Development of a new nuclide generation and depletion code using a topological solver based on graph theory

S. Kasselmann¹, S. Scholthaus¹, C. Rössel¹, H.-J. Allelein¹²
¹Forschungszentrum Jülich
52425 Jülich, Germany
phone: +49-2461-616071, s.kasselmann@fz-juelich.de
²Institute for Reactor Safety and Reactor Technology
RWTH-Aachen
52064 Aachen, Germany

Abstract – The problem of calculating the amounts of a coupled nuclide system varying with time especially when exposed to a neutron flux is a well-known problem and has been addressed by a number of computer codes. These codes cover a broad spectrum of applications, are based on comprehensive validation work and are therefore justifiably renowned among their users. However, due to their long development history, they are lacking a modern interface, which impedes a fast and robust internal coupling to other codes applied in the field of nuclear reactor physics. Therefore a project has been initiated to develop a new object-oriented nuclide transmutation code. It comprises an innovative solver based on graph theory, which exploits the topology of nuclide chains. This allows to always deal with the smallest nuclide system for the problem of interest. Highest priority has been given to the existence of a generic software interfaces well as an easy handling by making use of XML files for input and output. In this paper we report on the status of the code development and present first benchmark results, which prove the applicability of the selected approach.

I. INTRODUCTION

Today's reactor simulation models are becoming more and more complex. Effects like Pellet Cladding Interaction (PCI) or High Burn up Structures (HBS) can only be studied with a high spatial resolution. While a typical model in the past had a few hundred burn up regions, a 3D simulation of a whole reactor core nowadays requests several thousand burn up regions. At the same time there is a need to increase the number of nuclides per depletion zone taken into account by a burn up code. This is because an integrated code package simulates the neutronics and the fission product release at the same time. But a lot of nuclides, which are important from the neutronics point of view, are not important with respect to fission product release and vice versa.

The increasing performance of computer systems in principle allows studying these kind of scientific problems. Unfortunately a lot of today's depletion codes are lacking a modern software interface for coupling internally to other codes and no parallelized versions of these codes exist. This is not least due to the fact that these codes share a development history of up to 40 years. Without such an interface the only way to share data between two codes is writing and reading files. Due to the limitations of hard disks this consumes a large amount of time only needed for I/O operations. In fact the authors faced the problem that in many cases reading and writing of files to disk took a reasonable fraction of the total simulation time. In some cases the I/O operations took more time than the actual simulation itself. The only way to solve this problem is the availability of an internal software interface for data communication via the main memory.
Furthermore many codes were developed at a time when computing power and data storage were very limited. Therefore those codes are highly optimized. While this is an advantage with respect to execution time, it often is a problem when it comes to readability and extensibility.

To solve these problems, a project has been initiated to develop a new object-oriented nuclide transmutation code.

II. FEATURES

TNT (Topological Nuclide Transmutation) offers some new features compared to recent burn up codes like ORIGEN [1]. These features can be divided into two groups with respect to physics and with respect to programming techniques applied.

II.A. Physics

One of the new features is the possibility to provide user-specific neutron energy spectra to online generate broad group cross sections for all nuclides and each burn up time step. So instead of having to choose between precompiled sets of one group cross sections for typical reactors, the user can provide the actual neutron energy spectrum derived by a neutronics code. This avoids loss of information in a sense that the burn up code is able to adapt to the group structure used by the neutronics code. Another interesting aspect is the calculation of nuclear heat separately for decay, fission and capture processes. For example the usual approach of taking average capture Q values for a specific fuel and reactor is not applied in TNT. Instead, the recent nuclide vector and distinct Q values for all capture processes available in the data library are used. Fig. 1 shows the nuclear power of a single HTR fuel sphere over time as calculated by TNT. The pebble is passing the core at random radial positions until the discharge burn up is reached after 15 cycles.

![Image](thermal_power.png)

Fig. 1: The contribution of decay, fission and capture processes to the power of a single HTR fuel sphere.

An additional new feature is the usage of energy-dependent fission yields when available in the data library. The ENDF/B-VII.0 data files offer fission yields for 30 heavy metal nuclides. Among them five nuclides with yields for two incident neutron energies and seven nucleides with yields for three different energies. When online cross section condensation is used in TNT, a fission yield for each multi-group flux is derived by interpolation. A third example is the flexibility in adding reactions supported by the code. So far the following processes are implemented:

- Decay: $\alpha$, $\beta^+$, $\beta^-$, EC, IT, SF, n-emission, p-emission (and any combinations, e.g. $\beta^+$/n)
- neutron-induced: $(n,f)$, $(n,\gamma)$, $(n,2n)$, $(n,3n)$, $(n,p)$, $(n,a)$

Further reactions can easily be added to the code. For a new library which includes additional reaction types, just minor changes have to be made in TNT. Therefore this code is capable of being applied to a wider field of applications (e.g. for use in nuclear medicine) as long as basic nuclear data is available for the reactions of interest.

II.B. Advanced Programming Techniques

TNT was developed as a new code from the ground up. Therefore it is free of any legacies. It is an object-oriented (OO) code written in C++. By using an OO approach the code features a high level of readability because it is based on problem-specific data types. TNT makes use of types like Nuclide, NuclideVector, Reaction, ReactionHandler, CrossSectionSet etc. Those classes help to encapsulate and therefore hide specific implementations which are not relevant with respect to physics. So all data handling routines, which make up a large part of such a code, are almost completely separated from the actual algorithms. As a result, the code becomes much clearer and future extensions are easier to implement.

A major drawback of current codes is the lack of internal programming interfaces. As a result, coupling codes often requires to write/read files to/from a hard disk. Even with recent solid-state disks (SSD) this is very time consuming compared to read/write operations of the main memory. Therefore TNT can be used in two different ways. First an executable operated via a graphical user interface is already available. This means TNT can be used as a stand-alone code. Second, if a coupling to other codes is needed, a shared object or .dll file accompanied by a set of generic interfaces will be made available to the user in the near future. So TNT will include a standard Application Programming Interface (API) to couple to other codes without the necessity to recompile the code. The interface consists of a set of C++ header files.
which simply can be included into the user’s code. TNT itself then will ship as a library for different operating systems like Linux or Windows.

For the generation and storage of basic nuclear data, an object oriented data model for the needs of a nuclear simulation code was designed. The library includes decay data, cross sections, independent interactions, neutron flux, scattering matrices, fission spectra etc. for each nuclide. It is created fully automatically for all nuclides available e.g. from ENDF/B-VII.0. The library generator includes NJOY input file creation, NJOY data processing, the reading of NJOY output files and the calculation of cross section sets as well as scattering matrices for different temperatures and background cross sections [2]. Once generated, the data library is serialized into a binary file. Therefore it can be read into the main memory in a short amount of time.

For the calculations well established libraries like boost [3] and Eigen [4] are applied. The boost graph library (BGL) is used for implementing the graph of nuclide chains while Eigen is included to calculate the matrix exponential described in section III. Both libraries make extensive use of template metaprogramming, which allows implementing computational algorithms on a high level of abstraction. At the same time it is easy to use, as template classes just need to be included into the own project code.

III. APPLIED MATHEMATICAL METHODS

A major improvement of TNT compared to former code systems is the use of graph theory for the analysis of nuclide chains. A graph consists of vertices and edges between them. With respect to the depletion equations, the vertices in TNT represent nuclides, while the edges represent different transmutation channels (e.g. decay, fission or neutron capture). The mathematical effort of solving a depletion problem can then be minimized by traversing the tree of nuclide chains in an optimized way. Due to the fact that graph theory is an important research field in computer science and mathematics, many optimized algorithms for several problems already exist.

The basic transmutation equation for a nuclide with amount \( N_i \) in a set of \( n \) nuclides can be written as shown in (eq. 1). The first expression describes the production of nuclide \( N_i \) by decay of precursor nuclides \( N_j \). In this case, \( \lambda_j \) is the decay constant of the precursor and \( l_{ij} \) the fraction of nuclide \( j \) decaying into nuclide \( i \). The second expression describes the production by neutron-induced processes of precursors \( N_k \) interacting with the neutron flux \( \Phi_k \). The sum over \( f \) adds up the rates for all broad neutron energy groups \( m \). Here \( \sigma_{ij,k,f} \) is the absorption cross section of precursor nuclide \( k \) in energy group \( f \). \( y_{i,k,f} \) is the energy-dependent (group function) fraction of nuclide \( k \) producing nuclide \( i \) (yield).

The last part of the equation describes the change of \( N_i \) by its own decay (\( \Lambda_i \)) and by all reactions (\( \sigma_{i,f}^{\text{tot}} \)) leading to other nuclides due to a group neutron flux.

\[
\frac{dN_i}{dt} = \sum_{j=1}^{n} l_{ij}\lambda_j N_j + \sum_{k=1}^{n} N_k \sum_{f=1}^{m} \phi_k \sigma_{k,f}^{\text{tot}} N_i
- \left( \lambda_i + \sum_{f=1}^{m} \phi_f \sigma_{i,f}^{\text{tot}} \right) N_i \quad \text{(eq. 1)}
\]

Applying basic transformations, this system of differential equations can be written as shown in eq. 2:

\[
\begin{pmatrix}
N(t)_1 \\
\vdots \\
N(t)_n
\end{pmatrix}
= \begin{pmatrix}
a_{11} & \cdots & a_{1n} \\
\vdots & \ddots & \vdots \\
a_{n1} & \cdots & a_{nn}
\end{pmatrix}
\begin{pmatrix}
N(t)_1 \\
\vdots \\
N(t)_n
\end{pmatrix}
= A \begin{pmatrix}
N(t)_1 \\
\vdots \\
N(t)_n
\end{pmatrix}
\]

The solution for this equation is well known and given in eq. 3:

\[
\begin{pmatrix}
N(t)_1 \\
\vdots \\
N(t)_n
\end{pmatrix}
= \exp \left( \begin{pmatrix}
a_{11} & \cdots & a_{1n} \\
\vdots & \ddots & \vdots \\
a_{n1} & \cdots & a_{nn}
\end{pmatrix} \cdot t \right)
\begin{pmatrix}
N(0)_1 \\
\vdots \\
N(0)_n
\end{pmatrix}
= A \begin{pmatrix}
N(0)_1 \\
\vdots \\
N(0)_n
\end{pmatrix}
\]

To solve this equation system the matrix exponential \( \exp(M) \) with \( |M|=|A|\cdot t \) has to be calculated. It is defined by the Taylor series expansion as shown in eq. 4.

\[
\exp(|M|) = |1| + |M| + \frac{|M|^2}{2!} + \cdots = \sum_{k=0}^{\infty} \frac{|M|^k}{k!} \quad \text{(eq. 4)}
\]

This reduces the differential equation to a computation of the transfer matrix \( |A| \) and the corresponding matrix exponential. Unfortunately, due to the large spread of half-lives, the \( a_{mn} \) differ by several orders of magnitude. Therefore the matrix products become numerically unstable.

To solve this problem, a two-step strategy is applied. In a first step the nuclides are divided into short- and long-lived ones. The threshold defining short- and long-lived nuclides can be set by the user. Typically, a nuclide is considered to be short-lived if the total removal rate reduces the nuclide amount by three orders of magnitude in a given time step. This corresponds to about 10 half-lives. As short-lived nuclides are in equilibrium with long-lived nuclides at the end of each time step, initial amounts of short-lived nuclides are ‘moved’ to long-lived nuclides they decay into. This is done by applying the Bateman equation (see eq. 5) for all possible paths between two long-lived nuclides. Here it is
important to correctly take cyclic chains into account.

\[ N_i(t) = N_i(0) \cdot e^{-d_i t} + \sum_{k=1}^{i-1} N_k(0) \left( \sum_{m=k}^{i-1} \frac{e^{-d_j t} - e^{-d_k t}}{(d_i - d_j)(d_i - d_k)} \prod_{n=m+1}^{i} a_{n+1,n} d_{n} - d_{j} \right) \]  

(eq. 5)

The Bateman equation is a differential equation for a chain of nuclides with non-zero initial amounts \( N_i(0) \). The removal rates \( d_i = -a_{i,i} \) include neutron-induced processes as well as radioactive decay. The factor \( a_{i,j} \) is the branching ratio between nuclide \( i \) and \( j \). The sum over \( k \) adds up all precursor contributions to nuclide \( N_i \). For a detailed derivation of the Bateman equation refer to [5,6]. As a result, effective rates have been obtained between all long-lived nuclides. In other words, the graph has been ‘shrunk’ to only consisting of long-lived nuclides with corrected amounts from initial short-lived nuclides. These amounts are then ‘projected’ to the beginning of the time step. Starting from these amounts, the evaluation of the matrix exponential \( |M| \) is performed which contains the effective rate transfer matrix. After this step the amounts for long-lived nuclides can easily be obtained by multiplying the matrix exponential with the initial nuclide vector.

In the second step the amounts of short-lived nuclides have to be calculated. Because the amounts depend on what is ‘delivered’ by their long-lived precursors, the short-lived nuclides are in equilibrium with the long-lived precursors. The precursor amounts are now fixed in this calculation. This leads to a linear equation system solved by the Jacobi method. By applying a breadth-first search pattern to the graph of nuclides starting from every long-lived nuclide, the necessary calculation steps can be minimized.

IV. VERIFICATION AND FIRST RESULTS

Introducing new codes to nuclear applications requires extensive work on validation and verification. To assure the quality of the code and the correctness of the results, TNT is subject to unit and regression tests. For the first tests of TNT, benchmark calculations with the ORIGEN 2.2 code were performed as well as comparisons to analytically solvable problems. In this paper we want to present first results for three different nuclide systems selected by increasing complexity. For all calculations the \textit{decay.lib} and \textit{thermal.lib} libraries which are part of the official ORIGEN 2.2 code package were used. To achieve comparable results, the internal modification of cross section libraries read by ORIGEN was switched off. Therefore the results presented in this paper do not reflect a real problem from reactor physics point of view. However they are well suited to do code-to-code comparisons. The following calculations were performed:

- The decay of 100 mole U-235 for a time of one half-life (\( T_{1/2} = 0.704 \cdot 10^9 \text{y} \))
- The activation of 100 mole H-001 with a flux of \( 10^{15} \text{n/cm}^2/\text{s} \) for \( 10 \text{y} \)
- The fission of 100 mole U-235 with a flux of \( 10^{13} \text{n/cm}^2/\text{s} \) for \( 1 \text{y} \) and up to \( 5 \text{y} \)

IV.A. Decay of U-235

As a first example the decay chain of U-235 without spontaneous fission was simulated. This decay chain contains 15 nuclides. It features basic phenomena (e.g. branching into different daughter nuclides) without reaching the full complexity of a burn up calculation. The resulting graph for the TNT calculations is shown in Fig. 2. Here colors indicate the different decay processes involved while the edge thickness indicates different rates (log. scale).

![Graph of an U-235 decay as used by TNT.](image)

Fig. 2: Graph of an U-235 decay as used by TNT.

As can be seen in Tab. 1, the results of ORIGEN and TNT are in a very good agreement.

\[ \text{Table 1: Comparison of amounts for selected nuclides after the decay of initially 100 mole U-235.} \]

<table>
<thead>
<tr>
<th>Nuclide</th>
<th>ORIGEN 2.2 (mole)</th>
<th>TNT 1.0 (mole)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-235</td>
<td>49.99</td>
<td>50.01</td>
</tr>
<tr>
<td>Pa-231</td>
<td>2.327E-03</td>
<td>2.328E-03</td>
</tr>
<tr>
<td>Ra-223</td>
<td>2.224E-09</td>
<td>2.225E-09</td>
</tr>
<tr>
<td>Po-211</td>
<td>3.529E-18</td>
<td>3.531E-18</td>
</tr>
<tr>
<td>Pb-207</td>
<td>50.00</td>
<td>49.99</td>
</tr>
<tr>
<td>He-004</td>
<td>350.1</td>
<td>349.9</td>
</tr>
</tbody>
</table>
IV. B. Activation of H-001

As a second example the activation chain of H-001 was calculated for a constant flux of $10^{15} \text{n/cm}^2/\text{s}$ and an irradiation time of 10 years. As can be seen, six nuclides are involved in this calculation. The irradiation of hydrogen leads to deuterium and tritium. The latter one then decays into He-003. This is one of the smallest nuclide systems where a cycle between two nuclides can be observed: For the He-003 two reactions take place, a (n,p)- and a (n,γ)-reaction. Those cycles are correctly taken into account during the calculation. The nuclide graph shown in Fig. 3 is based on ENDF/B-VII.0 data, as the recent ORIGEN library still contains the capture reaction from H-003 to H-004. Probably due to the very short half-life of H-004 ($\approx 10^{-22}\text{s}$) this reaction is not part of the ENDF/B-VII.0 library anymore. Nevertheless, the calculations are based on the ORIGEN library for comparison reasons.

Fig. 3: Graph of the H-001 activation as used by TNT (ENDF/B-VII.0 library).

As can be seen in Tab. 2, the results of ORIGEN and TNT are again in a very good agreement.

Table 2: Activation results for the irradiation of initially 100 mole H-001.

<table>
<thead>
<tr>
<th>Nuclide</th>
<th>ORIGEN 2.2 (mole)</th>
<th>TNT 1.0 (mole)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H-001</td>
<td>90.05</td>
<td>90.06</td>
</tr>
<tr>
<td>H-002</td>
<td>9.946</td>
<td>9.940</td>
</tr>
<tr>
<td>H-003</td>
<td>8.462E-04</td>
<td>8.451E-04</td>
</tr>
<tr>
<td>H-004</td>
<td>7.325E-21</td>
<td>7.315E-21</td>
</tr>
<tr>
<td>He-003</td>
<td>2.826E-07</td>
<td>2.823E-07</td>
</tr>
<tr>
<td>He-004</td>
<td>5.406E-10</td>
<td>5.377E-10</td>
</tr>
</tbody>
</table>

IV. C. Irradiation of U-235

As a last example, results of an U-235 irradiation by a neutron flux is presented. A part of the graph set up and used by TNT is shown in Fig. 4. To keep the graph simple, all capture reactions were switched off while the picture was created.

Fig. 4: Part of the graph of an U-235 irradiation (fission and decay only) as used by TNT

However, during the calculation all reactions were taken into account. As can be seen in Tab. 3 and Fig. 5 the results of ORIGEN and TNT are again in a very good agreement.

Table 3: Comparison of selected results for the irradiation of initially 100 mole U-235.

<table>
<thead>
<tr>
<th>Nuclide</th>
<th>ORIGEN 2.2 (mole)</th>
<th>TNT 1.0 (mole)</th>
</tr>
</thead>
<tbody>
<tr>
<td>As-081</td>
<td>4.346E-08</td>
<td>4.346E-08</td>
</tr>
<tr>
<td>Cd-119</td>
<td>2.127E-08</td>
<td>2.127E-08</td>
</tr>
<tr>
<td>Cs-141</td>
<td>7.488E-07</td>
<td>7.488E-07</td>
</tr>
<tr>
<td>Ru-224</td>
<td>4.542E-14</td>
<td>4.593E-14</td>
</tr>
<tr>
<td>Th-232</td>
<td>4.287E-08</td>
<td>4.281E-08</td>
</tr>
<tr>
<td>U-233</td>
<td>2.676E-10</td>
<td>2.672E-10</td>
</tr>
<tr>
<td>U-235</td>
<td>80.67</td>
<td>80.68</td>
</tr>
<tr>
<td>U-236</td>
<td>2.798</td>
<td>2.796</td>
</tr>
<tr>
<td>U-238</td>
<td>7.976E-06</td>
<td>7.975E-06</td>
</tr>
</tbody>
</table>

Minor differences in the results can be explained by the fact that TNT uses double precision floating point numbers throughout the code while ORIGEN uses double precision numbers only in some parts of the calculation.

Fig. 5: Comparison of time dependent nuclide amounts for selected nuclides during U-235 irradiation.
V. PERFORMANCE

Besides the applicability and readability of a code, performance is an important issue. Here TNT can bring to bear the advantage of having a topological solver to the full extend. The code reads the initial nuclide vector defined by the user and analyzes all possible reaction channels starting from the given nuclides. This results in a minimum graph of nuclides, which then is used as a basis for the solver algorithms. This means TNT, in contrast to some other codes, only uses the nuclides of interest and not automatically all nuclides available on the data library. For example, to calculate the decay of U-235 (without SF) the amounts of only 15 nuclides have to be calculated as shown in Fig. 2. In addition, the way of calculating the amounts of short-lived nuclides within a chain was optimized. As TNT knows about the topology of the complete nuclide system, the solver can walk along the chain in a way that leads to only one iteration within the Jacobi solver. ORIGEN 2.2 for example is ordering nuclide data according to their ZA number, which leads to a higher number of iterations in this case.

To quantify the performance, several benchmark calculations have been performed on a normal desktop PC (Intel Core i7-2600, 3.40GHz) with a Linux operating system. Four scenarios with an increasing number of nuclides were defined as listed in Tab. 4. The execution time was measured using the time command and comparing the real time results.

Table 4: Scenarios for the benchmark between ORIGEN 2.2 and TNT 1.0.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Number of nuclides involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>H-001 irradiation</td>
<td>6</td>
</tr>
<tr>
<td>S2</td>
<td>U-235 decay only</td>
<td>15</td>
</tr>
<tr>
<td>S3</td>
<td>U-235 decay + SF</td>
<td>334</td>
</tr>
<tr>
<td>S4</td>
<td>U-235 irradiation</td>
<td>908</td>
</tr>
</tbody>
</table>

In this benchmark, ORIGEN 2.2 and TNT 1.0 were compared while reading the same data library for each new burn up time step. This simulates the application mode when burn up codes are coupled to neutronics codes via input/output files. The results are shown in Fig. 6.

As ORIGEN 2.2 always performs the calculation for all nuclides on the library, the execution time is the same for all benchmark cases. Therefore it appears just once (S0) in Fig. 6. As can be seen, for a large nuclide system (S4) TNT is somewhat slower than ORIGEN. This is mainly due to the object-oriented approach and because the nuclide graph is rebuild for each time step. Recent optimizations however have shown that TNT is capable of doing about 12 calculations per second in scenario S4.

Due to the graph optimization, TNT becomes faster if the number of nuclides in the system is reduced (S1, S2, S3). Therefore TNT may especially be interesting for nuclear fusion research for example. It is also obvious, that there is no performance gain between S2 and S1. This is because the reading of the data library for each time step is the dominant factor here.

As mentioned before TNT can take different neutron energy spectra for each burn up time step. It then will read the data library (e.g. with microscopic cross sections) only once. In this mode all cross sections are internally condensed to a user-defined energy group structure and then applied in the burn up equations. As I/O workload is minimized in this case, the performance of TNT is much higher. Systematic studies are work in progress. In first calculations 4200 calc./s were reached for example in case of H-001 activation. Compared to about 35 calc./s in the benchmark case this is a speedup factor of 120. Of course also codes like ORIGEN 2.2 are much faster than measured in this benchmark if the data library is read only once. Unfortunately, due to the lack of an internal interface, ORIGEN 2.2 can not be used in this mode if the user wants to provide problem-specific cross section sets for each burn up time step.

Fig. 6: Number of calculations per second (Intel Core i7-2600, 3.40GHz) for different scenarios (see Tab. 4) performed with ORIGEN 2.2 (red) and TNT 1.0 (green) when reading a pre-compiled data library for each burn up time step.
VI. CONCLUSIONS

With TNT a new generation burn up code was developed. The extensive use of modern programming techniques and the availability of a generic API make the code very flexible, especially when coupled to other codes. From a physics point of view, the online condensation of macroscopic cross sections for given neutron energy spectra as well as the usage of energy-dependent fission yields are interesting features. At the moment TNT is a beta version for internal use only. The verification work is ongoing and a documentation report is being written. All results obtained up to now are in excellent agreement with codes like ORIGEN or FISPACT [7].

Nevertheless, further extensions of TNT are planned already. One major topic is the extension of TNT and the underlying library to be usable in the field of nuclear medicine (radioisotope production). Later a radio toxicity code module will be added. The next version of the library generator code will bring support for other data libraries (e.g. JEFF or JENDL). With respect to optimization, new ideas to speed up the code are to fix specific nuclides (e.g. for fast calculations of I/Xe transients) or to leave out nuclides at all. In the latter case effective rates between the remaining nuclides have to be derived of course. Due to the modern code structure TNT is already prepared to be parallelized using OpenMP or MPI. First steps towards a parallel TNT version are currently being investigated.

REFERENCES


Oak Ridge National Laboratory, ORNL-TM-361, pp. 1-18, 1962