ORILL Code

Release 1.0

Franck CHANTELOUP

Oct 30, 2018

Contents

ORILL Code is a 1D transmutation, fuel depletion (burn-up) and radiological protection code useful for nuclear research reactors design and operational safety. ORILL has been cooked at the **Nuclear Safety Unit** of the Institut Laue-Langevin (ILL), Grenoble, France.

Actually, there is a need for such a pocket tool, which is not addressed by available heavy codes (FISPACT, CINDER, SERPENT, ORIGEN, VESTA, MURE).

The name "ORILL" comes from "ILL" and the prefix "OR" is a tribute to the excellent ORIP_XXI freeware from *E.G.Romanov, V.A.Tarasov, F.Z.Vahetov, Research Institute of Atomic Reactors, Dimitrovgrad, Russia.*

CHAPTER 1

Authors

Franck CHANTELOUP, **Nuclear Safety Unit** @ Institute Laue-Langevin, Grenoble, France.

CHAPTER 2

Nuclear Data

2.1 Libraries

ORILL Code data are parsed from evaluated libraries in ENDF-6 or ACE format by using the PyNE toolkit. Parsing features are not provided with ORILL public version. However, data provided with public ORILL have been selected from several libraries in order to cover a wide range of situations:

- Nuclide set consists of 3820 nuclides from the US Evaluated Nuclear Data ENDF/B-VII.1 decay sub-library (**only childs with branching ratios > 0.1%** are selected).
- Incident neutron data cover 816 nuclides cross sections and multiplicities from the European Activation File (EAF 2010) at 293.6K: (**only (n,2n), (n,3n), (n,fission), (n,gamma), (n,p), (n,d), (n,t), (n,3He), (n,alpha)** reactions are selected).

- Direct (independent) induced fission yields of 83 nuclides are from TENDL 2011 TALYS nuclear model code system (**only products with yields > 0.01%** are selected).
- Effective dose coefficients for inhalation (adult, public) includes 1975 nuclides from EAF 2010 and ICRP 72.
- Fluence to effective dose coefficients (Hp, rotational ROT) are from ICRP 74 (we consider ICRP 116 values questionable).

The choice of EAF 2010 and TENDL 2011 instead of ENDF/B-VII.1 was made to enlarge the data set in terms of available cross sections and fission products. Moreover, independent fission yields are sometimes difficult to measure and the TALYS nuclear model has a good agreement with experimental data.

2.2 Radiological protection

The effective dose coefficients for inhalation (Sv/Bq) (adult, public) from EAF 2010 comprises ICRP 72 data and calculated coefficients by the EAF project. Be careful that EAF 2010 coefficients for inter gases (Ar, Kr, Xe) have been converted from dose rate unit per air concentration to Sv/Bq, based on the metabolic behavior of Yttrium (EAF 2010 biological, clearance and transport libraries, *L. W. Packer and J-Ch. Sublet, EURATOM/CCFE Fusion Association*). These coefficient are therefore much more conservative than in the ICRP 72.

2.3 Warning

ORILL Code public version is provided for educational purpose, with limited nuclear data. Additional data is available for individuals or institutions that would like to collaborate.

CHAPTER 3

ORILL Code

3.1 Purpose

ORILL Code is a 1D transmutation, fuel depletion (burn-up), and radiological protection code useful for nuclear research reactors design and operational safety. ORILL has been cooked at the **Nuclear Safety Unit** of the Institut Laue Langevin (ILL), Grenoble, France.

3.2 General Features

Small Code and nuclear data are small (only 4 Mo).

Fast Vectorization and parallelization associated with Python/Scipy is transparently available.

User friendly ORILL is designed to address practical questions about isotopes inventories, decay heat, dose rate, photons spectrum. It is usable by everyone for 1D point-flux calculation at the speed of light.

3.3 Prerequisites

ORILL needs only Python 2.7 or later, with Scipy and standard modules installed (re, io, yaml, time). Python is available on all platforms (WinPhyton, MiniConda, Debian/Ubuntu, Intel Distribution,...).

3.4 Installation

Uncompress ORILL files in a directory where a Python 2.7+ shell is available (with Scipy). In the Python shell, type:

```
>>> from ORILL import ORILL_CODE
>>> ORILL_CODE('ORILL_TEST.yml')
```

The command input file 'ORILL_TEST.yml' located in (/input) subdirectory is processed and the corresponding output files are created in (/output) subdirectory. Have a look at ORILL_TEST.yml file to understand the file syntax. It is recommended to edit this file with Notepad++ with the YAML markup language.

3.5 Contribute

If you want to contribute, let us know.We have a mailing list located at: orill_code@googlegroups.com

3.6 Support

If you are having issues, please let us know. We have a mailing list located at: orill_code@googlegroups.com

3.7 License

CHAPTER 4

Results

Based on neutron flux at each time period, ORILL Code compute:

- nuclides inventory after irradiation (Atom, Becquerel, Gram)
- fission rate and power (Fission/s, Watt)
- decay power (Alpha, Beta, Gamma) (Watt)
- effective dose per inhalation (Sv)
- effective dose rate (decay) at 1 m distance (mSv/h)
- gamma spectrum (decay) with MicroShield®V5 formatted input file for further shielding studies

ORILL Code compute sets of nuclides and/or elements. Elements are automatically converted into nuclides according IUPAC natural abundance.

CHAPTER 5

Technical features

5.1 Neutron flux spectrum

The neutron flux is specially formatted in:

- 1-point **thermal** flux at **0.0253 eV**
- 2-points **fast** flux at **0.5 MeV** and **14 MeV**

For thermal and cold neutrons transmutation, only the thermal value is useful. For fuel depletion, only the relative values of thermal and fast neutrons are useful (because of threshold reactions). A multiplication factor can be used to adjust the fission power to the right value, without changing the flux shape.

5.2 Evolution algorithm

The choice of the algorithm (Bateman/TTA, Matrix Exponential, Runge-Kutta, Adams, BDF...) is a tricky problem in case of loops and stiffness in nuclides chains. After some trials, only three methods were selected.

5.2.1 BDF

ORILL can use the old LLNL Fortran77 VODE code (SUNDIALS) with Backward Differentiation Formula (BDF) as provided with Python/Scipy to solve the stiff differential equations. VODE/BDF is faster than MMPA on old CPU and the difference with MMPA is generally < 2%. This method is generally stable, but instabilities can occur sometimes.

5.2.2 MMPA

Mini-Max Polynomial Approximation (MMPA) of matrix exponential can be used in ORILL. It is extremely simple to code on sparse matrix, and has several advantages over Chebyshev Rational Approximation Method (CRAM). MMPA method is very stable. It is described by *Yosuke Kawamoto and al., Numerical solution of matrix exponential in burn-up equation using mini-max polynomial approximation, Annals of Nuclear Energy, Volume 80, 2015, Pages 219-224.*

5.2.3 DIAG

The DIAG method uses diagonalization of the evolution matrix in **C** field, which is a more general analytical method than the classic Bateman / TTA. If diagonalization is possible, each eigenvalue is the pseudo decay constant of the corresponding eigenvector. A decay matrix is always diagonalizable because there is no loop and it can be rearranged in a lower triangular matrix with its real eigenvalues on the

diagonal. Unfortunately, large burn-up matrix with loops in transmutation chains due to neutron absorbtion are not always diagonalizable in **C** field with enough accuracy. If diagonalization is impossible, the DIAG method will fail.

5.3 Inside ORILL engine

Nuclide names are translated in ZAAAm numbers, like 'Am-242m' in 952421, or 'He-4' in 20040. ORILL uses Python dictionaries and Numpy arrays to process data before hard computation. A nuclide set is defined at the beginning: it depends on the problem and on the computation depth into chains (set by the user). In the set, nuclides are sorted on an index [0,1,. . . ,N-1], from which a sparse evolution matrix is build. The sparse matrix is processed with previously described numerical methods. At the end of each time step, the desired values are computed and the nuclide list is build back from the index.

CHAPTER 6

Usage and download

6.1 Download

ORILL is located at Github

If you want to contribute, let us know.We have a mailing list located at: orill_code@googlegroups.com

6.2 Run test

Uncompress ORILL files in a directory where a Python 2.7+ shell is available (with Scipy). In the Python shell, type:

```
>>> from ORILL import ORILL_CODE
>>> ORILL_CODE('ORILL_TEST.yml')
```

The command input file 'ORILL_TEST.yml' located in (/input) subdirectory is processed and the corresponding output files are created in (/output) subdirectory.

6.3 Input file syntax

Have a look at 'ORILL_TEST.yml' file to understand the file syntax. It is recommended to edit this file with Notepad++ with the YAML markup language.

```
Flux: [1.0e+14, 1.0e+14, 1.0e+14]

# 3-points flux at [0.0253 eV, 0.5 MeV, 14.0 MeV]␣
↪(neutron/cm2/s)
# Fuel depletion: 3-points flux values, adjust the␣
↪flux multiplier in "Periods:" to adequate␣
↪Fission Power
# Transmutation with cold neutrons only: adjust␣
↪thermal flux with wavelength(cold)/wavelenght(T)␣
↪factor [adjusted_thermal_flux, 0.0, 0.0]

Nuclides:
 Cl: 0.1
 Na: 0.1
 U-235: 0.1

# Valide nuclides names are: 'U','PU239','94Pu238',
↪'PU-240','Bi-194m','BI194M2','83-BI-194m2',...
# Names are not case sensitive, metastable states␣
↪are m = m1, m2, m3
# Elements like 'U','Al','Ni' are converted into␣
↪isotopes (according natural abundance mass␣
↪fraction)

Mass: 10.0
Unit: 'w'

# Mass (total, grams) used only if Unit = 'w'
```

(continues on next page)

(continued from previous page)

```
# Unit for nuclides set: mass fraction('w'), grams(
↪'g'), mol('mol'), atoms('atm'), becquerels('bq')

Periods: [['Period1', [3600], 1.0, 'MMPA'], [
↪'Period2', [1.0e+3,1.0e+4,1.0e+5], 0.0, 'DIAG']]

# [['Identifier', [Duration(s),...], Flux␣
↪multiplyer, 'Method'],...]
# 'Identifier' is used to identifiy output files␣
↪on each period
# [Duration(s),...] are durations steps to compute␣
↪in each period
# Flux multiplyer: 1.0 (full flux), 0.0 (zero flux,
↪ pure decay), 0.5 (50% flux)
# 'Method' are: 'MMPA', 'BDF' or 'DIAG'

Depth: 6

# Computation depth into chains: 6 to 12␣
↪recommanded (more depth = more nuclides = more␣
↪computation time)
```