\mathrm{s}^{-1})$. The used and discharged fuel compositions are obtained from fresh composition after irradiations of 300 and 1500 days respectively with the presence of the same neutron flux as case 1. Then, in case 2, the used fuel composition is irradi-

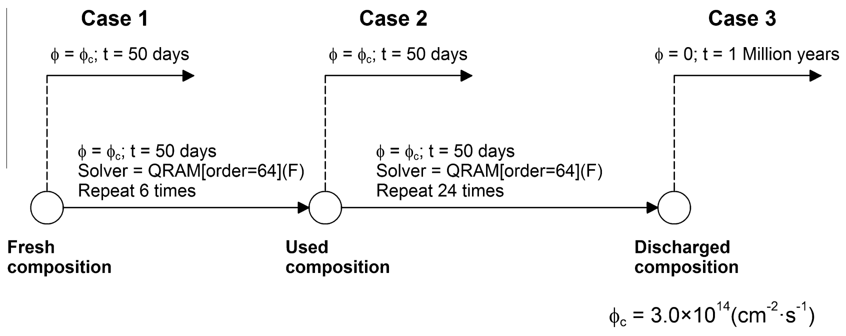

Fig. 2. Sketch of defined test cases.

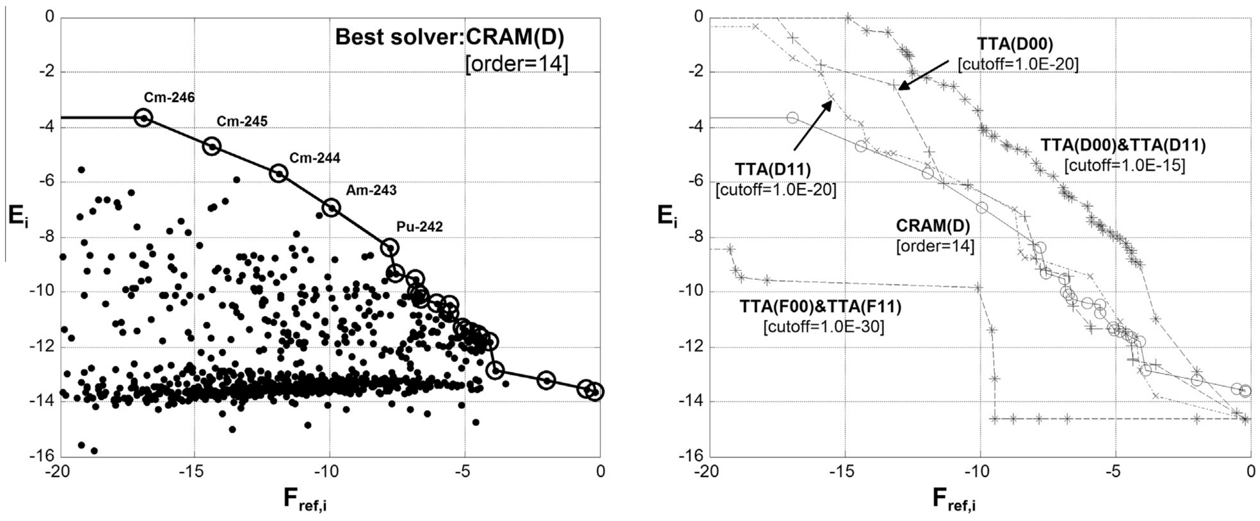

Fig. 3. Relative errors for individual nuclides and error bounding curves in CASE 1.

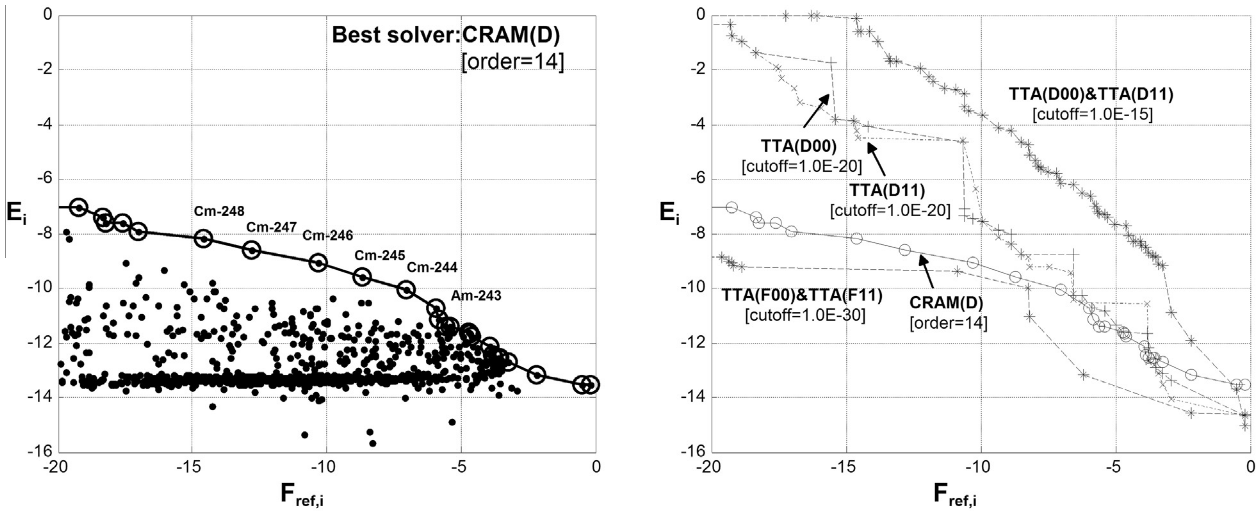

Fig. 4. Relative errors for individual nuclides and error bounding curves in CASE 2.

ated for 50 days; and in case 3, the discharged fuel decays for one million years. Compared with case 1, case 2 features the composition changing effect, and case 3 reflects the problem type changing effect additionally.

Each of the designation of solvers contains two parts, the name of the method and the implementation details. Three methods are compared, which are the TTA method (Cetnar, 2006), CRAM (Pusa and Leppänen, 2010; Pusa, 2011) and QRAM (Quadrature Rational Approximation Method) (Pusa, 2011; Trefethen et al., 2006). The implementation details take the form of “D/F” + “0/1” + “0/1”, where “D” and “F” respectively stand for double precision and extended double precision, “1” or “0” indicate the implementation

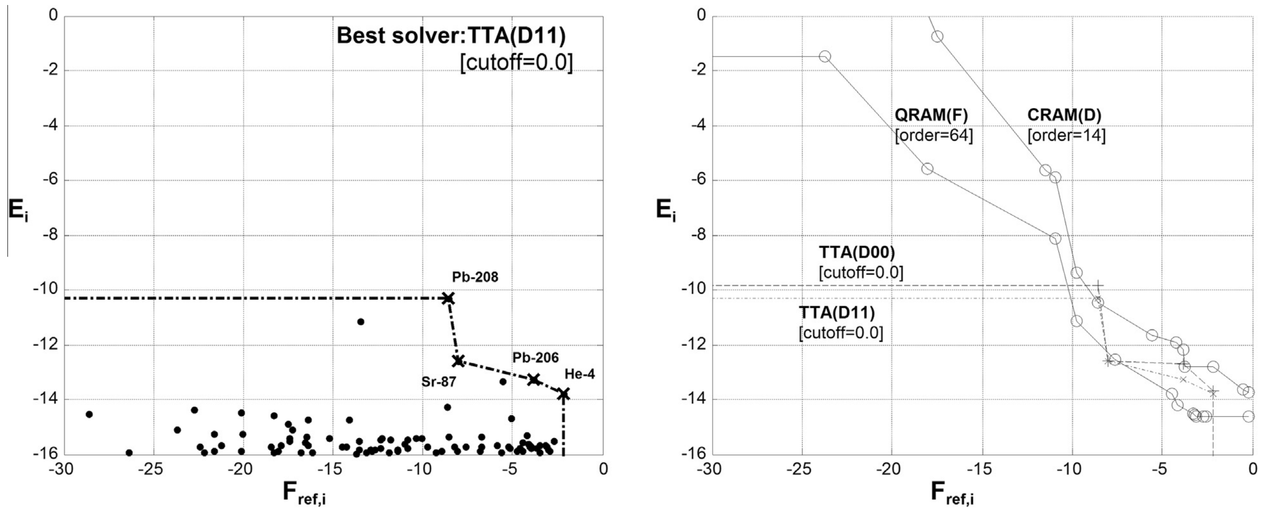

Fig. 5. Relative errors for individual nuclides and error bounding curves in CASE 3.

Table 1
Performance of various solvers and selected parameters of linear chains in CASE 1.

Methods		Time	Characters of processed linear chains			The maximum E_i value for i satisfying $F_{ref,i} \in$		
Part 1	Part 2	(ms)	C_t	L_{ave}	L_{max}	[−5,0)	[−10,−5)	[−20,−10)
TTA [10^{-15}]	(D00)	77.30	20498	8.8094	16	−7.9819	−4.0320	> 0
	(D10)	37.60	20,498	8.8094	16	−7.9819	−4.0320	> 0
	(D11)	10.80	20,498	8.8094	16	−7.9819	−4.0320	> 0
TTA [10^{-20}]	(D00)	645.8	133,508	10.4179	21	−11.3625	−7.2417	> 0
	(D10)	312.6	133,362	10.4116	20	−11.3678	−7.4208	−0.2903
	(D11)	74.26	133,670	10.4221	20	−11.0767	−6.9905	−0.3370
TTA [10^{-20}]	(F00)	15,630	133,344	10.4110	20	−12.1997	−8.1636	−0.4280
	(F11)	2345	133,344	10.4110	20	−12.1997	−8.1636	−0.4280
TTA [10^{-30}]	(F00)	9.394E5	4,494,055	13.9549	26	−14.6170	−11.4003	−8.4262
	(F11)	1.036E5	4,494,055	13.9549	26	−14.6170	−11.4003	−8.4262
CRAM [14]	(D)	4.742	N/A			−11.4616	−6.9242	−3.6610
QRAM [64]	(F)	2387	N/A			Reference solution		

or no implementation of the two proposed improvements. For instance, TTA(F10) represents the extended double precision TTA, with the new cutoff check and original direct formula implemented. Both CRAM and QRAM are suitable for irradiation cases, while QRAM has less convergence efficiency, but more easily obtains coefficients for the rational function approximation. Reference solutions for the first two cases are calculated by QRAM(F) with order of 64, which is much more accurate than the other solvers for solving burnup type problems. TTA could calculate decay problems exactly without consideration of finite precision, and the reference solution for the last case is provided by TTA(F00) with cutoff of zero. Pseudo codes of the two analytic formulas and CRAM are provided in Appendix A, since they are essential for the efficiency consideration.

Numerical results for selected test cases are listed in Table 1–3, and detailed error plots and error bounding curves could be found in Fig. 3–5. C_t represents the number of linear chains, L_{ave} and L_{max} denote the average and the maximum linear chain lengths. E_i and $F_{ref,i}$ are defined as the base 10 logarithms of absolute relative error, and number density fraction of ith nuclide. For each test case, the solver that has an accuracy well below nuclear data uncertainty level and the least running time is chosen as the best solver. Since the burnup type depletion problems were already discussed thoroughly in a previous study (Isotalo and Aarnio, 2011), we shall focus on the discussion of TTA performance and the numerical behavior of introduced improvements rather than performance differences among different methods.

For test case 1, the numerical results are shown in Table 1. The three selected parameters, C_t, L_{ave} and L_{max} of linear chains grow as the cutoff value approaches zero. The identical results for TTA solvers with cutoff being 10^{-15} indicate the absence of notable numerical instability. Whereas for cutoff being 10^{-20}, accuracy damage is observed in double precision solvers with comparison to their extended double precision counterparts, in fact choosing smaller cutoff value is almost meaningless. Nevertheless, all double precision TTA methods have similar accuracy, indicating that the two proposed improvements do not cause numerical deficiency.

Compared with case 1, case 2 as shown in Table 2 includes fewer solvers for comparison. Because the same observations and conclusions from case 1 apply as well. For TTA solvers, a slowing down factor of about 4 together with slightly decreasing average linear chain length is observed. The reason is that case 2 contains much more nuclides initially, and the linear chains originating from these latest emerged nuclides are considered, while their smaller average length is due to relatively smaller number

Table 2
Performance of various solvers and selected parameters of linear chains in CASE 2.

Methods		Time	Characters of processed linear chains			The maximum E_i value for i satisfying $F_{ref,i} \in$		
Part 1	Part 2	(ms)	C_t	L_{ave}	L_{max}	[−5,0)	[−10,−5)	[−20,−10)
TTA [10^{-15}]	(D00)	200.2	65,261	7.8490	16	−7.7281	−3.6858	> 0
	(D11)	30.67	65,261	7.8490	16	−7.7281	−3.6858	> 0
TTA [10^{-20}]	(D00)	2298	536,415	9.5488	20	−11.3405	−7.5419	−0.3362
	(D11)	287.0	536,934	9.5554	22	−10.5850	−7.5419	−0.3362
TTA [10^{-30}]	(F00)	4.107E6	22,806,207	12.8436	26	−14.5929	−9.9934	−8.8761
	(F11)	4.971E5	22,806,207	12.8436	26	−14.5929	−9.9934	−8.8761
CRAM [14]	(D)	4.945	N/A			−11.6036	−9.5802	−7.0412
QRAM [64]	(F)	2356	N/A			Reference solution		

Table 3
Performance of various solvers and selected parameters of linear chains in CASE 3.

Methods		Time	Characters of processed linear chains			The maximum E_i value for i satisfying $F_{ref,i} \in$		
Part 1	Part 2	(ms)	C_t	L_{ave}	L_{max}	[−10,0)	[−20,−10)	[−30,−20)
TTA [0.0]	(D00)	8.443	4558	7.1531	23	−9.8193	−10.5809	−13.7935
	(D11)	1.448	4558	7.1531	23	−10.3268	−11.1502	−14.3434
CRAM [14]	(D)	2.109		N/A		−9.3630	> 0	> 0
QRAM [64]	(F)	405.3		N/A		−11.1502	−5.6023	−1.4819
TTA [10^{-30}]	(F11)	183.6	4558	7.1531	23	< -15	< -15	< -15
	(F00)	39.41	4558	7.1531	23	Reference solution		

densities of such root nuclides. Nevertheless, in view of the huge performance gap between TTA and CRAM methods for solving burnup problems, the introduced improvements for TTA method are not very important.

Numerical results of case 3 are listed in Table 3, and the detailed error plot is presented in Fig. 5. There is no closed cycle in the decay system, and all linear chains are terminated by stable nuclides and have well bounded lengths. With the help of the introduced improvements, the TTA(D11) solver outperformed other solvers in terms of accuracy and efficiency.

4. Conclusion

In this paper, two improvements are proposed to the TTA method. The first one utilizes the time-averaged number densities to simplify the cutoff check. The second one replaces the legacy direct formula with a new recursive formula for finding the analytic solution for the linear chains. Numerical tests based on typical PWR depletion calculations show that, these two improvements offer speedup factors of about 2 and 4 respectively, while the accuracy of TTA method is preserved. As a consequence, TTA method is recommended for solving decay problems. It should also be noticed that the newly proposed recursive formula only involves one subchain solution, and is more concisely implemented.

Acknowledgment

This research was carried out under the financial support from the National Science Foundation of China (approved number: 91126005 and 91226106). Acknowledgment goes to Prof. Barry D. Ganapol from University of Arizona, USA, for providing assistance in the preparation of the revised paper.

Appendix A

The important variables appeared in the following description of algorithms are listed below:

- n: The number of nuclides;
- A: The $n \times n$ depletion matrix;
- t: Depletion time;
- *DEN*: The initial number density vector;
- *DENP*: The solution number density vector;
- *max_len* : The maximum length of linear chain;
- *chain_len* : The length of linear chain;
- $node\_order(max\_len)$: The ith element keeps the number of nodes in the first $(i-1)$ nodes, which have the same vanishing coefficients with the ith node;
- $node\_pre(max\_len)$: The *i'th* element keeps the level number of the node, which has the same vanishing coefficient with, resides in the upstream of and most close to the ith node. If such node does not exist, it is assigned with -1;
- $lambda(max\_len)$: The vanishing coefficients of nodes;
- $lambda\_next(max\_len)$: The ith element represents the transfer coefficient from the ith node nuclide to $(i+1)$th node nuclide;
- $p(max\_len) : p(i) = i(i-1)/2$;
- $r(p(max\_len) + max\_len) : r(p(i) + 1 : p(i) + i)$ stores the solution coefficients for the ith node.

The solution to the ith node is expressed as:

$$N_i(t) = \sum_{j=1}^{i} r(p(i)+j)t^{node\_order(j)} \exp(-lambda(j) \cdot t) \tag{35}$$

The pseudo code of originally implemented TTA is briefly presented below:

Algorithm 1

```
1: for $m = 1$ to $n$ do                 # Loop over nuclides with positive initial densities
2:   if $DEN(m) \leqslant 0.0$; continue
3:   Initialize a linear chain with $m$'th nuclide being the root node
$chain\_len \leftarrow 1$                 # Length of the chain
$restart\_pos \leftarrow 1$               # The position of restart node
$grow\ flag \leftarrow 0$                 # The node will be added is pesudo/actual (0/1)
$r(1) \leftarrow DEN(m)$                  # Apply Eq. (10)
......
4:   Tally contribution of the root node
5:   while True do                        # Loop for enumerating all important chains
6:     while True do                      # Loop for chain growing from restart node
7:       if The last node does not have successors; exit
8:       if $grow\ flag = 1$ and Cutoff check fails; exit
9:       if $chain\_len \geqslant max\_len$; exit
10:       if $grow\ flag = 1$ then
11:         Append an unexplored successor of the last node
12:       else
13:         Append a pseudo node
14:       end if
15:       $i \leftarrow chain\_len + 1$
16:       DIRECT_SOLVE($i$)               # Use direct formula to solve the $i$'th node
17:       if $grow\ flag = 1$; $chain\_len \leftarrow chain\_len + 1$
18:       $grow\ flag \leftarrow 1 - grow\ flag$
19:       end while
20:       Tally contributions of the $(restart\_pos + 1)'th$ node to the last node
21:       $chain\_len \leftarrow chain\_len - 1$
22:       while $chain\_len > 0$ do       # Loop for identifying restart node
23:         if All successors of the last node have been explored then
24:         $chain\_len \leftarrow chain\_len - 1$; continue
25:       else
26:         exit
27:       end if
28:     end while
29:     if $chain\_len = 0$; exit         # The restart node can not be found
30:     $restart\_pos \leftarrow chain\_len$
31:     $grow\ flag \leftarrow 1$
32:   end while
33: end for
```

Algorithm 2

```
1: procedure DIRECT_SOLVE (i)
2:   Calculate $N_1(0)\Lambda_{i-1}$
3:   for $j = 1$ to $a_i$ do
4:     Calculate $\hat{\alpha}_{i,j}$
5:     if $b_{i,j} = 0$ then
6:       $\gamma_{i,j,0} \leftarrow N_1(0)\Lambda_{i-1}\hat{\alpha}_{i,j}$; continue
7:     end if
8:     for $p = 1$ to $a_i$ do
9:       if $p = j$; continue
10:         for $k = 0$ to $b_{i,j}$ do
11:         $\hat{\Omega}^p_{i,j,k} \leftarrow \sum_{q=0}^{k} \hat{\Omega}^{p-1}_{i,j,k-q} \binom{q + b_{i,p}}{b_{i,p}} \left(\frac{1}{\tilde{\lambda}_j - \tilde{\lambda}_p}\right)^q$
12:         end for
13:     end for
14:     for $k = 0$ to $b_{i,j}$ do
15:       $\gamma_{i,j,k} \leftarrow \frac{N_1(0)}{k!}\Lambda_{i-1}\hat{\alpha}_{i,j}\hat{\Omega}_{i,j,b_{i,j}-k}$
16:     end for
17:   end for
18: end procedure
```

The improvement on cutoff check is achieved by removing the pseudo node calculations and adding an array to keep track the average number densities of the nodes.

The pseudo code of the direct solution procedure in Algorithm 1 is provided below. The term $\hat{\Omega}^p_{i,j,k}$ is defined as:

$$\begin{aligned} H^p_{i,j,k} &= \left\{ \boldsymbol{h} | \boldsymbol{h} \in H_{i,j,k} \ \textbf{and} \ \sum_{q=p+1}^{a_i} h_q = 0 \right\} \\ \hat{\Omega}^p_{i,j,k} &= \sum_{\boldsymbol{h} \in H^p_{i,j,k}} P_{i,j,k}(\boldsymbol{h}) \end{aligned} \tag{36}$$

The pseudo code of the recursive formula is presented below, it is supposed to replace the direct solution procedure in Algorithm 1. *node_mark* is one dimensional integer array with length being *max_len*.

The complex linear systems in CRAM are solved by Gauss–Seidel method with a simple preconditioning technique. α_* and θ_* are coefficients of the rational approximation function in partial fraction form. And setting cutoff value as 10^{-30} will obtain sufficient accuracy. The pseudo code is presented below:

Algorithm 3

```
1: $\lambda \leftarrow lambda(chain\_len + 1)$
2: $node\_mark \leftarrow 0$
3: $tmp1 \leftarrow 0.0$
4: for $j = chain\_len$ to $1$ step $-1$ do
5:   if $node\_mark(j) \neq 0$; continue
6:   if $j = node\_pre(chain\_len + 1)$ then
7:     $pos \leftarrow j$
8:     for $k = 1$ to $node\_order(j) + 1$ do
9:       $node\_mark(pos) \leftarrow 1$
10:      $pos \leftarrow node\_pre(pos)$
11:    end for
12:    continue
13:  end if
     # Apply Eq. (16)
14:  $tmp2 \leftarrow lambda\_next(chain\_len) \times r(p(chain\_len) + j)/(\lambda - lambda(j))$
15:  $r(p(chain\_len + 1) + j) \leftarrow tmp2$
16:  $pos \leftarrow j$
17:  for $k = node\_order(j)$ to $1$ step $-1$ do
18:    $pos \leftarrow node\_pre(pos)$
19:    $tmp2 \leftarrow (lambda\_next(chain\_len) \cdot r(p(chain\_len) + pos) - k \cdot tmp2)/(\lambda - lambda(j))$
20:    $r(p(chain\_len + 1) + pos) \leftarrow tmp2$
21:    $node\_mark(pos) \leftarrow 1$
22:  end for
23:  $tmp1 \leftarrow tmp1 + tmp2$
24: end for
25: if $node\_order(chain\_len + 1) = 0$ then
     # Apply Eq. (18)
26:  $r(p(chain\_len + 1) + chain\_len + 1) \leftarrow -tmp1$
27: else
    # Apply Eq. (17)
28:  $pos \leftarrow chain\_len + 1$
29:  for $j = node\_order(chain\_len + 1)$ to $1$ step $-1$ do
30:    $pos2 \leftarrow node\_pre(pos)$
31:    $r(p(chain\_len + 1) + pos) \leftarrow lambda\_next(chain\_len) \times r(p(chain\_len) + pos2)/j$
32:    $pos \leftarrow pos2$
33:  end for
34:  $r(p(chain\_len + 1) + pos) \leftarrow -tmp1$
35: end if
```

Algorithm 4

```
1: $D \leftarrow \mathbf{diag}(tA)$
2: $\varepsilon \leftarrow \text{cutoff} \times \sum_{i=1}^{n} |DEN(i)|$        # Convergence criteria
3: $DENP \leftarrow 0.0$
4: for $i = 1$ to $order/2$ do                  # Solve $order/2$ linear systems
5:   $AI \leftarrow (tA - \theta_i \cdot \mathrm{I})(D - \theta_i \cdot \mathrm{I})^{-1}$        # Preconditioning
6:   $DENI \leftarrow AI^{-1} \cdot DEN$              # Solve linear system by Gauss–Seidel method
7:   $DENI \leftarrow \alpha_i \cdot (D - \theta_i \cdot \mathrm{I})^{-1} \cdot DENI$
8:   $DENP \leftarrow DENP + \mathrm{Re}(DENI)$
9: end for
10: $DENP \leftarrow 2\, DENP + \alpha_0 \cdot DEN$
```

Appendix B

The lower triangular Bateman equations is presented below:

$$\begin{aligned} \frac{dN_1(t)}{dt} &= -\lambda_1 N_1(t) \\ \frac{dN_{i+1}(t)}{dt} &= \sum_{p=1}^{i} \lambda_{i+1,p} N_p(t) - \lambda_{i+1} N_{i+1}(t) \quad (i \geqslant 1) \end{aligned} \tag{37}$$

The solution form and the solution to first node is the same as bi-diagonal case, and could be found in Eqs. (9) and (10). The inductive part of the recursive formula is as follows:

At first, for all j that $\widetilde{\lambda}_j \neq \lambda_{i+1}$,

$$\begin{aligned} & b_{i+1,j} = b_{i,j} \\ & \gamma_{i+1,j,b_{i+1,j}} = \frac{\sum_{p=q(j,b_{i,j})}^{i} \lambda_{i+1,p}\gamma_{p,j,b_{i,j}}}{\lambda_{i+1} - \widetilde{\lambda}_j} \\ & \gamma_{i+1,j,k} = \frac{\sum_{p=q(j,k)}^{i} \lambda_{i+1,p}\gamma_{p,j,k} - (k+1)\gamma_{i+1,j,k+1}}{\lambda_{i+1} - \widetilde{\lambda}_j} \quad 0 \leqslant k < b_{i,j} \end{aligned} \tag{38}$$

Secondly, if $-\lambda_{i+1}$ coincides with vanishing coefficients introduced by the first i nuclides, that is $\exists \hat{j} \in \{1,2,\ldots,a_i\}, \widetilde{\lambda}_{\hat{j}} = \lambda_{i+1}$, then:

$$\begin{aligned} & a_{i+1} = a_i \quad b_{i+1,\hat{j}} = b_{i,\hat{j}} + 1 \\ & \gamma_{i+1,\hat{j},k} = \frac{\sum_{p=q(\hat{j},k-1)}^{i} \lambda_{i+1,p}\gamma_{p,\hat{j},k-1}}{k} \quad 1 \leqslant k \leqslant b_{i,\hat{j}} + 1 \\ & \gamma_{i+1,\hat{j},0} = N_{i+1}(0) - \sum_{\substack{j=0 \\ j \neq \hat{j}}}^{a_{i+1}} \gamma_{i+1,j,0} \end{aligned} \tag{39}$$

Otherwise $-\lambda_{i+1}$ is a new vanishing coefficient, which means $\forall \hat{j} \in \{1,2,\ldots,a_i\}, \lambda_{\hat{j}} \neq \lambda_{i+1}$, then:

$$\begin{aligned} & a_{i+1} = a_i + 1 \quad b_{i+1,a_{i+1}} = 0 \\ & \widetilde{\lambda}_{a_{i+1}} = \lambda_{i+1} \quad \gamma_{i+1,a_{i+1},0} = N_{i+1}(0) - \sum_{j=1}^{a_i} \gamma_{i+1,j,0} \end{aligned} \tag{40}$$

The notation $q(int1, int2)$ is defined as:

$$q(int1, int2) = min\{q \in \mathbb{N}^+ : a_q \geqslant int1, b_{q,int1} \geqslant int2\} \tag{41}$$

References

Bateman, H., 1910. The solution of a system of differential equations occurring in the theory of radioactive transformations. In: Proc. Cambridge Philos. Soc, Vol. 15, pp. 423–427.

Cetnar, J., 2006. General solution of bateman equations for nuclear transmutations. Ann. Nucl. Energy 33 (7), 640–645. http://dx.doi.org/10.1016/j.anucene.2006.02.004.

Croff, A.G. (1980). User's Manual for the Origen2 Computer Code. Tech. Report. Oak Ridge National Lab., TN, USA.

Dreher, R., 2013. Modified bateman solution for identical eigenvalues. Ann. Nucl. Energy 53, 427–438. http://dx.doi.org/10.1016/j.anucene.2012.06.019.

Hamawi, J., 1971. Useful Recurrence Formula for the Equations of Radioactive Decay. Tech. Report. Stone and Webster Engineering Corp., Boston.

Hermann, O., Westfall, R., 1998. Origen-s: scale system module to calculate fuel depletion, actinide transmutation, fission product buildup and decay, and associated radiation source terms. In: Vol. II, Sect. F7 of SCALE: A Modular Code System for Performing Standardized Computer Analyses for Licensing Evaluation, NUREG/CR-0200, Rev 6.

Isotalo, A., 2013. Computational Methods for Burnup Calculations with Monte Carlo Neutronics (Ph.D. thesis). Aalto University.

Isotalo, A., Aarnio, P., 2011. Comparison of depletion algorithms for large systems of nuclides. Ann. Nucl. Energy 38 (2), 261–268. http://dx.doi.org/10.1016/j.anucene.2010.10.019.

Leonard, I., 1996. The matrix exponential. SIAM Rev. 38 (3), 507–512. http://dx.doi.org/10.1137/S0036144595286488.

Miles, R.E., 1981. An improved method for treating problems involving simultaneous radioactive decay, buildup, and mass transfer. Nucl. Sci. Eng. 79, 239–245.

Moler, C., Van Loan, C., 2003. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. 45 (1), 3–49. http://dx.doi.org/10.1137/S00361445024180.

Pusa, M., 2011. Rational approximations to the matrix exponential in burnup calculations. Nucl. Sci. Eng. 169 (2), 155–167.

Pusa, M., Leppänen, J., 2010. Computing the matrix exponential in burnup calculations. Nucl. Sci. Eng. 164 (2), 140.

Rubinson, W., 1949. The equations of radioactive transformation in a neutron flux. J. Chem. Phys. 17, 542. http://dx.doi.org/10.1063/1.1747317.

Stamm'ler, R.J., Abbate, M.J., 1983. Methods of Steady-State Reactor Physics in Nuclear Design. Academic Press.

Trefethen, L.N., Weideman, J., Schmelzer, T., 2006. Talbot quadratures and rational approximations. BIT Numer. Math. 46 (3), 653–670. http://dx.doi.org/10.1007/s10543-006-0077-9.