




CNUCTRAN: A C++ NUCLIDES TRANSMUTATION SIMULATION CODE BASED ON THE PROBABILISTIC METHOD



**M. R. OMAR
SCHOOL OF PHYSICS
UNIVERSITI SAINS MALAYSIA**

Contents

Section 1: Introduction	2
Section 2: Calculation Objective	2
Section 3: What is <i>CNUCTRAN</i> ?	2
Section 4: How Does <i>CNUCTRAN</i> Works?	3
Section 5: How to Install.....	4
Section 6: Extensible Markup Language (XML)	4
Section 7: Creating an Input File.....	5
Section 8: Sample Input File	7
Section 9: Performing Calculations & Program Output.....	9
Section 10: Theory	10
10.1 Bateman Equations.....	10
10.2 Discrete Probability Distribution of Transmutation Events	11
10.3 The Probabilistic Method	12
10.4 Approximation Order	13
Section 11: Some Remarks	14
License	15
References	16

Section 1: Introduction

Welcome to the *CNUCTRAN* Beginner's Guide. This document will guide you through the essentials of using *CNUCTRAN* to perform nuclides transmutation simulations. Throughout the reading, you will go through the following key topics:

- Calculation Objective
- What is *CNUCTRAN*?
- How does the *CNUCTRAN* code work?
- *CNUCTRAN* Installation
- The basics of Extensible Markup Language (XML)
- Creating an input file: Defining Zones & Simulation Settings
- Examples of an input file.
- Program output
- Theory

Section 2: Calculation Objective

Suppose you have a system of I nuclides, where $I \in \{1, 2, 3, \dots\}$. As time progresses from $t = 0$ to $t = T$, the initial concentration of these nuclides will evolve into the final concentration. Such an evolution is due to various transmutation reactions occurring within the time interval. *CNUCTRAN* aims to compute the final concentration of these transmuting nuclides. Some nuclides will increase in concentration, and some will deplete. Note also that the phrase "concentration of nuclide A" also means the nuclei count of nuclide A.

Section 3: What is *CNUCTRAN*?

CNUCTRAN is a C++ code that simulates various nuclear transmutations such as decays, fissions, and neutron absorptions. Most importantly, it helps physicists avoid cumbersome numerical issues of solving the nuclide burnup equations using the probabilistic approach. Several other techniques, such as the Transmutation Trajectory Analysis (TTA) and Chebychev Rational Approximation Method (CRAM), are also available, and these methods use different computational approaches. The primary purpose of *CNUCTRAN* is to explore the feasibility of using the probabilistic approach to solving nuclides burnup problems. When we talk about the probabilistic method, the method may sound stochastic. In reality, the technique does not incorporate any means of random sampling, and the final concentration produced is not contaminated with random errors.

CNUCTRAN works based on the probabilistic method, where it carefully tracks nuclide transformations into one another in a depletion problem. Interestingly, the probabilistic approach does not directly solve Bateman equations. Instead, the technique reduces nuclide transformations into a sparse transfer matrix, \mathbf{T} , whose elements are made up of various transformation probabilities. Next, the transfer matrix serves as a multiplier of the initial nuclide concentration, \mathbf{y}_0 , producing the final nuclide concentrations, \mathbf{y} .

Technically, the users may incorporate as many nuclides as possible into a burnup calculation. The calculation involves I nuclides, then, \mathbf{y}_0 and \mathbf{y} are column vectors of I elements, and \mathbf{T} is a $I \times I$ sparse square matrix. *CNUCTRAN* aims to compute matrix \mathbf{T} , so that the final nuclides concentration can be expressed as,

$$\mathbf{y} = \mathbf{T} \mathbf{y}_0 \quad (1)$$

Constructing the matrix **T** requires several vital pieces of information. These include the half-life, λ , of the nuclides involved in the calculation and the condition of the environment. An example of such a condition is the presence of the neutron flux, ϕ , which will initiate neutron-induced transmutation reactions. However, the program does not accept ϕ as its input. Instead, it needs the reaction rate constant, Λ , which has a unit of reaction per second. For a decay reaction, Λ itself is the decay constant,

$$\Lambda = \frac{\ln 2}{\lambda} \quad (2)$$

Of course, the probabilistic approach has its pros and cons. The central valuable aspect of the probabilistic method is that it imitates the actual nuclides transformation processes. Therefore, it can be a reference solution for other calculational techniques. However, it comes at the expense of facing the limitations presented in Table 1.

Table 1: The pros and cons of the probabilistic method.

Pros	Cons
It is easy to understand. The method of solution does not require knowledge of advanced mathematics.	The method requires arbitrary precision arithmetics. The conventional float/double data types may contaminate the accuracy of the final nuclides concentration.
The solution produced is an approximated nuclides concentration, and the accuracy of the approximation is prescribed by the user via the input file (See Section 7).	The method requires sparse matrix-matrix multiplication (SpMM) parallelization to improve its computational efficiency.
Its computational efficiency is practical. For instance, it can solve pure decay problems with ~1500 nuclides in less than 1 second using a 3.2GHz eight cores CPU.	The neutron flux is assumed to be constant throughout the calculation time step for a system of nuclides under neutron irradiation.
The method can simulate complex burnup chains, and one does not need to linearize the burnup chains.	
The method faithfully simulates the transformation of nuclides and is free from any mathematical jargon. There is no risk of producing negative nuclides concentrations.	

Section 4: How Does *CNUCTRAN* Works?

Before doing anything, the code must first have a model of some problem of interest. The model comprises one or more zones in a nuclear reactor or any other physical system containing the nuclides under study. As the code user, you must describe the model so the code can do something with it. A basic model consists of a few things:

- The description of zones – the problem must be split up into regions with different nuclide compositions. For example, each nuclear fuel pin can serve as a single zone. The various fuel pins with varying burnup levels correspond to the other zones.
- Various simulation settings tell the code how many precision digits need to include, the time step, the order of the calculation and what options to use.

CNUCTRAN retrieves these definitions in terms of XML markups written in a single input file (with the *.xml* extension). Therefore, the user must know how to use XML markups. But do not worry. XML is just a

simple markup language and is easy to learn and use. If you have no idea what XML is, please visit this tutorial website: <https://www.w3schools.com/xml/>.

Section 5: How to Install

The current development of *CNUCTRAN* only provides a Windows executable. For other operating systems, you may have to compile it on your own. Unfortunately, you may have to replace the [Microsoft Parallel Pattern Library](#) (PPL) with any other compatible infrastructure, such as OpenMP. Note that *CNUCTRAN* requires PPL for its optimization.

The installation of *CNUCTRAN* involves the following simple steps:

Step 1: Visit <https://github.com/rabieomar92/cnuctran/releases>. Select the latest release and download the compressed file named *cnuctran.zip*.

Step 2: Extract the contents of *cnuctran.zip* into the same working directory.

Step 3: Double-click the *cmd* link file. The command prompt console will pop up from the current working directory.

Step 4: Prepare the XML input file. You may rename the input XML file according to your needs. If you are a first-timer, you may skip this step because a predefined input file is already included in *cnuctran.zip*.

Step 5: If the input file name is "*input.xml*", then type in "*cnuctran*" in the command prompt console, then press "enter". Or else, type in "*cnuctran [anyname].xml*".

When running the code, make sure that it is error-free. If it runs properly, then you are ready to go!

Section 6: Extensible Markup Language (XML)

You may skip this section if you are used to XML. Unlike many other nuclear-related codes, which use an arbitrary-format ASCII file with "cards" to specify a particular geometry, materials, and associated run settings, the input files for *CNUCTRAN* are structured in an XML file. XML, which stands for eXtensible Markup Language, is a simple format that allows data to be exchanged efficiently between different programs and interfaces.

Anyone who has seen webpages written in HTML will be familiar with the structure of XML, whereby "tags" enclosed in angle brackets denote that a particular piece of data will follow. Let us examine the following example:

```
<person>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
  <age>27</age>
  <occupation>Health Physicist</occupation>
</person>
```

Here we see that the first tag indicates that the following data will describe a person. The nested tags *firstname*, *lastname*, *age*, and *occupation* exhibit the characteristics of the person being described. They are the child of *person* tag.

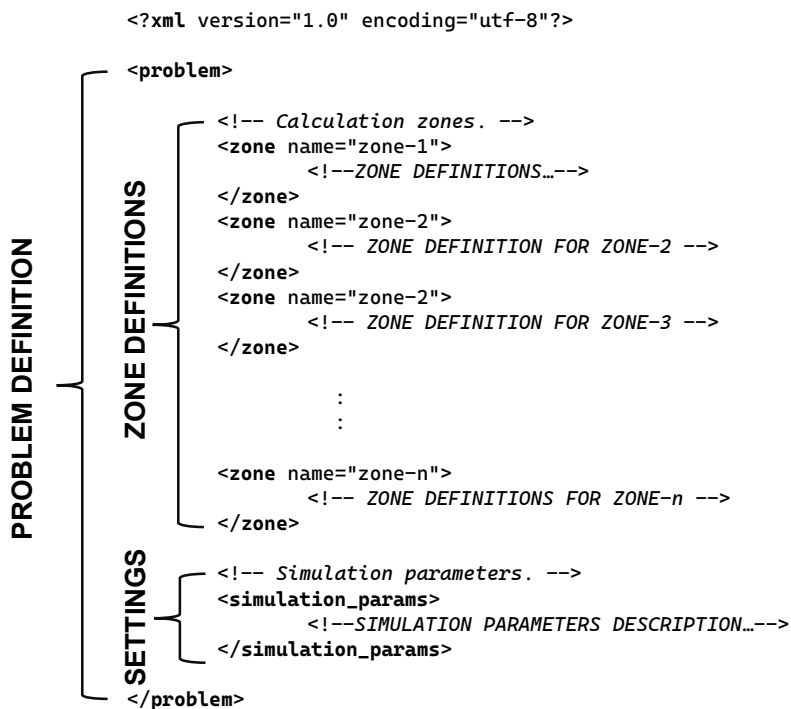
In much the same way, *CNUCTRAN* input uses XML tags to describe the calculation zones, nuclides list, and settings for a burnup calculation.

Section 7: Creating an Input File

Before preparing the input file for *CNUCTRAN*, ensure you have gone through Section 5. *CNUCTRAN* requires two input XML files to work:

- An input XML file specifying the problem zones and computation settings. By default, *CNUCTRAN* assumes the input file as *input.xml*.
- An XML file containing the nuclides library. The user does not need to prepare this XML file, as it is already included in *cnuctran.zip*. The name of the XML file is *chain_endfb71.xml*. The currently available nuclides data library is based on the Evaluated Nuclear Data File (ENDF) version B-VIII.0. To discover more about ENDF: <https://www-nds.iaea.org/exfor/e4explorer.htm>

The structure of the XML input file is given as follows.



There should be at least one zone defined in the input file. Also, there is only one attribute for the `<zone>` tag: the zone's name. All tags within `<zone>` are the child elements of `<zone>`. The specifications of all child tags within the `<zone>` tag are given in Table 2. In addition, Table 3 lists the specifications of all child tags within the `<simulation_params>` tag.

Table 2: Specifications of all child tags within <zone> tag.

Tag Name	Attributes	Description	Child Value
<species> List all nuclides species involved in the calculation.	source	Sets the nuclides library location on the disk. The value should be the location of the <code>chain_endfb71.xml</code> file.	The child value must be a list of nuclide names defined in the nuclides library. A newline, whitespaces or commas should separate the names. If <code>amin</code> and <code>amax</code> are prescribed, <i>CNUCTRAN</i> will ignore the specified list of nuclide names (if any).
	amin	Sets the minimum atomic number, A , to be included in the calculation. Any nuclides with $A < A_{\min}$ will not be included in the calculation. Optional.	
	amax	Sets the maximum atomic number, A , to be included in the calculation. Any nuclides with $A > A_{\max}$ will not be included in the calculation. Optional.	
<initial_concentrations> Specifies the initial concentration of all nuclides involved in the calculation.	source	Sets the initial concentration XML file location on the disk. This file can be the XML output of the previous calculation. Optional.	See <concentration> tag.
	override	This attribute is helpful if the <code>source</code> attribute is prescribed. There are two allowed values: true – all nuclides listed in the initial concentration source that is still not included in the code will be automatically registered. false – otherwise. Optional if <code>source</code> attribute is not set.	
<concentration> Defines a nuclide concentration value.	species	The name of the nuclide species.	n/a.
	value	The value of the concentration. The concentration unit can be anything, including mol, kg, g, unitless (nuclei count), etc. However, the unit must be consistent throughout the input file.	
<reaction_rates> List all neutron-induced reactions associated with various nuclides involved in the calculation. Optional.	n/a	n/a	See <reaction> tag.
<reaction> Defines a reaction rate value with the type of reaction and its target nuclide. Optional.	species	The name of the target nuclide species undergoing the reaction.	n/a
	type	The reaction type. Values can be <code>fission</code> , <code>(n,gamma)</code> , <code>(n,2n)</code> , <code>(n,a)</code> depending on the prescribed nuclides data library.	
	rate	Reaction rate value in the unit of per second. This rate value can be calculated using the neutron flux obtained from neutronic simulations, i.e. Monte Carlo codes and deterministic transport codes.	

Table 3: Specifications of all child tags that appear within the `<simulation_params>` tag.

Tag Name	Child Value
<code><n></code> Sets the approximation order of the calculation. The precision is measured in terms of 10^{-n} . For more info on the approximation order, refer to Section 9.4.	The child value must be an integer greater than 0. A value greater than ten is recommended. However, a higher value will increase the computational cost.
<code><time_step></code> Sets the time step of the calculation in seconds.	The child value must be a float number greater than 0.
<code><precision_digits></code> Sets the minimum number of accurate digits maintained in the arithmetics.	The child value must be an integer greater than 30. A higher value will increase the computational cost. Depending on the problems, it is recommended to set the precision to >40 for the approximation order, $n > 10$. The default value is 45.
<code><output_digits></code> Sets the number of decimal points to be displayed in the output.	The child value must be an integer greater than 0. The default value is 16.
<code><output></code> Sets the location of the output XML file on the disk.	The child value must be a valid target file name of the XML output file. The default value is "output.xml".
<code><verbosity></code> Sets the verbosity level. A higher verbosity means more console output messages will be displayed.	A value of one enables a higher verbosity level. The default verbosity level is 0.

Section 8: Sample Input File

Example #1: A burnup zone consists of nuclides with an atomic number between $A = 100$ and $A = 260$. The problem is a pure decay problem, and we are interested in calculating the final nuclides concentration after 1 million years ($\sim 3.1556926 \times 10^{13}$ seconds). The location of the nuclides data library is in the same working directory with the name *chain_endfb71.xml*. The order of the calculation is $n = 15$, and the number of precision digits is 50. The final nuclide concentrations should be stored in *output.xml*, located within the same working directory. The initial concentration of Np237 is 1 mol.

```
<?xml version="1.0" encoding="utf-8"?>

<problem>

  <zone name="myzone">
    <species source=".\\chain_endfb71.xml" amin="150" amax="250"/>
    <initial_concentrations>
      <concentration species="Np237" value="1.0" />
    </initial_concentrations>
  </zone>

  <simulation_params>
    <n>15</n>
    <time_step>3.1536e+13</time_step>
    <precision_digits>50</precision_digits>
    <output_digits>15</output_digits>
    <verbosity>1</verbosity>
    <output>.\output.xml</output>
  </simulation_params>

</problem>
```

Example #2: A burnup zone consists of nuclides with an atomic number between $A = 100$ and $A = 260$. The problem is a pure decay problem, and we are interested in calculating the final nuclides concentration after 1 million years ($\sim 3.1556926 \times 10^{13}$ seconds). The location of the nuclides data library is in the same working directory with the name *chain_endfb71.xml*. The order of the calculation is $n =$

15, and the number of precision digits is 50. The final nuclide concentrations should be stored in *output.xml*, located within the same working directory. The initial concentration of Np237 is 1 mol. The defined zone is under neutron irradiation, such that Np237 experiences (n, a) and fission reactions with the rates at 0.0001 per sec. and 0.00001 per sec., respectively.

```
<?xml version="1.0" encoding="utf-8"?>

<problem>

  <zone name="myzone">
    <species source=".\\chain_endfb71.xml" amin="150" amax="250"/>
    <initial_concentrations>
      <concentration species="Np237" value="1.0" />
    </initial_concentrations>
    <reaction_rates>
      <reaction species="Np237" type="(n,a)" rate="1E-4" />
      <reaction species="Np237" type="fission" rate="1E-5" />
    </reaction_rates>
  </zone>

  <simulation_params>
    <n>15</n>
    <time_step>3.1536e+13</time_step>
    <precision_digits>50</precision_digits>
    <output_digits>15</output_digits>
    <verbosity>1</verbosity>
    <output>.\output.xml</output>
  </simulation_params>

</problem>
```

Section 9: Performing Calculations & Program Output

The computation can be performed with all input files ready by simply executing *cnuctran.exe*. Remark: The executable requires the two dynamic libraries, i.e. *mpfr.dll* and *mpir.dll*, to be placed next to it, or otherwise, an error is thrown.

At the end of the calculation, *CNUCTRAN* will produce an output XML file containing the final nuclide concentrations. The user can set the name and location of the output XML manually through the input file. It can be done using the **<output>** tag (See Table 3). An example of the output XML file is given as follows:

```
<?xml version="1.0" encoding="utf-8"?>

<output>
  <nuclide_concentrations zone="myzone" amin = "150" amax="250"
    total_nuclides="1489" time_step="3.1536e+13">
    <concentration species="Np237" value="7.237642260179204e-01" />
    <concentration species="Pa233" value="2.493122050609457e-08" />
    <concentration species="U233" value="5.702155930771045e-02" />
    <concentration species="Th229" value="2.635908035643954e-03" />
    <concentration species="Ra225" value="1.464973737813805e-08" />
    <concentration species="Ac225" value="9.832038604770428e-09" />
    <concentration species="Fr221" value="3.345624247467714e-12" />
    <concentration species="At217" value="3.675634802490104e-16" />
    <concentration species="Bi213" value="3.112578215430705e-11" />
    <concentration species="Po213" value="4.673987818740959e-20" />
    <concentration species="Tl209" value="3.304426094035331e-14" />
    <concentration species="Pb209" value="1.332557613880556e-10" />
    <concentration species="Bi209" value="2.165630965799592e-01" />
  </nuclide_concentrations>
</output>
```

Users can also set the verbosity level of the console output via the **<verbosity>** tag. The tag's value can be either 0, 1 or 2. A higher value indicates a higher verbosity level. If the user set the verbosity level to zero, then *CNUCTRAN* will suppress all progress messages. However, it does not hide the error messages. Setting the verbosity level to 1 allows *CNUCTRAN* to print several important messages to the console output. These messages include the time step, the total number of nuclides included in the calculation, and the CPU time of the computation.

Section 10: Theory

10.1 Bateman Equations

Bateman's equation of a burnup problem is given as follows,

$$\frac{dy_i}{dt} = \underbrace{\sum_{j=1}^I b_{ij} \lambda_j y_j + \sum_{k=1}^I f_{ik} \Lambda_{ik} y_k}_{\text{production}} - \underbrace{(\lambda_i + \Lambda_i) y_i}_{\text{removal}} \quad (3)$$

y_i is the atom density of nuclide- i ;

λ_i is the radioactive decay constant of isotope- i causing its removal from the system (the decay constant);

λ_{ij} is the decay constant of nuclide- j causing the production of nuclide- i in the system;

b_{ij} is the decay branching ratio from nuclide- j into nuclide- i ;

f_{ik} is the production yield from nuclide- k into nuclide- i ; and,

Λ_{ij} is the production rate of nuclide- i due to the removal of nuclide- k from the system.

Eq. (3) is a system of I linear differential equations dependent on one another. The solution of Eq. (3) is the concentration of I nuclides. The term on the LHS of the equation is known as the rate of change of nuclide- i concentration. The LHS of the equation is balanced with the terms on the RHS. Here, each term describes the various transmutations of nuclides that lead to the production and removal of nuclide- i within the system.

In particular, the first term on the RHS represents the natural decay of various nuclides that transforms themselves into nuclide- i . In contrast, the second term of the RHS describes the formation of nuclides- i via different transmutation reactions of all nuclides, excluding the natural decay reaction. Finally, the last term on the RHS is a combination of natural decay and mixed transmutation reactions that transform nuclide- i into other nuclides, which removes it from the system.

Alternatively, Eq. (3) can be expressed in its compact matrix form given by,

$$\frac{d\mathbf{y}}{dt} = \mathbf{A}\mathbf{y} \quad (4)$$

where \mathbf{A} is the transmutation matrix such that its elements correspond to all coefficients defined in Eq. (3). In general, \mathbf{A} is a sparse $I \times I$ square matrix, where most of its elements are zeros. On the other hand, \mathbf{y} is a column vector corresponding to the concentration of all I nuclides defined in the calculation. Technically, the norm of \mathbf{A} is in the order of $\sim 10^{21}$ and contains both off-diagonal elements and diagonal elements.

At a glance, obtaining the solution of Eq. (4) seems to be straightforward where the separation of variables is applicable, giving,

$$\mathbf{y} = \exp(\mathbf{A}t) \mathbf{y}_0 \quad (5)$$

Alas, the evaluation of the matrix exponential in Eq. (5) is not an easy job. Naively, one may approximate the matrix exponential by using the Taylor series,

$$\exp(\mathbf{A}t) \approx \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{A}^k t^k \quad (6)$$

However, burnup problems are stiff, such that matrix \mathbf{A} consists of elements with minimal and significant numeric values. Thus, the evaluation of Eq. (6) for a transmutation matrix is impractical and does not always lead to an accurate and reliable result. Several other methods include the Padé approximation of the matrix exponential, which unfortunately suffers the identical drawback as the Taylor series approximation.

Pusa (2016) discovered that the eigenvalues of the matrix \mathbf{A} lie near the negative real axis, thus enabling the approximation of the matrix exponential via the Chebychev approximants. Here, the technique is known as the Chebychev Rational Approximation Method (CRAM), where the matrix exponential approximation is given by

$$\exp(\mathbf{A}t) = \alpha_0 + 2 \operatorname{Re} \left[\sum_{i=1}^{k/2} \alpha_i (\mathbf{A}t - \theta_i \mathbf{I})^{-1} \right] \quad (7)$$

where α_i and θ_i are the complex numbers published by Pusa (2016) for the approximation order $k \in \{16, 32, 48\}$. Here, the computation of Eq. (7) requires solving $k/2$ linear systems. Due to the sparsity pattern of the burnup matrix, the linear systems can be solved efficiently using sparse Gaussian elimination (Pusa & Leppänen, 2013). To date, CRAM is the most efficient and can handle significant burnup problems.

10.2 Discrete Probability Distribution of Transmutation Events

Suppose we select a nucleus during a particular time t . Depending on the environment, the nucleus has various transmutation event options that cause it to transform into another daughter nucleus. It is convenient to index these possible events as $j \in \{1, 2, 3, \dots, J_i\}$. Conveniently, one may assign event-1 as neutron absorption, event-2 as fission, event-3 as radioactive decay, and so on. Also, J_i is the total number of events defined for nuclide i . For simplicity, $j = 0$ is designated as the case when no transmutation is occurring within the interval $(t, t + \Delta t)$.

Now, consider that the parameter Λ_{ij} , characterizing the frequency of each possible event- j is known. Also, $\Lambda_{i,j}$ is a fixed constant known as the transmutation coefficient. For a decay reaction that transforms nuclide i into its daughter nuclide, $\Lambda_{i,j}$ is the decay rate constant or the decay constant. For systems of nuclides under neutron irradiation, $\Lambda_{i,j}$ will be a function of neutron flux. Therefore, it is necessary to assume that the flux is approximately constant over time. The probability for event- j occurring within the interval $(t, t + \Delta t)$ is given by

$$p_{i,j} = 1 - e^{-\Lambda_{i,j}\Delta t} \quad (8)$$

Also, the probability of zero-chance of event- j to occur is the complement of the chance given in Eq. (8),

$$q_{i,j} = e^{-\Lambda_{i,j}\Delta t} \quad (9)$$

The probability, $p_{i,j}$ can be easily derived and theoretically exact.

So far, we have established the probability of a single event. Bear in mind that a parent nucleus is offered with various other possible transmutation events, $j \in \{0, 1, 2, \dots, J_i\}$, and only one of these events will be finally selected. For simplicity, let E_1 be the condition where event- j is occurring, and E_2 be a condition where all events except removal event j are not happening. The joint probability of both E_1 and E_2 are true is given by,

$$P(E_1 \cap E_2) = p_{i,j} \times \prod_{\forall l \neq j} q_{i,l} \quad (10)$$

Substituting Eqs. (8) and (9) into Eq. (10) yield the un-normalized probability distribution,

$$\tilde{\pi}_{i,j} = (1 - e^{-\Lambda_{i,j}\Delta t}) \times \prod_{\forall l \neq j} e^{-\Lambda_{i,l}\Delta t} \quad (11)$$

The probability of no removal, $j = 0$, is given by the product of the zero-chance probability of all possible events,

$$\tilde{\pi}_{i,0} = \prod_{\forall l} e^{-\Lambda_{i,l}\Delta t} \quad (12)$$

Therefore, the normalized joint Poisson distribution, $\pi(\Delta t; \Lambda_{i,1}, \dots, \Lambda_{i,J_i})$, can be formally written as,

$$\pi_{i,j}(\Delta t) = c_i \prod_{l=1}^{J_i} (\delta_{jl} + (-1)^{\delta_{jl}} e^{-\Lambda_{i,l}\Delta t}) \quad (13)$$

where c_i is the normalization constant and δ_{jl} is the Kronecker delta. The value of c_i is given by,

$$c_i = \left(\sum_{j=0}^{J_i} \tilde{\pi}_{i,j} \right)^{-1} \quad (14)$$

10.3 The Probabilistic Method

So far, the establishment of $\pi_{i,j}(\Delta t)$ in Eq. (13) allows us to estimate the probability of various transmutation events experienced by a nucleus. To verify the distribution, we aim to estimate the time evolution of nuclide concentrations using $\pi_{i,j}(\Delta t)$. Generally, the concentration of nuclides i at time $t + \Delta t$ is contributed by two factors. Firstly it is contributed by the un-reacted nuclides i within the interval $(t, t + \Delta t)$. Secondly, it is contributed by the transmutation of other nuclides that occur within the interval $(t, t + \Delta t)$, which leads to the production of nuclide i . Of course, the concentration of nuclide i after the next sub-step, $t + \Delta t$, can be expressed as,

$$y_i(t + \Delta t) = \pi_{i \rightarrow i} y_i(t) + \sum_{\forall l \neq i} \pi_{l \rightarrow i} y_l(t) \quad (15)$$

where $\pi_{i \rightarrow i} \equiv \pi_{i,0}$ is the probability of nuclide i not undergoing a transmutation event; $\pi_{l \rightarrow i}$ is the probability of any nuclides l undergoing a transmutation event that transforms themselves into nuclide i . By definition,

$$\pi_{l \rightarrow i} = \sum_{\forall j \in \mathcal{R}} \pi_j \quad (16)$$

where \mathcal{R} is a set of transmutation events that transform nuclide l into daughter nuclide i . If removal event j is a fission reaction, then π_j in Eq. (16) must be scaled to the fission yield of the daughter nuclide i . Eq. (15) can be systematically written in matrix form via the definition given in Eq. (16). Here, by considering all I nuclides in a depletion problem, then,

$$\begin{pmatrix} y_1(t + \Delta t) \\ y_2(t + \Delta t) \\ \vdots \\ y_I(t + \Delta t) \end{pmatrix} = \begin{pmatrix} \pi_{1,0} & \pi_{2 \rightarrow 1} & \cdots & \pi_{I \rightarrow 1} \\ \pi_{1 \rightarrow 2} & \pi_{2,0} & \cdots & \pi_{I \rightarrow 2} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{1 \rightarrow I} & \pi_{2 \rightarrow I} & \cdots & \pi_{I,0} \end{pmatrix} \begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_I(t) \end{pmatrix} \quad (17)$$

Or simply,

$$\mathbf{y}(t + \Delta t) = \mathbf{P}(\Delta t) \mathbf{y}(t) \quad (18)$$

Notice that the elements of \mathbf{P} are calculated based on the substep interval, Δt . Eq. (18) is a recursion relation where the future nuclide concentrations, $\mathbf{y}(t + \Delta t)$, can be obtained by transforming the nuclide concentrations of the previous time interval via the transfer matrix, \mathbf{P} . Here, \mathbf{P} serves as the transformation operator. The column indices of \mathbf{P} correspond to the various parent nuclides, and the row indices of \mathbf{P} correspond to the various daughter nuclides. Notice that the diagonal elements of square matrix \mathbf{P} are the probabilities of no transmutation events occurring. The elements of \mathbf{P} can be directly calculated using Eq. (15), provided that all reaction rates associated with the nuclides involved are known.

Suppose that the evolution of nuclides concentration is monitored from the initial time, $t = 0$ s, to the final time, $t = T$. Then, a small time interval, Δt , is chosen such that the total number of sub-steps, $l = T/\Delta t \in \mathbb{N}$. Finally, $\mathbf{y}(t)$ can be obtained by iterating Eq. (18) for l times, giving,

$$\mathbf{y}(T) \equiv \mathbf{y}(l \Delta t) = \mathbf{P}^l \mathbf{y}(0) \quad (19)$$

where $\mathbf{y}(0)$ is the initial nuclides concentration at $t = 0$. Here, the smaller the value of Δt , the larger the number of sub-steps, thus, the better the accuracy. However, such a computation with small Δt requires arbitrary precision arithmetic to avoid the floating-point error. The evaluation of sparse matrix exponentiation, \mathbf{P}^l , can be accomplished at an incredible speed via exponentiation by squaring. Evaluating \mathbf{P}^l requires l repeated multiplications of \mathbf{P} . Such a naive approach will incur an unreasonably long CPU time, especially when l is large. Gratefully, the exponentiation by squaring helps to evaluate \mathbf{P}^l with only $\log_2 l$ multiplications (Gordon, 1998). The trick is that \mathbf{P}^l can be efficiently calculated by repeated squaring of \mathbf{P} . Since evaluating \mathbf{P}^l also involves repeated matrix multiplications, using the naive matrix multiplication algorithm will impose unnecessary operations on the zero elements of the sparse matrix, \mathbf{P} . Therefore, this work also implemented a sparse matrix multiplication algorithm proposed by Gustavson (1978).

10.4 Approximation Order

Notice that the probabilistic method requires l self-multiplication of the transfer matrix \mathbf{P} . In the previous section, such a computation can be done at an incredible speed via exponentiation by squaring. To save computational time, the number of substeps is chosen as

$$l = 2^k \quad (20)$$

where k is the total number of self-multiplications of \mathbf{P} required during the implementation of exponentiation by squaring. Our objective is to obtain the suitable value of k such that the substep interval Δt is in the order of 10^{-n} . Here, n is known as the *approximation order*, which helps us to quantify how accurate the approximation is. Therefore, the time step, T , of the calculation can be expressed as,

$$T \approx 2^k \Delta t = 2^k 10^{-n} \quad (21)$$

Finally, solving for k gives the corresponding number of self-multiplications needed so that $\Delta t = 10^{-n}$,

$$k = \lfloor \log_2 10^n T \rfloor \quad (22)$$

Of course, the floor operation is needed because k must be a positive integer number. To compensate for the error due to floor operation, Δt must be corrected,

$$\Delta t = \frac{T}{2^{\lfloor \log_2 10^n T \rfloor}} \quad (23)$$

Section 11: Some Remarks

1. The code user must study the probabilistic method before applying it to real burnup problems. The accuracy of the solution heavily depends on the choice of the approximation order, n , and the number of precision digits. A higher approximation order necessitates more precision digits, thus, eliminating round-off errors during the exponentiation of the transfer matrix \mathbf{P} .
2. *CNUCTRAN* efficiency is optimized through code parallelization. It uses Microsoft Parallel Pattern Library (PPL) to facilitate the parallelization of sparse matrix-matrix multiplication. PPL library improves its efficiency and makes *CNUCTRAN* an applicable code.
3. The probabilistic method is free from any assumptions. Its only sources of error can be due to incorrect nuclides data prescriptions and round-off errors which propagate through the sparse multiplications.
4. *CNUCTRAN* produces high-fidelity final nuclides concentration, and accuracy remains our primary focus. Remarkably, *CNUCTRAN* allows unlimited accuracy levels. *CNUCTRAN* enable the user to adjust the order of the computation and the number of precision digits. However, this costs a significant computing time, which can be easily compensated with the parallelization of a sparse matrix-matrix multiplication algorithm.
5. Based on the verification result of *CNUCTRAN*, it could serve as a reference tool for verifying other burnup solvers.

License

CNUCTRAN is distributed under the MIT software licensing agreement.

Copyright © 2022 M. R. Omar.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

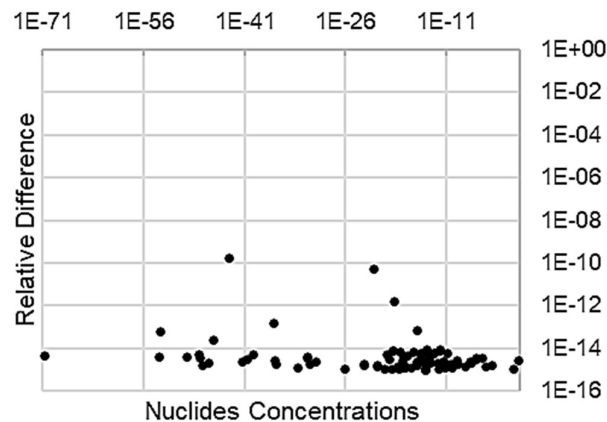
References

- Calvin, O., Schunert, S., & Ganapol, B. (2021). Global error analysis of the Chebyshev rational approximation method. *Annals of Nuclear Energy*, 150, 107828. <https://doi.org/https://doi.org/10.1016/j.anucene.2020.107828>
- Gordon, D. M. (1998). A Survey of Fast Exponentiation Methods. *Journal of Algorithms*, 27(1), 129–146. <https://doi.org/https://doi.org/10.1006/jagm.1997.0913>
- Gustavson, F. G. (1978). Two Fast Algorithms for Sparse Matrices: Multiplication and Permuted Transposition. *ACM Trans. Math. Softw.*, 4(3), 250–269. <https://doi.org/10.1145/355791.355796>
- Pusa, M. (2016). Higher-Order Chebyshev Rational Approximation Method and Application to Burnup Equations. *Nuclear Science and Engineering*, 182(3), 297–318. <https://doi.org/10.13182/NSE15-26>
- Pusa, M., & Leppänen, J. (2013). Solving Linear Systems with Sparse Gaussian Elimination in the Chebyshev Rational Approximation Method. *Nuclear Science and Engineering*, 175(3), 250–258. <https://doi.org/10.13182/NSE12-52>

Appendix A: Comparison between CNUCTRAN and CRAM

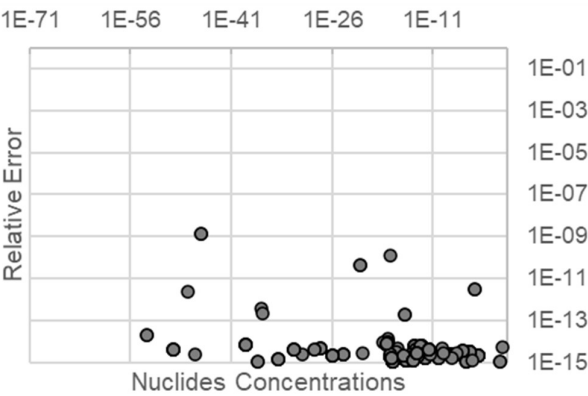
Benchmark #1

Total Nuclides : 1489
 Initial Concentrations : U-235 (0.1mol), U-238 (0.9mol)
 Reaction Rates : U-235
 Fission at 1×10^{-5} per sec.
 (n, α) at 1×10^{-4} per sec.
 (n, γ) at 1×10^{-4} per sec.
 U-238
 Fission at 1×10^{-5} per sec.
 (n, α) at 1×10^{-4} per sec.
 (n, γ) at 1×10^{-4} per sec.
 Time step (s) : 8.64×10^4 sec. (1 day)
 CRAM Order : 48
 Probabilistic Method Order : 20
 No. of precision digits : 90
 Computational time result (s) : 1.6500.988 (CNUCTRAN) @3.2GHz, 8 cores;), 0.360 (CRAM, SciPy) @3.2GHz, 1 core8 cores.
 Result : 224 nuclides with non-zero concentration.
 14 nuclides with negative CRAM concentration and these nuclides have zero cnucltran concentration.
 1251 nuclides with both zero CRAM and cnucltran concentration.
 135 nuclides having a relative error equal to zero (identical up to 15 significant digits; CRAM solution is limited to 15 significant digits).
 57 nuclides having a relative error between 10^{-13} and 10^{-15} .



Benchmark #2

Total Nuclides	: 1600
Initial Concentrations	: U-235 (0.1mol), U-238 (0.9mol)
Reaction Rates	: <u>U-235</u> Fission at 1×10^{-5} per sec. (n, α) at 1×10^{-4} per sec. (n, γ) at 1×10^{-4} per sec. <u>U-238</u> Fission at 1×10^{-5} per sec. (n, α) at 1×10^{-4} per sec. (n, γ) at 1×10^{-4} per sec.
Time step (s)	: 8.64×10^6 sec. (100 days)
CRAM Order	: 48
Probabilistic Method Order	: 15
No. of precision digits	: 60
Computational time (s)	: 0.978 (<i>CNUCTRAN</i>) @3.2GHz, 8 cores; 0.360 (CRAM, SciPy) @3.2GHz.
Result	: 214 nuclides with non-zero concentration. 17 nuclides with negative CRAM concentration having a zero <i>CNUCTRAN</i> concentration. 1266 nuclides having a zero <i>CNUCTRAN</i> concentration. 53 nuclides having relative error equal to zero (identical up to 15 significant digits; CRAM solution is limited to 15 significant digits). 57 nuclides having a relative error between 10^{-13} and 10^{-15} . 8 nuclides having a relative error of $> 10^{-13}$.



Benchmark #2

Total Nuclides : 1639
 Initial Concentrations : Np-237 (1mol)
 Reaction Rates : Np-237
 Pure Decay
 Time step (s) : 1×10^{13} sec.
 CRAM Order : 48
 Probabilistic Method Order : 15
 No. of precision digits : 60
 Computational time (s) : 0.765 (CNUCTRAN), 0.452 (CRAM, SciPy) @3.2GHz, 8 cores.
 Result : 214 nuclides with non-zero concentration.
 149 nuclides having relative error equal to zero (identical up to 15
 significant digits; CRAM solution is limited to 15 significant digits).
 57 nuclides having a relative error between 10^{-13} and 10^{-15} .
 8 nuclides having a relative error of $> 10^{-13}$.

