UNIVERSITY OF BIRMINGHAM

Department of Metallurgy & Materials

Activity V2 Manual

Contents

Chapter 1

Overview

1.1 Reason for the Program

The Activity program calculates how radioactive a target becomes after being irradiated by high energy ions. It uses the TENDL-2019[1] database that contains cross-section data for protons and deuterons, and the JEFF-3.3 Radioactive Decay Data File[2].

1.2 Program Requirements

The user must provide a exyz file from the SRIM[3] ion transport program and a breakdown of the composition of the target, as well as other irradiation parameters (beam projectile, beam duration, beam area, target thickness, target composition, simulation end time etc).

Chapter 2

Installation

The program needs Python 3 installed in order to run. At the time of writing, it has only been developed to run on a Linux operating system, but it shouldn't require much adjusting to run on a Windows computer too.

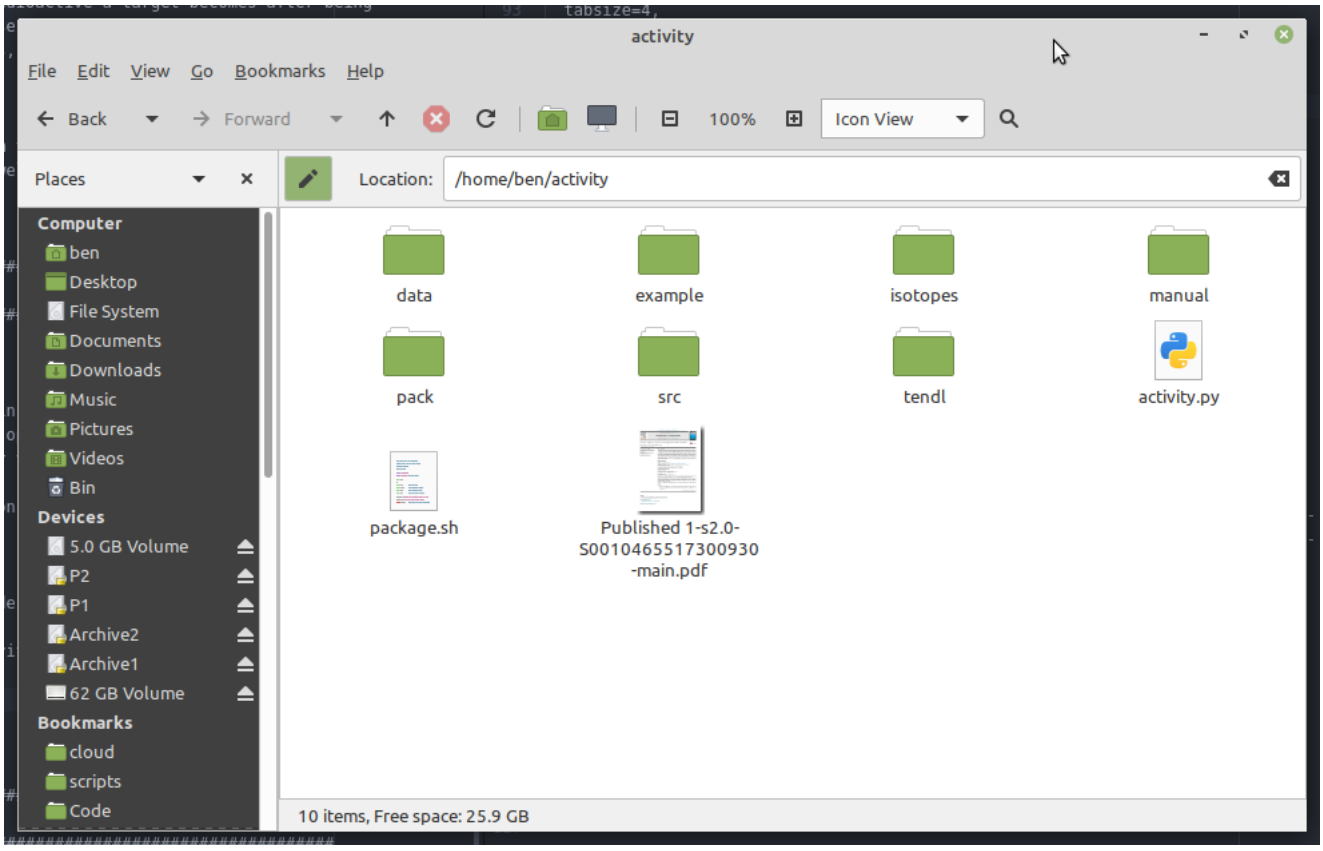
Download and install the latest version of Python 3:

https://www.python.org/downloads

The latest version of the activity code with data files must also be downloaded:

https://github.com/BenPalmer1983/activity_v2

For example, I have installed to /home/ben/activity

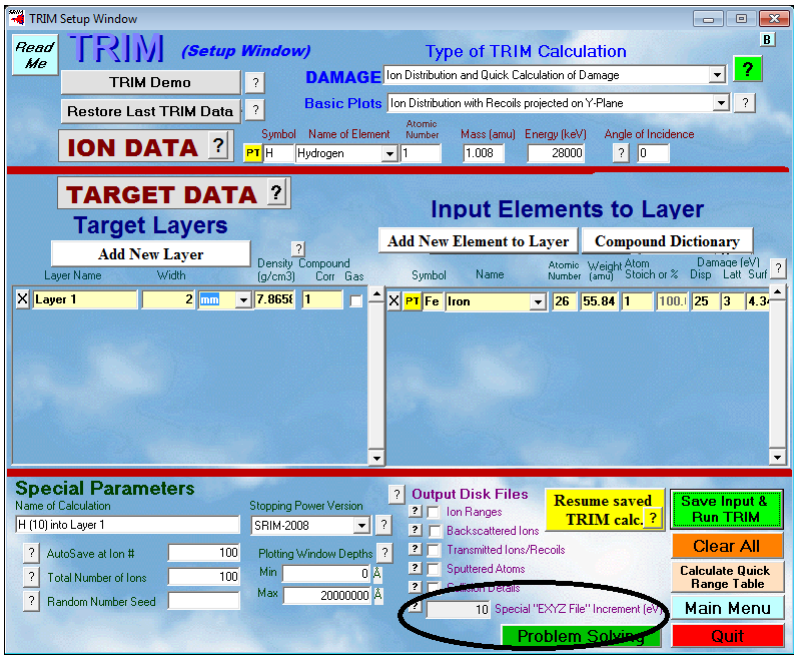


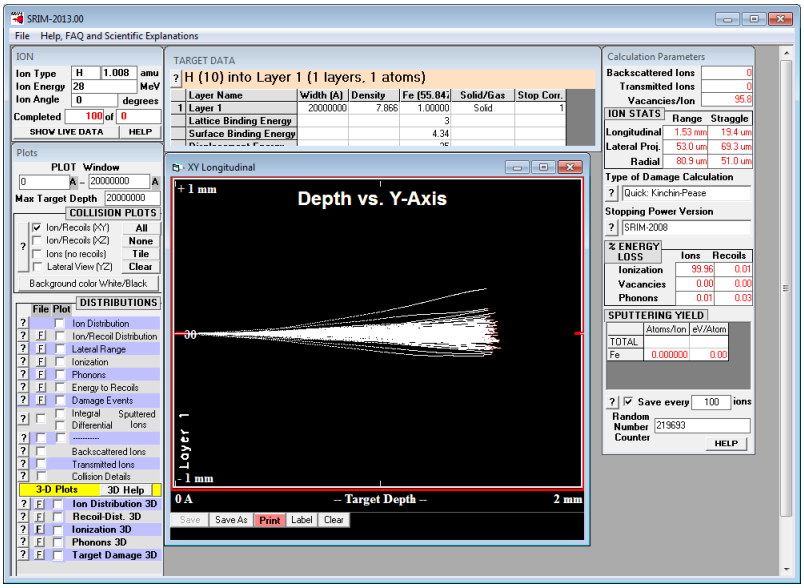
Chapter 3

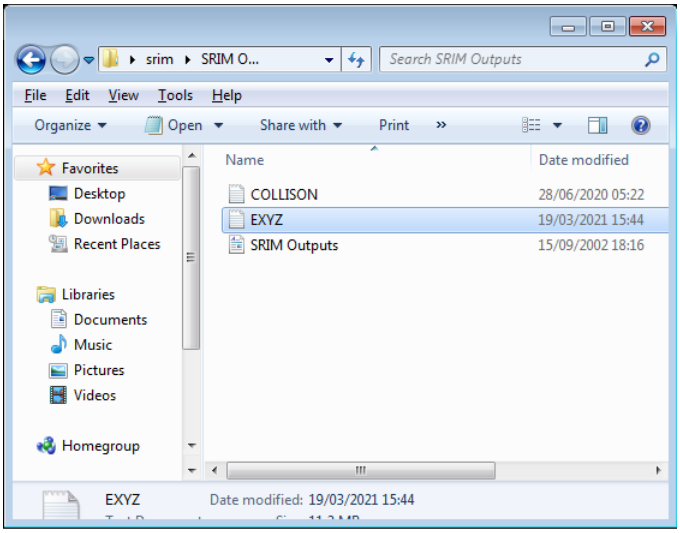
How To Use

3.1 SRIM

If you haven't done so already, run the required simulation in SRIM. When setting up the calculation, be sure to set an increment for the EXYZ file. Sane values that have been used in examples range from 10eV to 10,000KeV, but this will depend on the resolution of the data in the TENDL database, the thickness of your target and the energy of the projectiles.



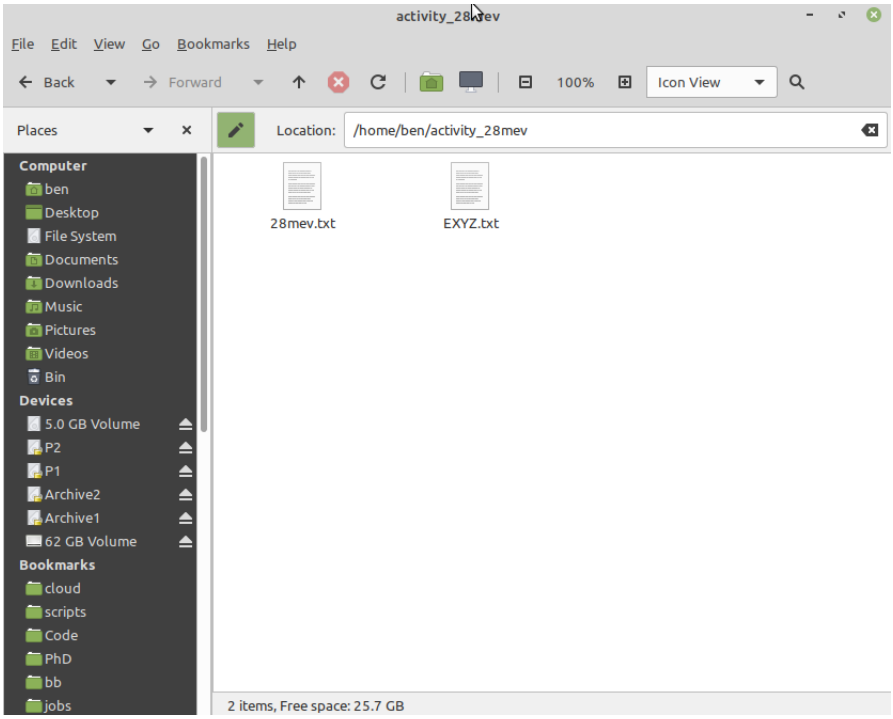




Take a copy of the EXYZ.txt file as it will be required by the activity program.

3.2 Activity

Create a directory to run the calculation. Copy in the EXYZ.txt file and create an input file.



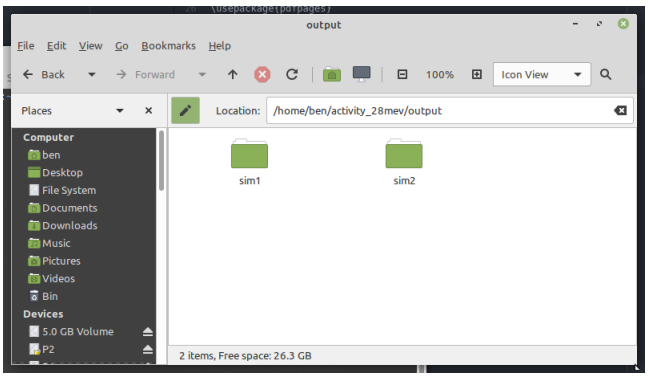
The input file is just a text file, and will specify the calculation details. The data file paths must be specified, and then the simulations can be detailed, each with it's own unique name. The example below is for two simulations - the first with a 300s beam, and the second with a 3000s beam.

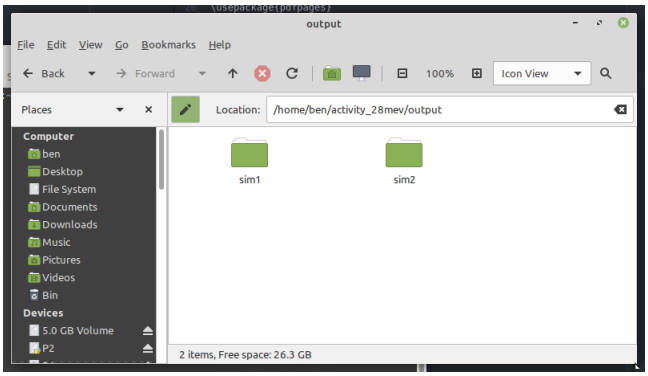
```
# Data files
data isotopes="/home/ben/activity/data/isotopes" xs="/home/ben/activity/data/xs"

# Sim 1
sim1 exyz="EXYZ.txt" target_composition=Fe,100.0 target_depth=0.5,mm target_density=7808,kgm3
    beam_projectile='proton' beam_energy=28,MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,
    uA end_time=260000,s

# Sim 2
sim2 exyz="EXYZ.txt" target_composition=Fe,100.0 target_depth=0.5,mm target_density=7808,kgm3
    beam_projectile='proton' beam_energy=28,MeV beam_area=64,mm2 beam_duration=3000,s beam_current
    =5.0,uA end_time=260000,s
```

The program is run from the terminal/command line. Depending on your computer system, python3 might run through the command python3 or python, and on my computer the command is python3.

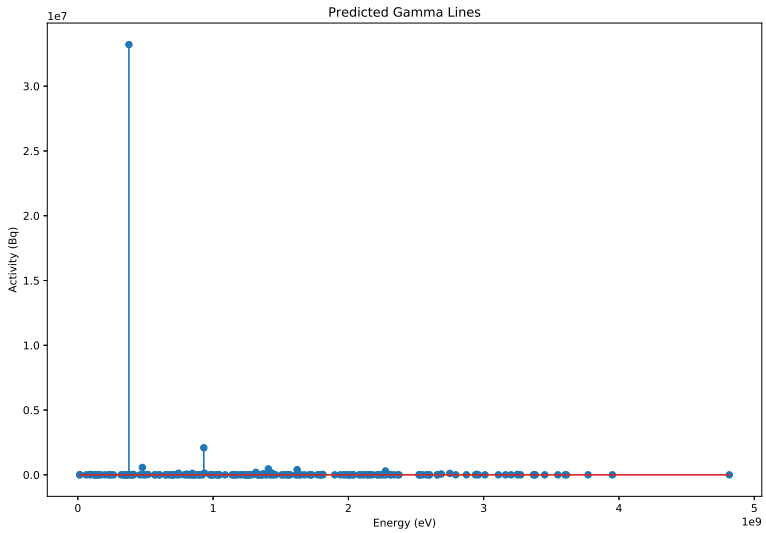
```
python3 /home/ben/activity/activity.py /home/ben/activity_28mev/28mev.txt
```

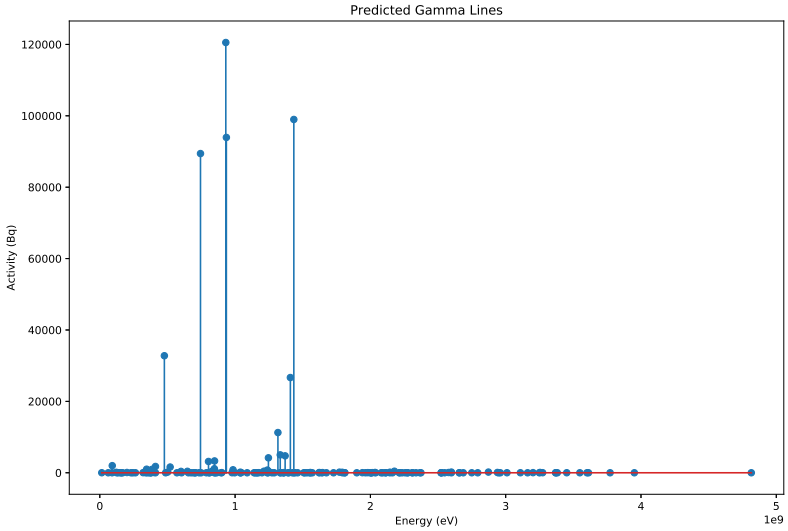
The program outputs into a directory for each simulation.

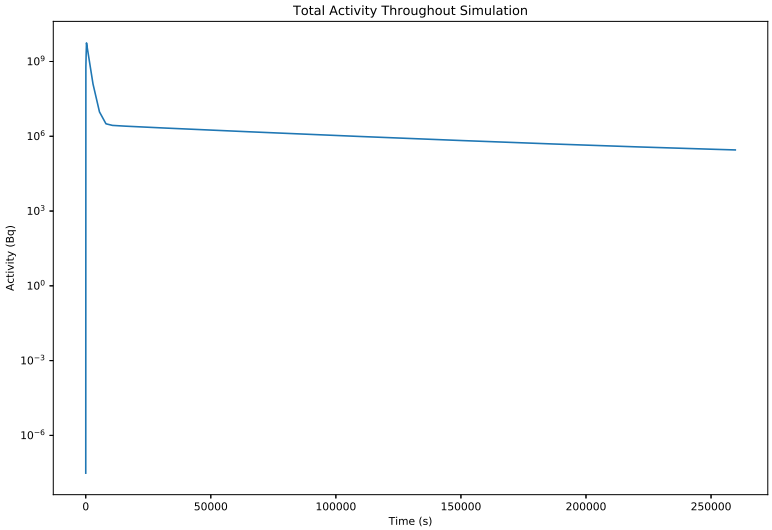




Within each directory are various plots and data files, including the activity over time and predicted gamma lines.



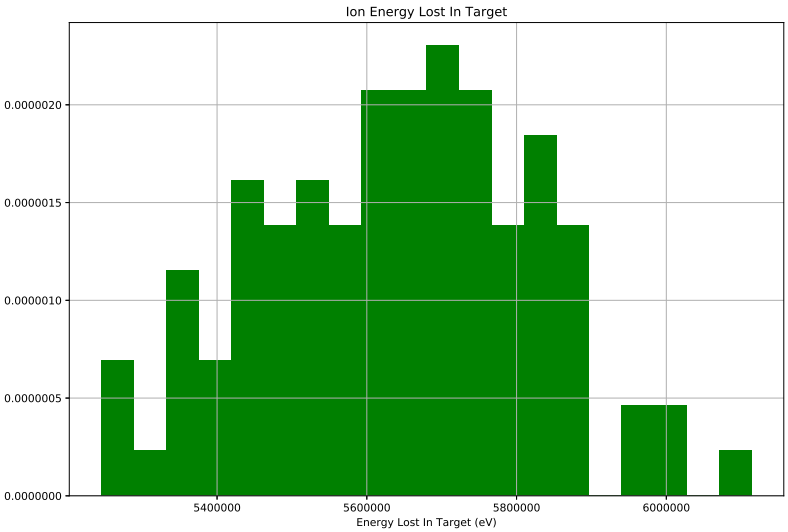




The program analyses the exyz file and output a chart showing the distribution of energy lost by projectiles as they pass through the target.

As a target is irradiated, the residual radioactive isotopes begin to decay. To begin with, the production outweighs the decay, but at a certain point there's enough of the radioactive isotope within the target that the decay balances with the production. At this point the radioactive isotope in the target is saturated, and will not increase above this amount unless the beam parameters are adjusted.

The saturation times depend on the isotope and the various reaction cross sections with the projectile. The saturation plots for each radioactive isotope are created and saved by the program.



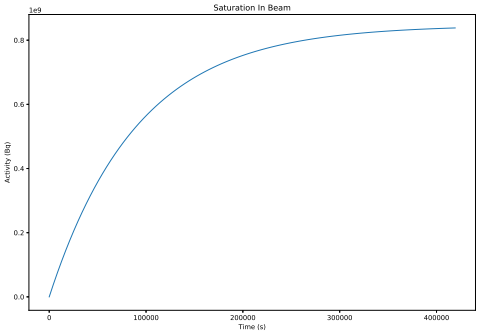


Figure 3.1: Saturation of Cobalt-55

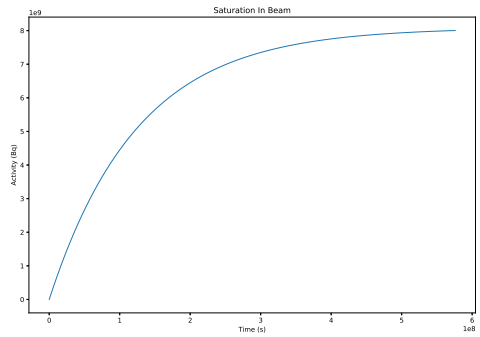


Figure 3.2: Saturation of Iron-55

The program calculates the dose a human may receive from the target at the time it is removed from the beam, and at the time the simulation stops.

Listing 3.1: Predicted Dose

```
Gamma Dose - Beam End
==================================================

Activity/Bq                 54.4306601689
Power eV/s                  28342871818.2
Power J/s                   4.54103823697e-09
Dose Gy/s                   4.51575003676e-12
Dose Gy/hr                  1.62567001323e-08
Percentage of annual dose/hr 0.0142506237521

Gamma Dose - Sim End
==================================================

Activity/Bq                 5.80221584491
Energy eV/s                 2227911995.02
Energy J/s                  3.56951604018e-10
Dose Gy                     8.52319971389e-13
Dose Gy/hr                  1.27786970412e-09
Percentage of annual dose/hr 0.00112018061534


Absorbed Dose Calculations
==================================================
Absorbed dose assumptions:
1. radiation from point, emitted isotropically
2. 80Kg human
3. 1m from point source
4. 1m squared surface area
5. all energy absorbed


Dose Limits
==================================================
employees 18+           20 millisieverts/year
trainees 18+            6 millisieverts/year
public and under 18s    1 millisievert/year
public and under 18s    1.140771128E-04 millisieverts/hour

Dose averaged over area of skin not exceeding 1cm2
Source: http://www.hse.gov.uk/radiation/ionising/doses/
```

3.3 Input Keywords

Command	***Sub Command***	Description
data		data fields are input here
data	isotopes	path to isotopes directory
data	xs	path to cross section directory
sim1		each simulation is numbered sequentially from 1 i.e. sim1 sim2 sim3...
sim1	exyz	path to exyz SRIM file
sim1	target_composition	isotopes that make up the beam target
sim1	target_depth	the depth of material through which the beam passes
sim1	target_density	how dense (specify kgm3 or GCM3)
sim1	beam_projectile	currently, only "proton", but this could be extended in the future
sim1	beam_energy	energy of the protons/projectiles
sim1	beam_area	how much area the beam covers
sim1	beam_duration	how long the beam will be on for (specify s,minute,hour) t=0 to beam_duration is the first part of the calculation
sim1	end_time	the time the calculation and final activity is measured after t=0 (specify s,minute,hour) end_time to beam_duration is the second part of the calculation

Chapter 4

Examples

4.1 Iron - 28MeV Protons

In this example, natural iron makes up the target .

```
# Data files
data isotopes="/home/ben/activity/data/isotopes" xs="/home/ben/activity/data/xs"

# Sim
sim1 exyz="EXYZ.txt" target_composition=Fe,100.0 target_depth=0.5,mm target_density=7808,kgm3
    beam_projectile='proton' beam_energy=28,MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,
    uA end_time=260000,s
```

The simulation section defines what the target material is made of as well as the beam parameters.

4.2 Iron Fe56 (only) - 28MeV Protons

In this example, the target is made of Fe56 only, and contains none of the other naturally occurring stable isotopes.

```
# Data files
data isotopes="/home/ben/activity/data/isotopes" xs="/home/ben/activity/data/xs"

# Sim
sim1 exyz="EXYZ.txt" target_composition=Fe56,100.0 target_depth=0.5,mm target_density=7808,kgm3
    beam_projectile='proton' beam_energy=28,MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,
    uA end_time=260000,s
```

4.3 Steel - 5MeV Protons

This example was provided by Alex Dickinson-Lomas, with a steel containing a wider range of elements.

```
# Data files
data isotopes="/home/ben/activity/data/isotopes" xs="/home/ben/activity/data/xs"

```

```
# Sim
sim1 exyz="EXYZ.txt" target_composition=Fe,96.375,C,0.772,Cu,0.024,Mn,1.36,Ni,0.698,Si,0.381,Cr,0.092,
    V,0.008,P,0.009,Si,0.003,Mo,0.278 target_depth=0.1,mm target_density=7808,kgm3 beam_projectile='
    proton' beam_energy=5,MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,uA end_time
    =260000,s
```

Chapter 5

Decay Equation

5.1 Bateman Equation

The Bateman equation was derived using Laplace transforms, and this same method has been used to develop a modified equation that incorporates branching factors and production rates for each isotope in the decay chain, as illustrated by Figure 5.1.

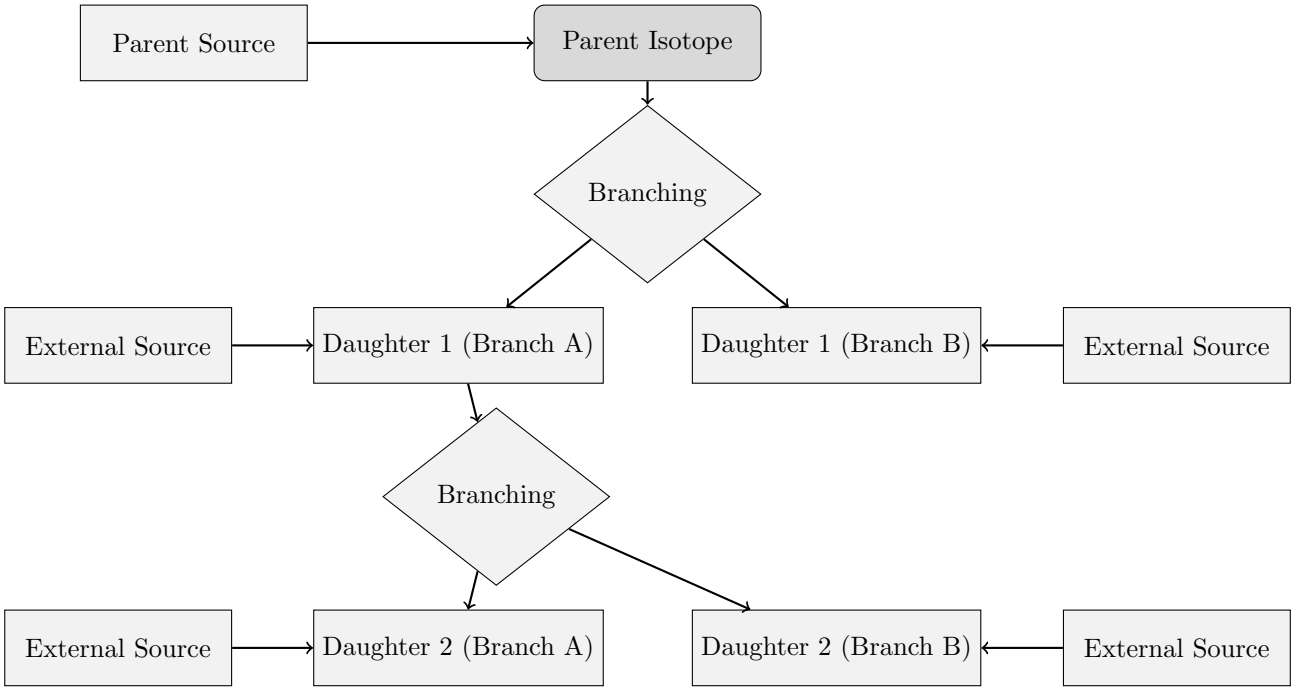


Figure 5.1: An example of several decay chains including branching factors and possible external source terms for each isotope on each chain.

5.1.1 Laplace Transform

Laplace Transforms (5.1) are a useful mathematical tool, and allow ordinary differential equations to be solved by simple algebraic manipulation in the s domain. Bateman took advantage of Laplace Transforms in deriving his equation, and this is the method that has been taken here as well.

$$F(s) = \int_0^\infty f(t) \exp(-st) dt \tag{5.1}$$

5.1.2 Constructing the Differential Equations

The first step is to set up differential equations for the parent isotope, unstable daughter isotopes and stable daughter isotope. The parent isotope has a source term, due to production, and a loss term, due to decay. The unstable daughter isotopes have two source terms, from the production by irradiation induced transmutation and the decay of preceding isotopes in the decay chain, and a loss term, due to decay. Finally, the stable daughter that finalizes the decay chain has two source terms (the same as the unstable daughters) but no loss term.

The variables (and vectors) used in these equations are defined as follows:

- $\vec{\lambda}$ vector containing isotope decay constants λ_i
- $\vec{b}$ vector containing isotope to isotope branching factors b_i
- $\vec{w}$ vector containing isotope production rates w_i
- t time at which activity/amount of isotope is measured
- $N_i(0)$ starting amount of the i^{th} isotope
- $N_i(t)$ amount of the i^{th} isotope at time t
- $N_i'(t)$ change in amount of the i^{th} isotope, with respect to time, at time t

The differential equations for the parent isotope (first isotope), unstable daughter isotopes (i^{th} isotopes) and stable, final, daughter isotope (zth isotope) in the time domain are as follows:

$$N_1'(t) = \omega_1 - \lambda_1 N_1(t) \tag{5.2}$$

$$N_i'(t) = \omega_i + b_{i-1}\lambda_{i-1}N_{i-1}(t) - \lambda_i N_i(t) \tag{5.3}$$

$$N_z'(t) = \omega_z + b_{z-1}\lambda_{z-1}N_{z-1}(t) \tag{5.4}$$

Applying the Laplace Transform to these three differential equations allows them to be manipulated and solved algebraically in the s-domain.

$$N_1(s) = \frac{1}{s+\lambda_1}N_1(0) + \frac{1}{s(s+\lambda_1)}\omega_1 \tag{5.5}$$

$$N_i(s) = \frac{1}{s(s+\lambda_i)}\left(\omega_i\right) + \frac{1}{s+\lambda_i}\left(b_{i-1}\lambda_{i-1}N_{i-1}(s)\right) + \frac{1}{s+\lambda_i}N_i(0) \tag{5.6}$$

$$N_z(s) = \frac{1}{s^2}\omega_z + \frac{1}{s}b_{z-1}\lambda_{z-1}N_{z-1}(s) + \frac{1}{s}N_z(0) \tag{5.7}$$

5.1.3 Numerical Inversion of the Laplace Transform

The Gaver-Stehfest[4] algorithm was developed in the 1960s and 1970s and is a method of calculating the inverse of a Laplace Transform in the real number domain. It is an easy to implement and reasonably accurate method,

although it is an approximation to the real value. A comparison between an analytic and numeric inversion for the unstable isotope Po-218 is discussed at the end of this section (figure 5.2).

$$f(t) \approx f_n(t) = \frac{\ln(2)}{t} \sum_{k=1}^{2n} a_k(n) F(s) \text{ where } n \geq 1, t > 0 \tag{5.8}$$

$$s = \frac{k \ln(2)}{t} \tag{5.9}$$

$$a_k(n) = \frac{(-1)^{(n+k)}}{n!} \sum_{j=Floor(\frac{k+1}{2})} j^{n+1} \binom{n}{j} \binom{2j}{j} \binom{j}{k-j} \tag{5.10}$$

The equation for the i$^{\text{th}}$ isotope may be calculated by recursively calculating the equations by numeric inversion, starting from the first (parent isotope) and inserting the result into each subsequent recursion until the i$^{\text{th}}$ isotope is reached (changing the equations appropriately for the parent, unstable daugher and stable daughter isotopes).

5.1.4 Analytic Solution by Partial Fraction Expansion

The equation for the i$^{\text{th}}$ isotope in the s domain can be written in full by substituting the preceding equation until the parent isotope is reached, and this full equation may be rearranged with the production amount of each isotope and starting amount of each isotope in individual terms. Each of these terms is multiplied by a fraction that can be expanded, using partial fractions, and inverted analytically.

This is illustrated with an example unstable isotope, fourth in the decay chain (including the parent isotope):

$$\begin{aligned} N_4(s) = {} & \frac{1}{(s+\lambda_1)(s+\lambda_2)(s+\lambda_3)(s+\lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\ & + \frac{1}{(s+\lambda_2)(s+\lambda_3)(s+\lambda_4)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\ & + \frac{1}{(s+\lambda_3)(s+\lambda_4)} b_4 \lambda_3 N_3(0) \\ & + \frac{1}{(s+\lambda_4)} N_4(0) \\ & + \frac{1}{s(s+\lambda_1)(s+\lambda_2)(s+\lambda_3)(s+\lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\ & + \frac{1}{s(s+\lambda_2)(s+\lambda_3)(s+\lambda_4)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\ & + \frac{1}{s(s+\lambda_3)(s+\lambda_4)} b_4 \lambda_3 \omega_3 \\ & + \frac{1}{s(s+\lambda_4)} \omega_4 \end{aligned} \tag{5.11}$$

An example stable isotope, fourth (last) in the decay chain (including the parent isotope):

$$
\begin{aligned}
N_4(s) = &\frac{1}{s(s+\lambda_1)(s+\lambda_2)(s+\lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\
&+ \frac{1}{s(s+\lambda_2)(s+\lambda_3)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\
&+ \frac{1}{s(s+\lambda_3)} b_4 \lambda_3 N_3(0) \\
&+ N_4(0) \\
&+ \frac{1}{s^2(s+\lambda_1)(s+\lambda_2)(s+\lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\
&+ \frac{1}{s^2(s+\lambda_2)(s+\lambda_3)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\
&+ \frac{1}{s^2(s+\lambda_3)} b_4 \lambda_3 \omega_3 \\
&+ \frac{1}{s^2} \omega_4
\end{aligned}
\tag{5.12}
$$

By using partial fraction expansion and standard Laplace Transforms, the set of equations below is used to calculate the amount of the m^{th} isotope in the decay chain, providing the m^{th} isotope is unstable.

$$
N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = \sum_{k=1,m} r(k; \vec{\lambda}, \vec{b}) \left[f(t; k, m, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k \right] \tag{5.13}
$$

$$
r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k,m-1} \left(b_{i+1} \lambda_i \right), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \tag{5.14}
$$

$$
f(t; k, m, \vec{\lambda}) = (-1)^{m-k} \sum_{i=k,m} \left[\exp(-\lambda_i t) \prod_{j=k,m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \tag{5.15}
$$

$$
g(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k,m} \lambda_i} + (-1)^{m-k+1} \sum_{i=k,m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k,m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \tag{5.16}
$$

The set of equations below is used to calculate the amount of the m^{th} isotope in the decay chain, where the m^{th} isotope is stable.

$$
N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = N_m + w_m t + \sum_{k=1,m-1} r(k; \vec{\lambda}, \vec{b}) \left[f(t; k, m-1, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k \right] \tag{5.17}
$$

$$
r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k,m-1} \left(b_{i+1} \lambda_i \right), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \tag{5.18}
$$

$$f(t;k,m,\vec{\lambda}) = \frac{1}{\prod_{i=k,m}\lambda_i} + (-1)^{m-k+1}\sum_{i=k,m}\left[\frac{1}{\lambda_i}\exp(-\lambda_i t)\prod_{j=k,m;j\neq i}\left(\frac{1}{\lambda_i-\lambda_j}\right)\right] \tag{5.19}$$

$$g(t;k,m,\vec{\lambda}) = \frac{1}{\prod_{i=k,m}\lambda_i}t + \frac{\sum_{i=k,m}\left[\prod_{j=k,m;j\neq i}\lambda_j\right]}{\prod_{i=k,m}\lambda_i^2} + (-1)^{m-k+1}\sum_{i=k,m}\left[\frac{1}{\lambda_i^2}\exp(-\lambda_i t)\prod_{j=k,m;j\neq i}\left(\frac{1}{\lambda_i-\lambda_j}\right)\right] \tag{5.20}$$

5.1.5 Preference: Analytic over Numeric

The numeric solution only requires the equation to be solved in the s-domain; the Gaver-Stehfest algorithm performs the inversion. It is worth the extra effort to derive and implement an analytic solution, as the numeric is only an approximation. Examples of the pitfalls of the numeric solution are that it can give negative amounts of an isotope and the difference between the numeric and analytic calculated amounts can become quite large when the isotope decays away to a very small value. Figure 5.2 shows the predicted decay of a sample of Po-218 irradiated for 1,000s, and sampled until 10,000s. In the region between 4,000s and 9,000s the amount from the numeric calculation drops below zero, whereas the analytic calculation remains above zero, as would be expected.

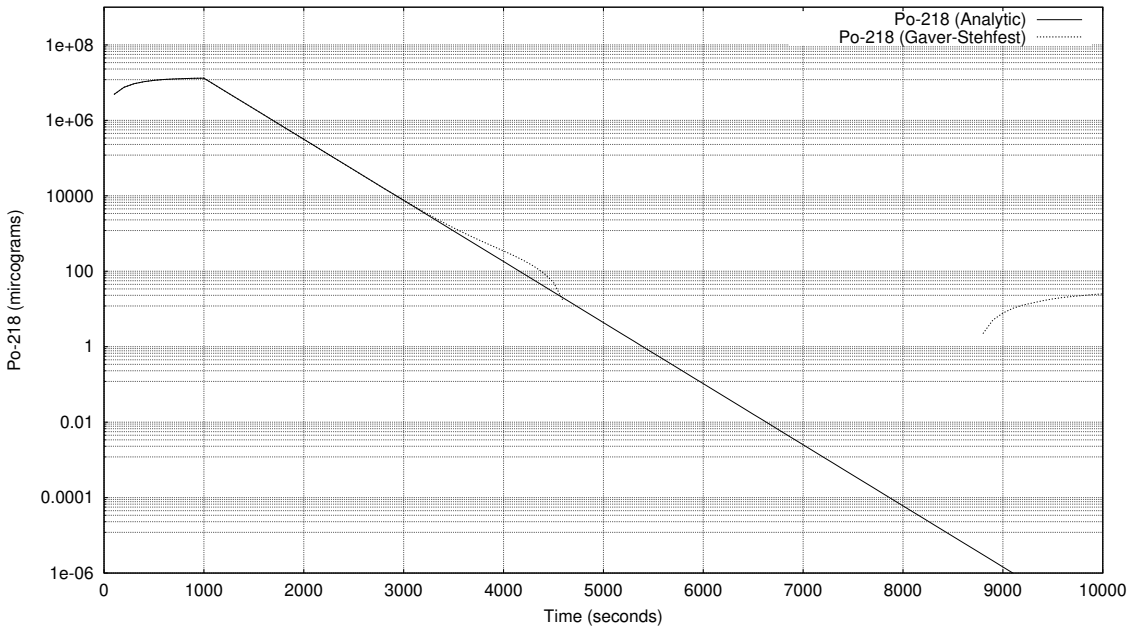


Figure 5.2: Decay of Po-218: Analytice and Gaver-Stehfest Calculations **[jeff311]**

5.2 Python Isotopes Class

The decay equations are computed within the isotopes class in the isotopes.py file.

```
1
2  class isotopes:
3
4    ##########################################
5    # DECAY EQUATIONS
6    ##########################################
7
8    @staticmethod
9    def calculate_activity(t, l, b, w, n0):
10     nt = numpy.zeros((len(n0),),)
11     for m in range(0,len(n0)):
12       if(l[m] > 0.0):
13         nt[m] = isotopes.activity_unstable(t, l, b, w, n0, m)
14       elif(l[m] == 0.0):
15         nt[m] = isotopes.activity_stable(t, l, b, w, n0, m)
16     return nt
17
18   @staticmethod
19   def activity_unstable(t, l, b, w, n0, m):
20     s = 0.0
21     for k in range(0, m+1):
22       s = s + isotopes.r(k, m, b, l) * ( isotopes.f_unstable(t,k,m,l) * n0[k] + isotopes.g_unstable(t,
           k,m,l) * w[k])
23     return s
24
25   @staticmethod
26   def activity_stable(t, l, b, w, n0, m):
27     s = n0[m] + w[m] * t
```

```
    for k in range(0, m):
      s = s + isotopes.r(k, m, b, l) * (isotopes.f_stable(t,k,m,l) * n0[k] + isotopes.g_stable(t,k,m,l
          ) * w[k])
    return s

  @staticmethod
  def r(k, m, b, l):
    if(k == m):
      return 1.0
    else:
      p = 1.0
      for i in range(k, m):
        p = p * (b[i] * l[i])
      return p

  @staticmethod
  def f_unstable(t,k,m,l):
    s = 0.0
    for i in range(k, m+1):
      p = 1.0
      for j in range(k, m+1):
        if(i != j):
          p = p * (1 / (l[i] - l[j]))
      s = s + numpy.exp(-1 * l[i] * t) * p
    s = (-1)**(m-k) * s
    return s

  @staticmethod
  def g_unstable(t,k,m,l):
    pa = 1.0
    for i in range(k,m+1):
      pa = pa * l[i]
    pa = 1.0 / pa
    s = 0.0
    for i in range(k, m+1):
      pb = 1.0
      for j in range(k, m+1):
        if(i != j):
          pb = pb * (1 / (l[i]-l[j]))
      s = s + (1/l[i]) * numpy.exp(-l[i]*t) * pb
    return pa + s * (-1)**(m-k+1)


  @staticmethod
  def f_stable(t,k,m_in,l):
    m = m_in - 1

    p = 1.0
    for i in range(k, m+1):
      p = p * l[i]

    s = 0.0
    for i in range(k, m+1):
```

```
      r = l[i]
      for j in range(k, m+1):
        if(i != j):
          r = r * (l[i] - l[j])
      s = s + (1/r)*numpy.exp(-1*l[i]*t)

    return (1.0/p) + s * (-1.0)**(m-k+1)


  @staticmethod
  def g_stable(t,k,m_in,l):
    m = m_in - 1

    pa = 1.0
    for i in range(k,m+1):
      pa = pa * l[i]
    pa = 1.0 / pa

    sa = 0.0
    for i in range(k, m+1):
      pb = 1.0
      for j in range(k,m+1):
        if(j != i):
          pb = pb * l[j]
      sa = sa + pb
    pc = 1.0
    for i in range(k, m+1):
      pc = pc * l[i]**2

    sb = 0.0
    for i in range(k, m+1):
      pd = 1.0
      for j in range(k, m+1):
        if(i != j):
          pd = pd * (1 / (l[i]-l[j]))
      sb = sb + (1/(l[i]**2)) * numpy.exp(-l[i]*t) * pd

    return pa * t + sa / pc + sb * (-1)**(m-k+1)
```

Chapter 6

Future Plans

6.1 Selection of Cross Section Databases

The Scanditronix MC40 Cyclotron at the University of Birmingham has several beamlines and is capable of accelerating protons, deuterons, Helium 3 and Helium 4 and fluxes and energy ranges detailed below.

Particle	Energy (MeV)	Max Current (micro A)	Flux (ions per second)
p	8-40	60	$3.75 \times 10^1 4$
d	8-40	30	$1.87 \times 10^1 4$
$^4He^{2+}$	8-53	30	$9.36 \times 10^1 3$
$^3He^{2+}$	4-20	60	$1.87 \times 10^1 4$

Table 6.1: Beam Characteristics of the Scanditronix MC-40

The current data file is for Protons, and the range of energies (for Fe-54 and Fe-56 at the very least) only range up to 30MeV. Previous versions of the TENDL data files have covered larger ranges (TENDL-2009 extended up to 200MeV).

As the energies do not cover the full range of possible energies for our own cyclotron, it would be desirable to use the TALYS program to calculate the reaction cross sections for a range up to at least 100MeV (for Protons, Deuterons, Helium 3 and Helium 4 ions) and save this cross section data into an alternate database.

Bibliography

[1] A.J. Koning et al. “TENDL: Complete Nuclear Data Library for Innovative Nuclear Science and Technology”. In: *Nuclear Data Sheets* 155 (2019).

[2] A.J.M. Plompen et al. “The JEFF-3.3 Nuclear Data Library”. In: *Eur. Phys. J. A* 56 (181 2020).

[3] J. P. Biersack James F. Ziegler M. D. Ziegler. “SRIM - The stopping and range of ions in matter”. In: *Nuclear Instruments and Methods in Physics Research Section B* 268 (2010), pp. 1818–1823.

[4] H. Stehfest. “Algorithm 368: Numerical Inversion of Laplace Transform”. In: *Communications of the ACM* 13 (1970), pp. 47–49.