NUEN 647 Final Project

Uncertainty quantification of depletion calculations for specific isotopes using ORIGEN2.

I Introduction

Determing composition of irradiated fuel is of importance for a myriad of reasons. Whether for flux calculations, reprocessing, or irradiation history verification, calculating fuel composition requires a Bateman solver, and a means for building a sparse matrix.

Applications using these compositions rarely report the uncertainty associated with results, even when inputs, such as flux shape, fission yield, cross sections, and half-lives have varying degrees of uncertainty. Further sources of error in this calculation are due to the multi-group approximation, and single point approximation, but will not be explored here.

Several isotope concentrations, shown in Table 1, were calculated as a function of burnup with the depletion code ORIGEN2 for a PWR system with 3 Wt% enriched uranium. ORIGEN2 solves the bateman equations with the matrix expoential method and requires a library with decay and cross section information. Cross sections and fission product yields are reduced to single group through flux averaging before execution of the code, with the assumption that the flux has the same shape as a typical PWR.

The uncertainty of concentrations were determined by varying the absorption and fission cross sections for ^{235}U, ^{238}U ^{239}Pu, ^{240}Pu, and ^{241}Pu. Originally, absorption cross sections and yields for the fission products were to be varied, but variance information for the lighter nuclides is either difficult to acquire or not available.

The uncertainties on cross sections were determined by calculating the range of the single group cross section, taking the mid point as a mean, the range as a standard deviation, and assuming a Gamma distribution.

Table 1: Isotope solve list.

^{133}Cs	^{136}Ba	^{153}Eu
^{134}Cs	^{138}Ba	^{154}Eu
^{135}Cs	^{149}Sm	^{239}Pu
^{137}Cs	^{150}Sm	^{242}Pu
^{148}Nd	^{106}Rh	^{125}Sb

II Objectives

- ☑ Build ORIGEN2 model for thermal system which calculates concentrations of isotopes shown in Table 1.

Listing 1: PWR Input Deck

```
-1
-1
-1
```

```
 RDA  Irradiation of 1 MT of PWR fuel
 RDA  Fuel enrichment is 3.0 w/o U-235
 RDA
 RDA  LPU 922350 922380 942390 942400 942410 -1
 LIB  1  1 2 3  601 602 603  9  8 0 1 38
 PHO     101 102 103  10
 INP  1   1  -1  -1   1   1
 BUP
 IRP 100.0 37.5 1 2 4 2 BURNUP=3,750 MWD/MT
 IRP 200.0 37.5 2 3 4 0 BURNUP=7,500 MWD/MT
 IRP 300.0 37.5 3 4 4 0 BURNUP=11,250 MWD/MT
 IRP 400.0 37.5 4 5 4 0 BURNUP=15,000 MWD/MT
 DEC 500.0      5 6 4 0 DECAY FOR 100.0 DAYS
 DEC 4150.0     6 7 4 0 DECAY FOR 10 YEARS
 DEC 73500.0    7 8 4 0 DECAY FOR 200.0 YEARS
 BUP
 OPTL 24*8
 OPTA 4*8 5 19*8
 OPTF 4*8 5 19*8
 OUT     8   1  -1   0
 END
2 922340 270. 922350 30000. 922380 969730. 0 0.0
0
```

The model irradiates 1 metric ton of US PWR fuel for a single cycle (15,000 MWd/Mt). The calculations use a constant power assumption of 37.5 W/g. The model does not include the oxygen because we are not interested in the activation of oxygen. Cross section modifcation throughout the calculation use the changing flux associated with a US PWR.

Initial verification of the model analyzed the end concentration of ^{137}Cs and calculated the burn-up from that value. This calculation does not have an exact value for the yield of ^{137}Cs and is used qualitatively as a sanity check.

$$\frac{552.8 \text{ g } ^{137}\text{Cs}}{Mt} \cdot \frac{6.022E23 \text{ atoms}}{137 \text{ g } ^{137}\text{Cs}} \cdot \frac{\text{Fission}}{0.06 \text{ atoms}} \cdot \frac{200 \text{ MeV}}{\text{Fission}} \cdot \frac{1.602E-19 \text{ MJ}}{1 \text{ MeV}} \cdot \frac{1 \text{ day}}{86400 \text{ s}} = 15,018 \frac{\text{MWd}}{Mt}$$

☑ Determine how to vary cross section and or flux spectrum inputs for calculation

ORIGEN2 reads in cross section information through a file named "TAPE9.inp", specified by the 8th input on the LIB card. "TAPE9.inp" needs at least 3 cross section libraries. These are specified by the 5th 6th and 7th inputs on the LIB card as 601, 602, and 603 for the activation products, actinides, and fission products, respectively. The input for ^{235}U from library 601 is shown in the listing below, with a corresponding key shown in Table 2 [1].

Listing 2: ^{235}U cross section library 602 input

```
602   922350  1.068E+01  2.338E-03  8.049E-07  4.752E+01  0.0          0.0          -1.0
```

Table 2: Key to parameters in cross section library

LIB	NUCLID	(n,γ)	(n,2n)	(n,3n)	(n,f)	(n,γ^*)	(n,2n*)	YYN

The cross sections, σ_γ and σ_f will be modified based on the locations in the cross section libraries shown above. Half-life information is contained in the decay libraries 1, 2, and 3 for activation products, actinides, and fission products, respectively. These values will not be modified to reduce the scope of the project.

Modifying the cross section libraries seems to have no bearing on the output of the code. This will be shown in the results section.

A program was writen to modify σ_γ, yield, or half-life based on lines from a text file. The project was simplified to only modify cross section input for the actnides, and therefore a modified version of is provided below. The code below is used to modify σ_γ and σ_f.

Listing 3: Script for modifying ORIGEN2 input.

```
#!/usr/bin/env python3

#Please note, might have to run command
# sed -i 's/E /E\+/g' TAPE9_BANK.inp
# on file to make sure there are no spaces after E's

########################################################################
######################### Import Packages ##############################
########################################################################

import time
start_time = time.time()
import numpy as np
import Functions as Fun
from subprocess import call



########################################################################
############################ Low Class #################################
########################################################################
#As opposed to high class
class ToPull:
    def __init__(self):
        self.Isotope  = ""
        self.Type     = ""

ListToPull=[]
with open('LISTTOPULL') as f:
    Lines=f.readlines()
for i in Lines:
    i=i.split()
    ListToPull.append(ToPull())
    ListToPull[-1].Isotope =i[0]
    ListToPull[-1].Type    =i[1]

########################################################################
######################### Load all samples #############################
########################################################################

with open('SAMPLES/94Pu239a') as f:
    Pu239a=f.readlines()
```

```
Pu239a=Fun.StripNL(Pu239a)
with open('SAMPLES/94Pu240a') as f:
    Pu240a=f.readlines()
Pu240a=Fun.StripNL(Pu240a)
with open('SAMPLES/94Pu241a') as f:
    Pu241a=f.readlines()
Pu241a=Fun.StripNL(Pu241a)

with open('SAMPLES/94Pu239f') as f:
    Pu239f=f.readlines()
Pu239f=Fun.StripNL(Pu239f)
with open('SAMPLES/94Pu240f') as f:
    Pu240f=f.readlines()
Pu240f=Fun.StripNL(Pu240f)
with open('SAMPLES/94Pu241f') as f:
    Pu241f=f.readlines()
Pu241f=Fun.StripNL(Pu241f)

with open('SAMPLES/92U235a') as f:
    U235a=f.readlines()
U235a=Fun.StripNL(U235a)
with open('SAMPLES/92U238a') as f:
    U238a=f.readlines()
U238a=Fun.StripNL(U238a)

with open('SAMPLES/92U235f') as f:
    U235f=f.readlines()
U235f=Fun.StripNL(U235f)
with open('SAMPLES/92U238f') as f:
    U238f=f.readlines()
U238f=Fun.StripNL(U238f)

######################################################################
#################### Load TAPE9 Template ############################
######################################################################

with open('ORIGENBACKUP/TAPE9_BANK.inp') as f:
    TAPE9content=f.readlines()
f.close()

######################################################################
#################### Loop through all samples ########################
############################ and ... #################################
######################################################################

for k in range(0,len(U238f)):
#for k in range(0,1):
    #k=2
    ##################################################################
    ################## Make new TAPE9 ################################
    ##################################################################

    #Open output file
    output=open("TAPE8.inp","w")

    #Loop through the TAPE9 template file, make changes,
```

```
    #and write to new TAPE9 (output)
    for i in TAPE9content:
        hold=i.split()
        toprint=False
        if '602' in hold[0] and "-" not in hold[0]:
            if "942390" in hold[1]:
                #Replace the gamma x-section
                i=i.replace(hold[2],Pu239a[k])
                #Replace the fission x-section
                i=i.replace(hold[5],Pu239f[k])
                toprint=True
            if "942400" in hold[1]:
                #Replace the gamma x-section
                i=i.replace(hold[2],Pu240a[k])
                #Replace the fission x-section
                i=i.replace(hold[5],Pu240f[k])
                toprint=True
            if "942410" in hold[1]:
                #Replace the gamma x-section
                i=i.replace(hold[2],Pu241a[k])
                #Replace the fission x-section
                i=i.replace(hold[5],Pu241f[k])
                toprint=True
            if "922350" in hold[1]:
                #Replace the gamma x-section
                i=i.replace(hold[2],U235a[k])
                #Replace the fission x-section
                i=i.replace(hold[5],U235f[k])
                toprint=True
            if "922380" in hold[1]:
                #Replace the gamma x-section
                i=i.replace(hold[2],U238a[k])
                #Replace the fission x-section
                i=i.replace(hold[5],U238f[k])
                toprint=True
            if "ACTINIDE+AND" in i:
                toprint=True
            if toprint:
                i=i.replace("\n","")
                print(i,file=output)

    print("  -1",file=output)
    output.close() #important or ORIGEN2 wont run
    #endfile record being detected otherwise

    ############################################################
    ##################### Run ORIGEN2 ##########################
    ############################################################

    #Run with new TAPE9, also deletes old files
    print("Run number :"+str(k+1))
    call(["./r"])

    ############################################################
    ################## COLLECT OUTPUT ##########################
    ############################################################
```

```
155
    #Open output file and save contents
    with open('TAPE6.OUT') as f:
        content=f.readlines()

160 #Work on Page by Page Basis for output
    #Also grab time steps
    Pages=[];Page=""
    TimeDays=[];AD="ACTINIDES+DAUGHTERS" #For space
    for i in content:
165     if "PAGE" in i:
            #Save the Page if it has useful information
            if AD in Page or "FISSION PRODUCTS" in Page:
                Pages.append(Page)
            Page=""
170     else:
            Page=Page+i
        #If we have located the time steps pull the info
        if "TIME, SEC" in i:
            i=i.replace("TIME, SEC","")
175         i=i.split()
            for j in i:
                TimeDays.append(str(round(float(j)/86400,2)))

    #Pages has a list of strings, each with a page on it
180 #from the output
    #The only pages saved in Pages are ones with text:
    #"ACTINIDES+DAUGHTERS" or "FISSION PRODUCTS" the rest
    #of the output is trashed

185 #Loop through all isotopes and grab info
    for item in ListToPull:
        Isotope=item.Isotope
        Type=item.Type
        Info=Fun.Graboutput(Isotope,Pages,Type)
190     if k==0:
            SaveFile=open("OUTPUTDATA/"+Isotope+'.out','w')
            print('D,'.join(TimeDays)+'D',file=SaveFile)
            print(','.join(Info),file=SaveFile)
            SaveFile.close()
195     else:
            SaveFile=open("OUTPUTDATA/"+Isotope+'.out','a')
            print(','.join(Info),file=SaveFile)
            SaveFile.close()


200
##########################################################################
######################### Time to Execute ################################
##########################################################################

205 print("--- %s seconds ---" % (time.time() - start_time))
```

☑ Determine variance of Cross-sections

Both σ_γ and σ_f were determined by flux averaging via:

$$\sigma = \frac{\int \sigma(E)\phi(E)dE}{\int \phi(E)dE}$$

where:

$$\begin{aligned}
\phi(E) =& C_1 \cdot \frac{E}{E_0^2} \cdot exp\left(-\frac{E}{E_0}\right) && E < E_{max,th} \\
=& \frac{C_2}{E} && E_{max,th} < E < E_{max,epi} \\
=& C_3 \cdot \frac{\sqrt{\frac{E}{E_f}}}{E_f} \cdot exp\left(-\frac{E}{E_f}\right) && E > E_{max,epi}
\end{aligned}$$

and:

$$\begin{aligned}
C_1 =& \frac{E_0^2}{E_{max,th}^2} e^{E_{max,th}/E_0} \\
C_2 =& 1 \\
C_3 =& \frac{E_f}{E_{max,epi}} \cdot e^{\frac{E_{max,epi}}{E_f}} \frac{1}{\sqrt{\frac{E_{max,epi}}{E_f}}}
\end{aligned}$$

Where: $E_{max,th} = 0.50$ eV, $E_{max,epi} = 1E5$ eV, $\theta_{th} = 0.09$ eV (764 K), and $\theta_{fis} = 1.35E6$ eV. These values were picked because they minimized the difference between the cross-sections in the TAPE9 file, and those calculated with ENDF-VII and the above method. This is highlighted in figure 1. Most of this error is from ^{238}U and ^{240}Pu.

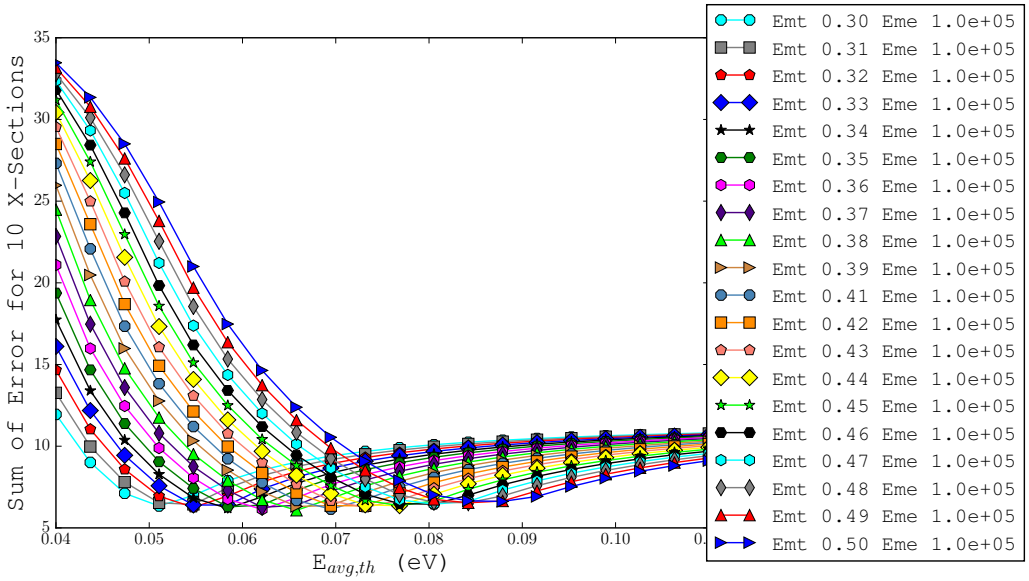

Figure 1: Minimized error for 10 cross section calculations

The flux spectrum is shown as below:

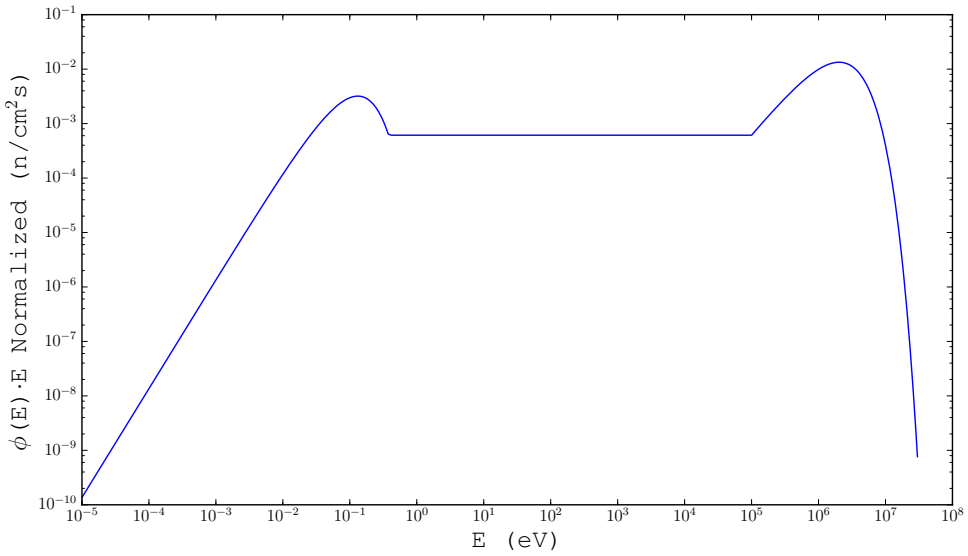

Figure 2: Flux Spectra used for weighting x-sections and yields

Table 3 shows the difference between my calculated cross section and the cross sections provided in ORIGEN2. The ratios will be used as a correction factor for the cross sections so that the input to ORIGEN2 will be constant. Differences have been attributed to the fact that the flux spectrum used does not capture reasonances. The cross sections were calculated with ENDF VII, and ORIGEN2 could have been processed with ENDF V, which would further account for differences, but has not been verified. The reason ENDF VII is being used for the current analysis is because the variance information is included in ENDF VII and not ENDF V.

Table 3: Comparison of one-group cross sections

Isotope^{Rxn}	ENDF VII	ORIGEN2	Ratio
$^{239}Pu^{\gamma}$	6.544e+01	6.909E+01	1.06
$^{240}Pu^{\gamma}$	1.521e+02	2.228E+02	1.46
$^{241}Pu^{\gamma}$	4.518e+01	4.202E+01	0.93
$^{235}U^{\gamma}$	9.387e+00	1.068E+01	1.14
$^{238}U^{\gamma}$	4.098e+00	8.872E-01	0.22
$^{239}Pu^{f}$	1.179e+02	1.211E+02	1.03
$^{240}Pu^{f}$	9.609e-01	5.787E-01	0.60
$^{241}Pu^{f}$	1.253e+02	1.259E+02	1.01
$^{235}U^{f}$	4.621e+01	4.752E+01	1.03
$^{238}U^{f}$	2.091e-01	9.281E-02	0.44

Both σ_{γ}^{error} and σ_{f}^{error} were determined by calculating the single group cross section with error subtracted, and then added. These values will constitue a mean with error. This was done with the following code:

Listing 4: Script calculating average x-sections

```
#!/usr/bin/env python3
"""
This program will compute 1-group cross sections with a weighted
flux. Parameters for the flux were
determined in a subdirectory called Reduce_Err.
"""

#########################################################################
######################### Import Packages ###############################
#########################################################################

import time
start_time = time.time()
import Functions as f
from scipy import interpolate
from scipy import integrate
from scipy.integrate import trapz

#########################################################################
########################### Calculations ################################
#########################################################################

#To fix X-section data to ORIGEN values
Ratios=[1.05578868993,1.46437937788,0.929974069639,1.13769098926,
        0.216470218472,1.02697467277,0.602248684356,1.0049132997,
        1.02836592353,0.443767334257]

#########################################################
############### Import X-Section Data ###################
#########################################################

#Get list of csv files with X-section information
Names=f.GETcsvFiles("X_Sections")
Names=["Pu_239_94_a.csv",
       "Pu_240_94_a.csv",
       "Pu_241_94_a.csv",
       "U_235_92_a.csv",
       "U_238_92_a.csv",
       "Pu_239_94_f.csv",
       "Pu_240_94_f.csv",
       "Pu_241_94_f.csv",
       "U_235_92_f.csv",
       "U_238_92_f.csv"]

#Flux Parameters
Emt=0.38        #Max thermal energy in ev
Eme=1e5         #Max epithermal energy in ev
E0=0.0658        #Thermal average in ev (1045 K)
Ef=1.35e6       #Fission average in ev

#Loop through all the X-sections I got
index=0
for Name in Names:

    Element=Name.split('_')[0]
```

```
    Isotope=Name.split('_')[1]
    Protons=Name.split('_')[2]
    Reaction=Name.split('_')[3].split('.')[0]

    #Do not do Averaging of variances
    if 'V' in Reaction:
        continue

    Xsec = f.np.genfromtxt('X_Sections/'+Name,delimiter=',')
    #Modify Xsections to match with ORIGEN2
    Xsec[:,1]=Xsec[:,1]*Ratios[index]
    index=index+1

    #Set energy, and convert from MeV to ev
    E=f.copy.copy(Xsec[:,0])*10**6

    #Gather Variance and make function for it
    VarName=Name.split(".")[0]+"V.csv"
    Var=f.np.genfromtxt('X_Sections/'+VarName,delimiter=',')
    Var_int=interpolate.interp1d(Var[:,0],Var[:,1],
                                 fill_value=0,bounds_error=False)

    #Determine the absolute err from the variance
    ErrAb=(Var_int(E)/100)*Xsec[:,1]
    #Find minimum X-section
    Xmin=Xsec[:,1]-ErrAb
    #Find Max X-section
    Xmax=Xsec[:,1]+ErrAb

    #Calculate flux (yes we need E)
    F=f.flux(E,Emt,Eme,E0,Ef)

    #Make function for Min-X-Section(E) * Flux(E)
    X_phimin=interpolate.interp1d(E,F*Xmin,
                                  fill_value=0,bounds_error=False)
    #Make function for Max-X-Section(E) * Flux(E)
    X_phimax=interpolate.interp1d(E,F*Xmax,
                                  fill_value=0,bounds_error=False)
    #Perform the integral for Max-X-Section(E) * Flux(E)
    X_int_max=integrate.trapz(X_phimax(E),E)
    #Perform the integral for Min-X-Section(E) * Flux(E)
    X_int_min=integrate.trapz(X_phimin(E),E)
    #Perform the integral for Flux(E)
    Phi_int=integrate.trapz(F,E)
    #Average X-section value
    Avgmin=X_int_min/Phi_int
    Avgmax=X_int_max/Phi_int

    Avg=(Avgmin+Avgmax)/2

    print(Protons+Element+Isotope+Reaction+' '+str(Avg)+
          '+/-'+str(Avg-Avgmin))

    #With the ratio fixes, Ratio should be one
    #Find TAPE9's X-section value for comparison
    #TAPE9_X=f.LoopTAPE(Protons,Isotope,Reaction)
```

```
    #Ratio=str(float(TAPE9_X)/Avg)
    #print(Protons+Element+Isotope+Reaction+' Average: %.3e' % Avg
    #       +', TAPE Value: '+TAPE9_X+
    #       ", Their Ratio: "+Ratio)


##########################################################################
######################### Time to Execute ################################
##########################################################################

print("--- %s seconds ---" % (time.time() - start_time))
```

With results in Table 4

Table 4: Errors in single group cross sections

Isotope^{Rxn}	σ with 1STD Error
$^{239}Pu^{\gamma}$	69.09±8.15
$^{240}Pu^{\gamma}$	222.8±50.9
$^{241}Pu^{\gamma}$	42.02±10.92
$^{235}U^{\gamma}$	10.68±3.23
$^{238}U^{\gamma}$	0.887±0.175
$^{239}Pu^{f}$	121.1±1.2
$^{240}Pu^{f}$	0.579±0.003
$^{241}Pu^{f}$	125.9±2.3
$^{235}U^{f}$	47.52±0.71
$^{238}U^{f}$	0.093±8.2e-7

Cross section versus flux for several isotopes are shown in the following figures.

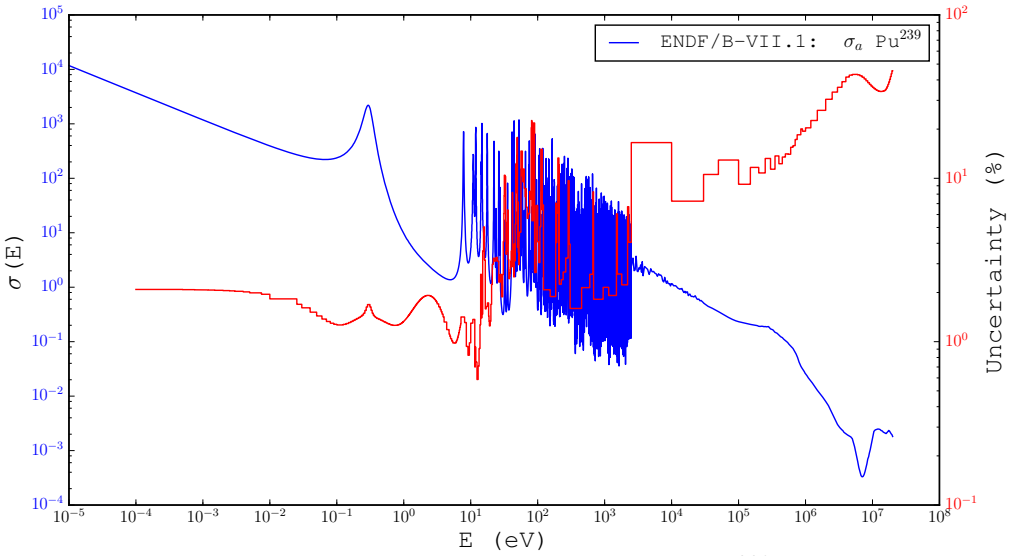

Figure 3: σ_a with corresponding error for ^{239}Pu

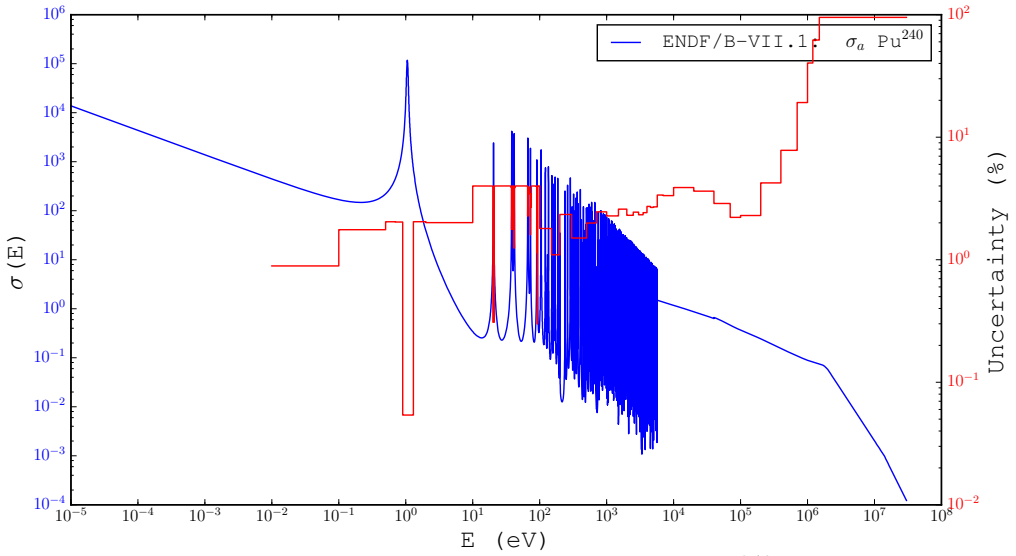

Figure 4: σ_a with corresponding error for ^{240}Pu

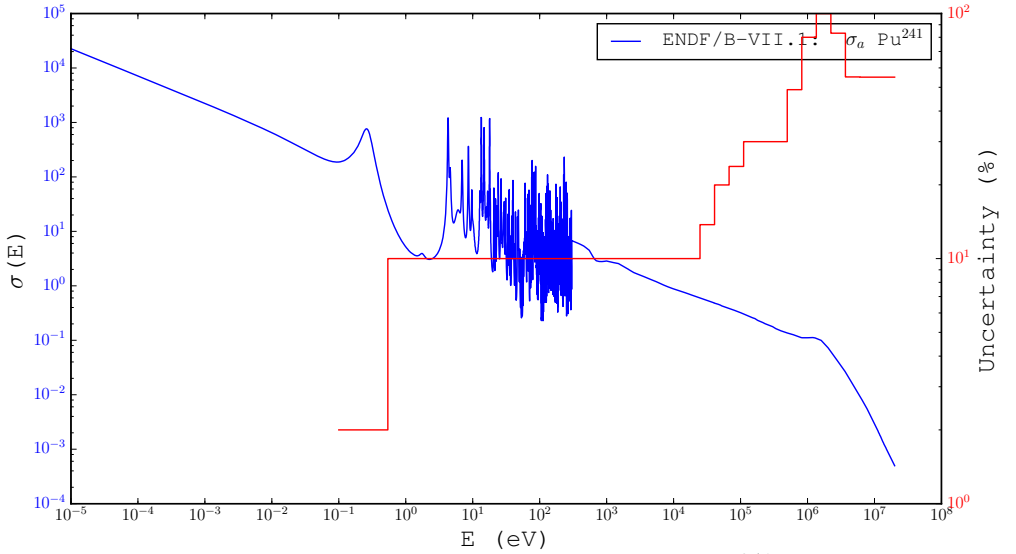

Figure 5: σ_a with corresponding error for ^{241}Pu

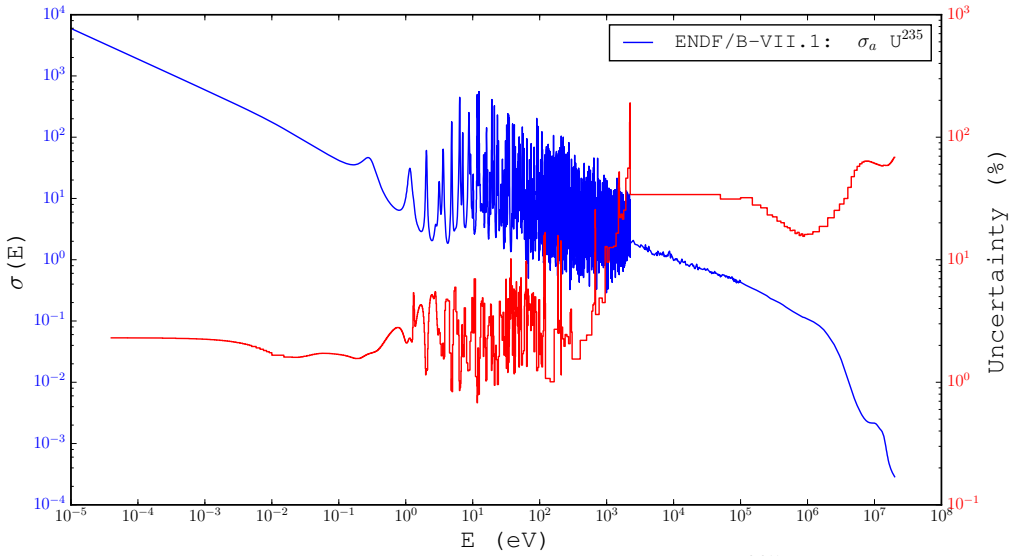

Figure 6: σ_a with corresponding error for ^{235}U

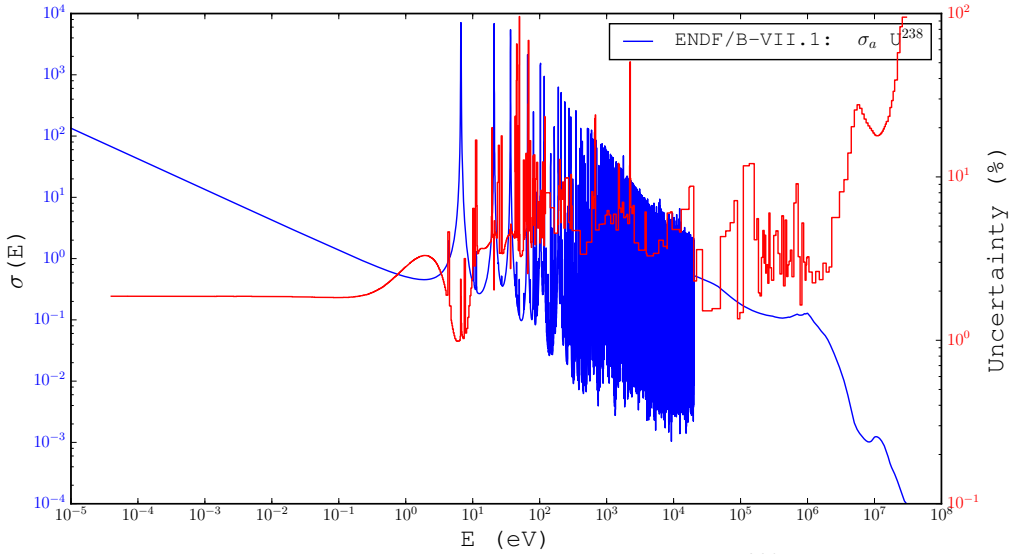

Figure 7: σ_a with corresponding error for ^{238}U

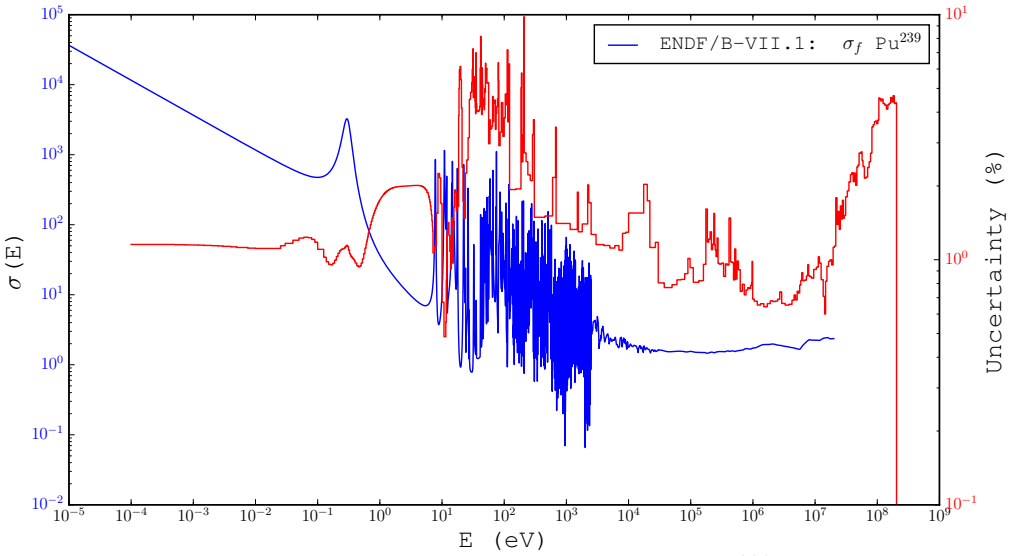

Figure 8: σ_f with corresponding error for ^{239}Pu

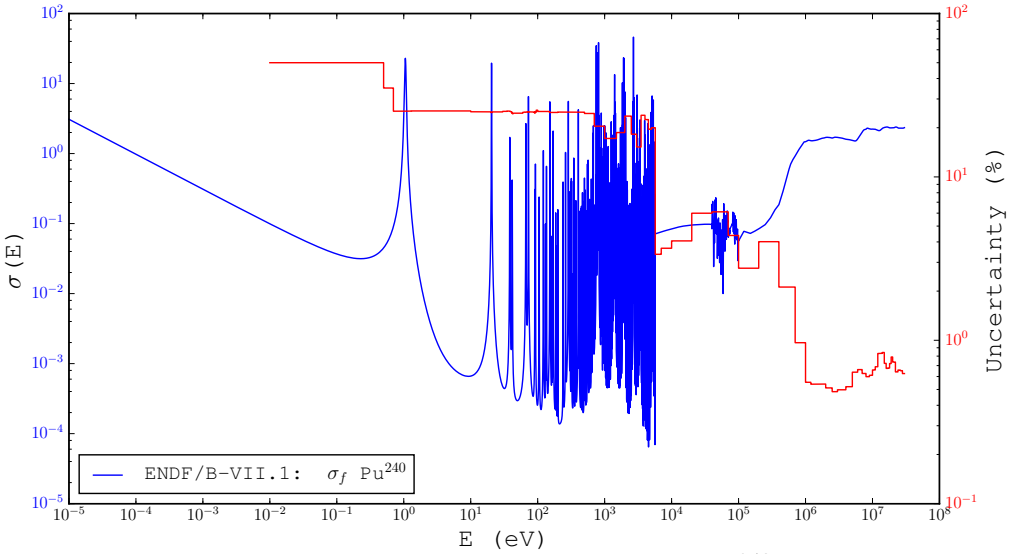

Figure 9: σ_f with corresponding error for ^{240}Pu

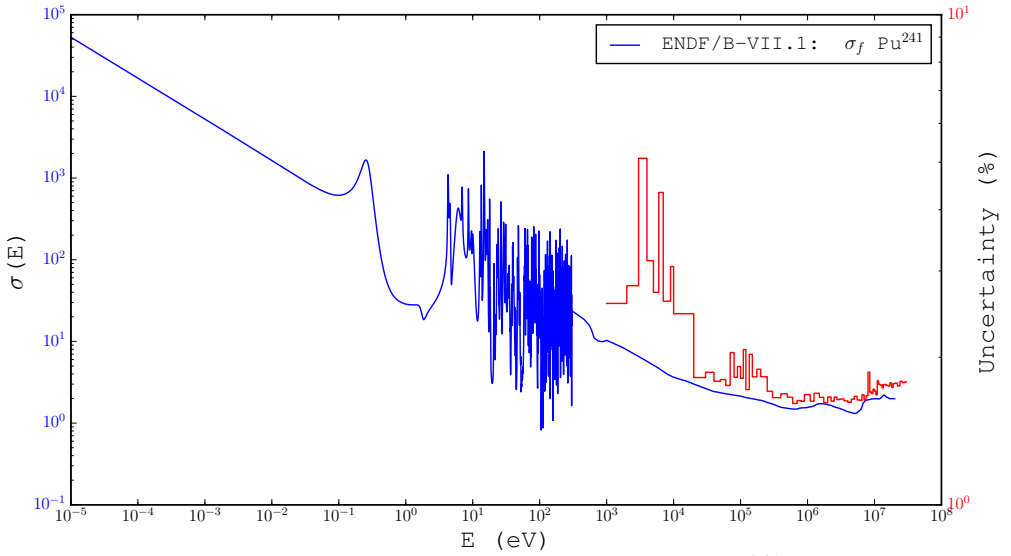

Figure 10: σ_f with corresponding error for ^{241}Pu

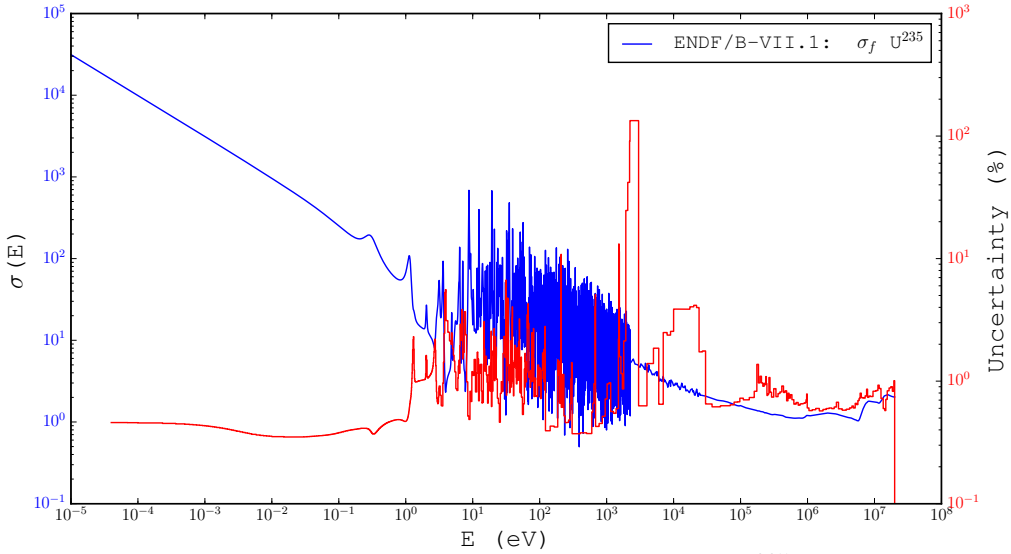

Figure 11: σ_f with corresponding error for ^{235}U

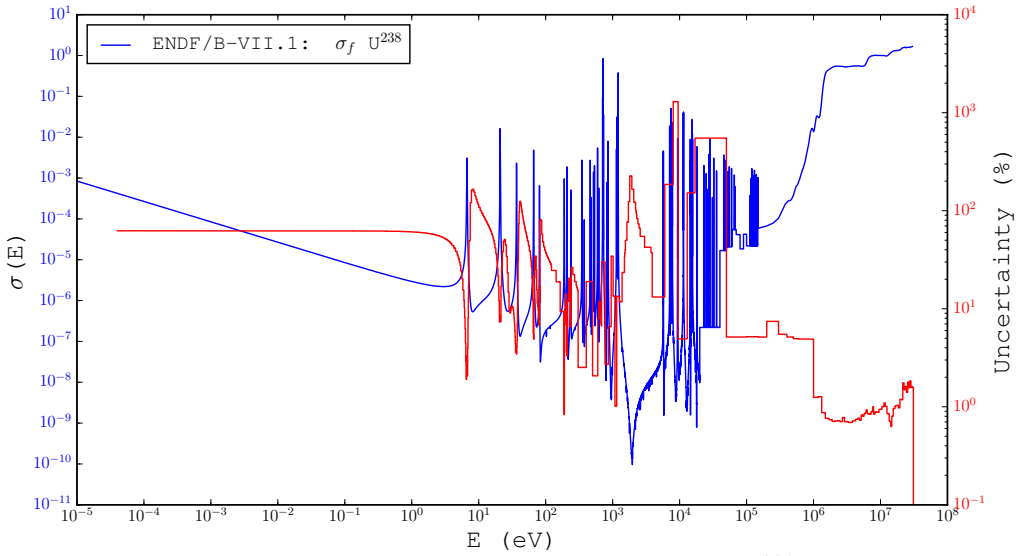

Figure 12: σ_f with corresponding error for ^{238}U

☑ Create a sampling space for all possible variations of calculations

The distribution for all 10 of the above cross sections are assumed as Gamma distributions to avoid negative cross section values.

$$\pi(\theta) = \frac{\theta^{\alpha-1}e^{-\theta/\beta}}{\Gamma(\alpha)\beta^{\alpha}}, \qquad \theta, \alpha, \beta > 0.$$

Therefore, we say that $\theta \sim G(\alpha, \beta)$. Where the mean and errors determined above fit into α and β via

$$\alpha = \frac{\text{Mean}^2}{\text{Error}^2}$$

and

$$\beta = \frac{\text{Error}^2}{\text{Mean}}$$

With the following codes the sampling space was detemined.

Listing 5: Sample Generation code

```
#!/usr/bin/env python3

"""
Will make N random samples and store in files with same
names as isotopes
"""
##########################################################################
######################### Import Packages ################################
##########################################################################

import time
start_time = time.time()
import numpy as np
```

```
import Functions as Fun


#######################################################################
######################### Class For Change ############################
#######################################################################

class OneGroupwError:
    def __init__(self):
        self.Element   = ""
        self.XSec      = 0     #X Section
        self.Err       = 0     #Error


#######################################################################
#################### Grab all X-sections ##############################
#######################################################################

N=Fun.N
Nbins=Fun.Nbins

O=[]
with open('OneGroupXSections') as f:
    Lines=f.readlines()
for i in Lines:
    i=i.split()
    O.append(OneGroupwError())
    O[-1].Element = i[0]
    O[-1].XSec    = float(i[1].split('+/-')[0])
    O[-1].Err     = float(i[1].split('+/-')[1])


for i in O:
    #Make Samples
    alpha=(i.XSec**2)/(i.Err**2)
    beta=(i.Err**2)/(i.XSec)
    Sample=np.random.gamma(alpha,beta,size=N)
    #Make histogram plot
    Fun.PlotHistSave(Sample,N,i.Element,Nbins)
    #Convert Data to string
    Sample=[str(x) for x in Sample]
    #open the output file and save
    output=open("SAMPLES/"+i.Element,"w")
    print("\n".join(Sample),file=output)
    output.close()

#######################################################################
######################### Time to Execute #############################
#######################################################################

print("--- %s seconds ---" % (time.time() - start_time))
```

Below are histograms of the sampling space, most of the others look the same, with samples surrounding the mean.

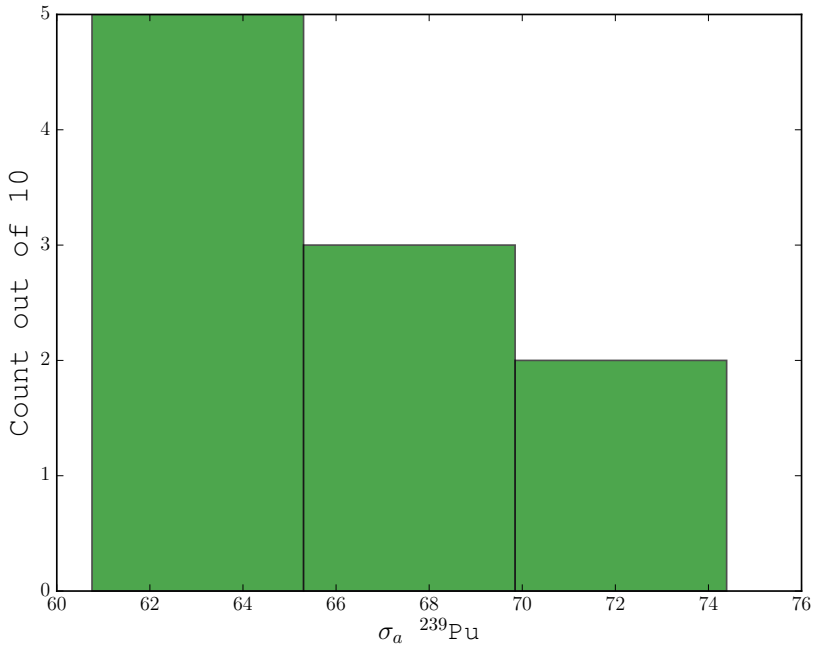

Figure 13: Histogram of sample space for σ_a ^{239}Pu

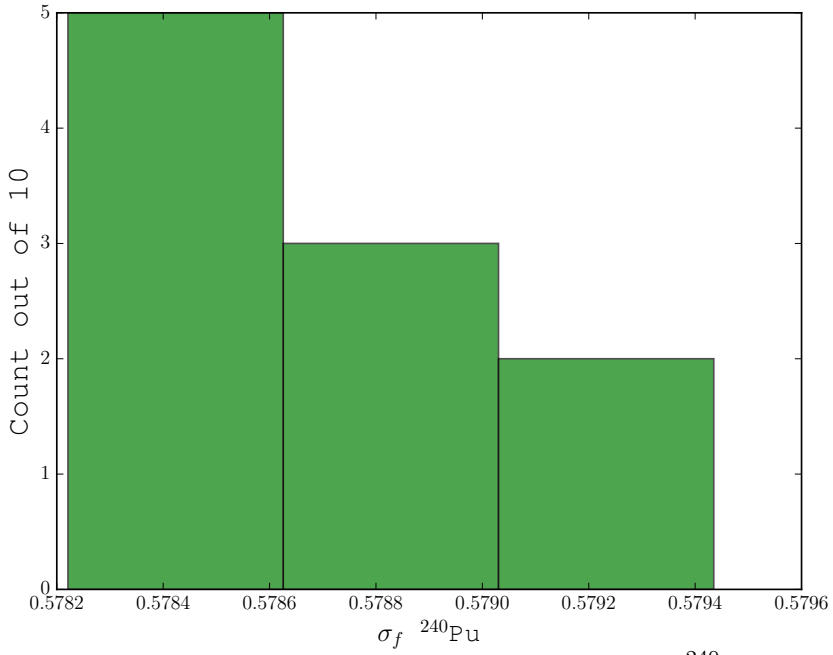

Figure 14: Histogram of sample space for σ_f ^{240}Pu

The sampling spaces were run through ORIGEN with the code shown in the first section.

☑ Plot Results

The function for plotting results are shown below.

Listing 6: Code for plots

```
#!/usr/bin/env python3
"""

"""


########################################################################
######################### Import Packages ##############################
########################################################################

import time
start_time = time.time()
import Functions as Fun
from scipy import interpolate
from scipy import integrate
from scipy.integrate import trapz
import sys
import os
import numpy as np
import copy

########################################################################
############################## Inputs ##################################
########################################################################

#For histogram plots
TimeIndex=3
#Below should be the same as in 'Sample_Gen' file (I know its lame)
N=Fun.N
Nbins=Fun.Nbins

#Which Time Steps for range...
T1=0
Tf=5

#15 Elements, cant do all at once Do 8 at a time (lame again)
#E1=9
#Ef=15
E1=0
Ef=9

########################################################################
############################## Low Class ###############################
########################################################################
#As opposed to high class
class ToPull:
    def __init__(self):
        self.File  = ""

#Gather all output file names
ListToPull=[]
with open('LISTTOPULL') as f:
    Lines=f.readlines()
for i in Lines:
```

```
    i=i.split()
    ListToPull.append(ToPull())
    ListToPull[-1].File =i[0]+'.out'

#######################################################################
############### Orchestral Manoeuvres in the Dark ###################
#######################################################################

#Loop through output file names
for index in range(E1,Ef):
    fileparse=ListToPull[index].File

    #Save output file information
    with open("OUTPUTDATA/"+fileparse) as f:
        Lines=f.readlines()

    #Get time information from first row
    TimeList=Lines[0].replace('\n','').replace('D','').split(',')
    Time=[]
    for times in TimeList:
        Time.append(float(times))

    #Make matrix of data for rest
    Data=[]
    del Data
    for i in range(1,len(Lines)):
        line=Lines[i].replace('\n','')
        line=line.split(',')
        line=[float(i) for i in line]
        try:
            Data=np.vstack((Data,line))
        except NameError:
            Data=copy.copy(line)

    #Plot Time vs grams for all different runs
    HISTSamples=[]
    fig=Fun.plt.figure(figsize=Fun.FigureSize)
    ax=fig.add_subplot(111)
    label=''
    for run in Data:
        HISTSamples.append(run[TimeIndex])
        #print(run) Plot
        (fig,ax)=Fun.plot(Time[T1:Tf],run[T1:Tf],
                          ax,'black',label,fig,fileparse)

    Element=fileparse.split('.')[0]
    Fun.plt.savefig("PLOTS/"+Element+'Post_XY.pdf')
    #Make histogram for specified time for all different runs
    Fun.PlotHistSave2(HISTSamples,N,fileparse,Nbins)

#######################################################################
######################### Time to Execute #############################
#######################################################################

print("--- %s seconds ---" % (time.time() - start_time))
```

III Results

Quantities of interest are shown in Table 1 above. The uncertain parameters were the absorption and fission cross-sections for ^{235}U, ^{238}U, ^{239}Pu, ^{240}Pu, and ^{241}Pu.

Results are graphically represented below.

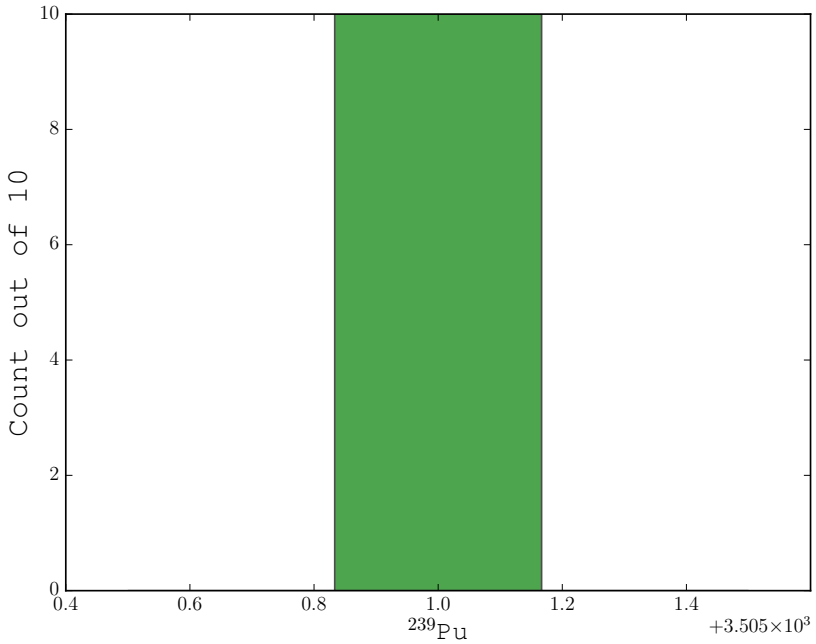

Figure 15: Histogram of QOI ^{239}Pu

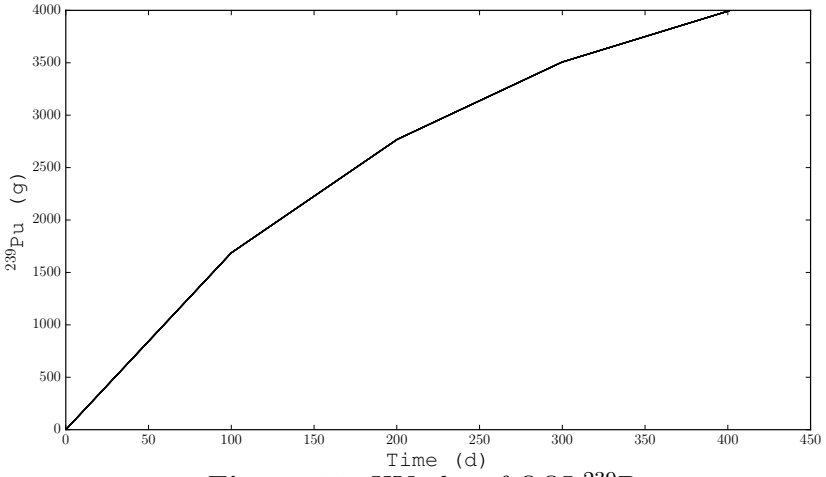

Figure 16: XY plot of QOI ^{239}Pu

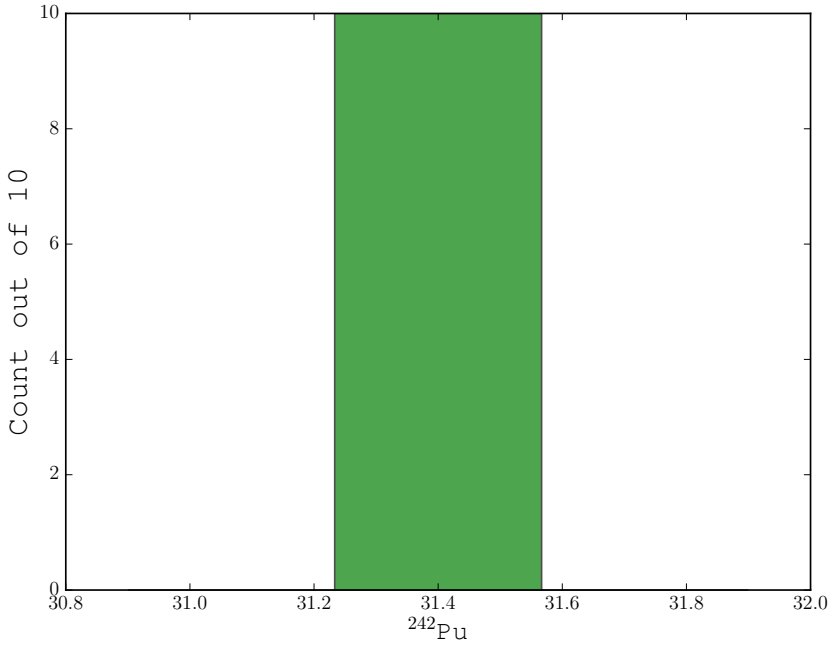

Figure 17: Histogram of QOI ^{242}Pu

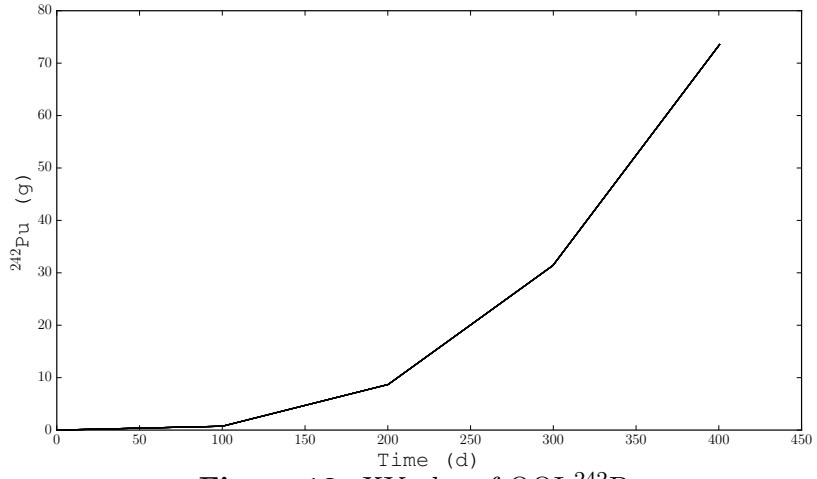

Figure 18: XY plot of QOI ^{242}Pu

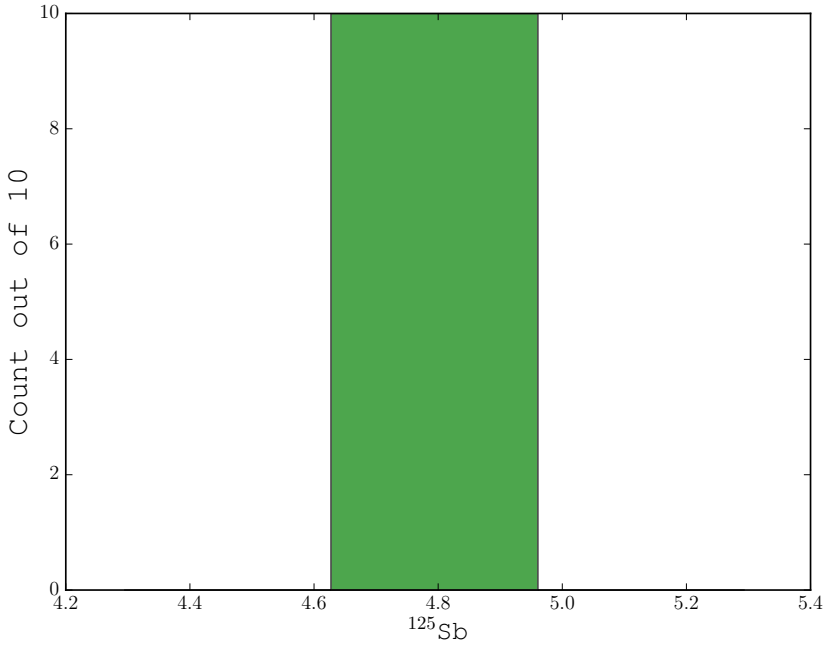

Figure 19: Histogram of QOI ^{125}Sb

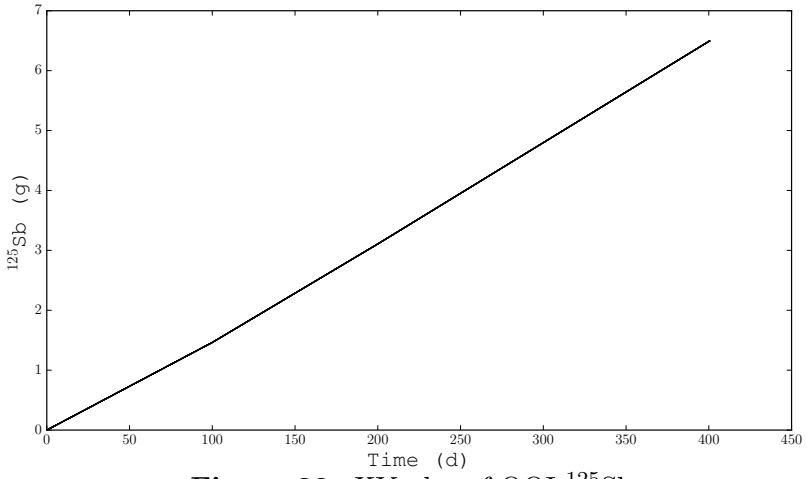

Figure 20: XY plot of QOI ^{125}Sb

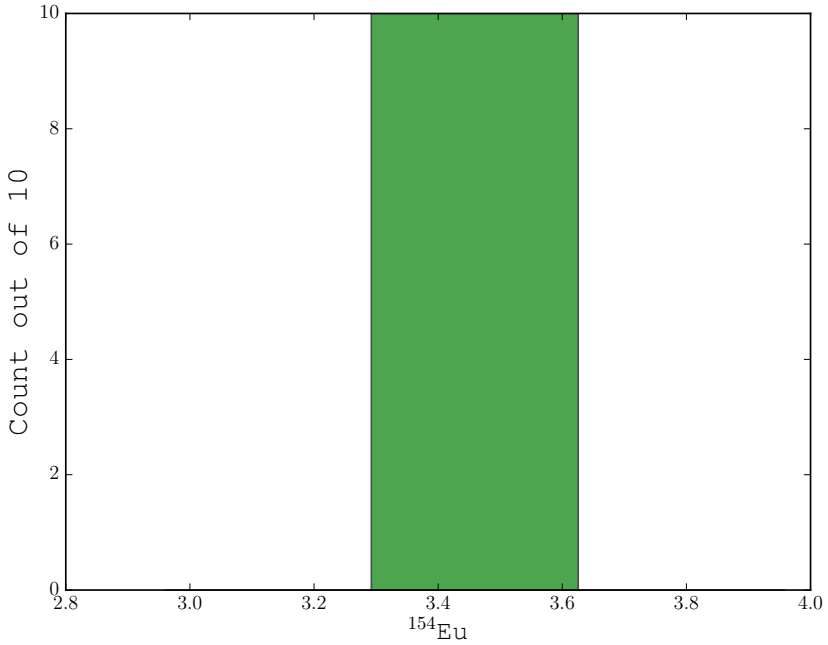

Figure 21: Histogram of QOI ^{154}Eu

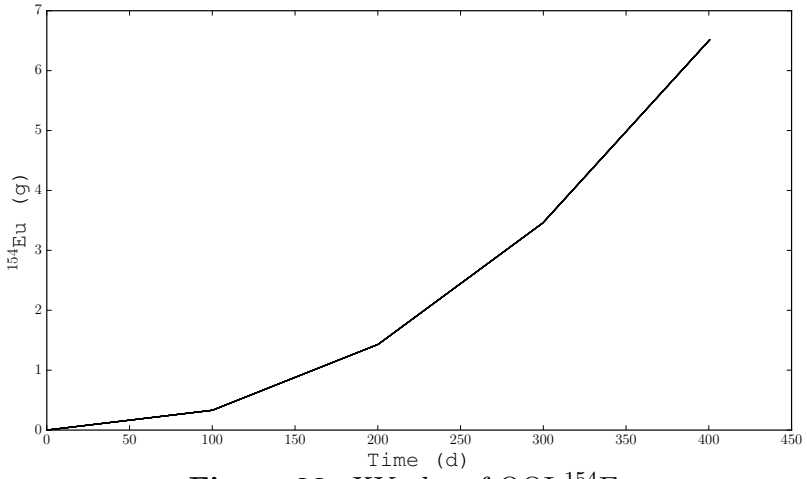

Figure 22: XY plot of QOI ^{154}Eu

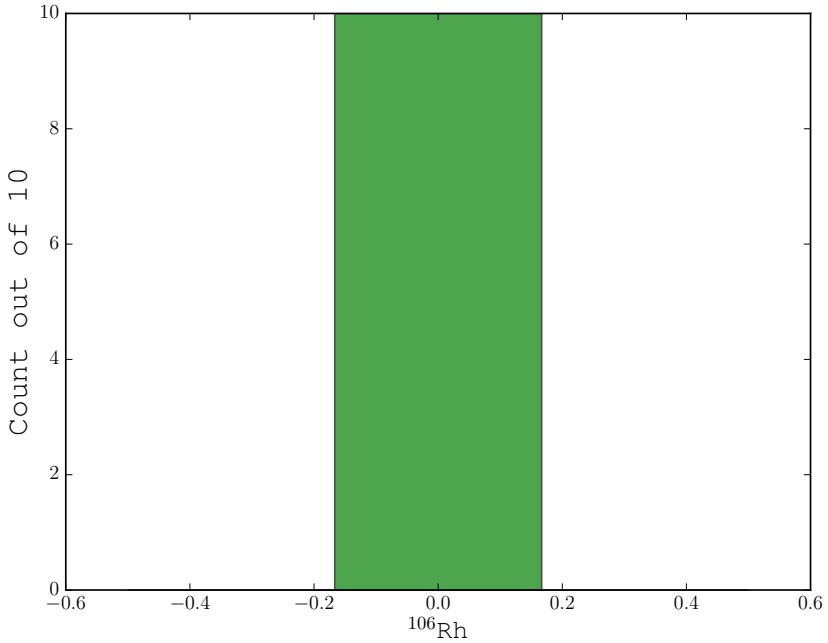

Figure 23: Histogram of QOI ^{106}Rh

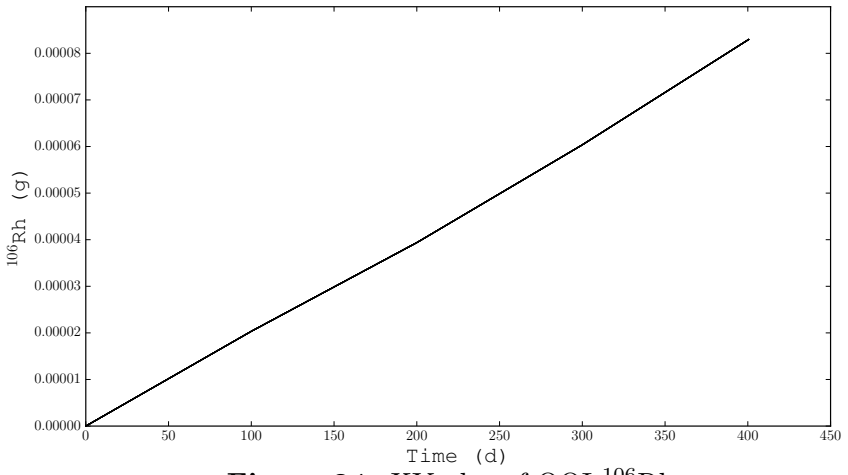

Figure 24: XY plot of QOI ^{106}Rh

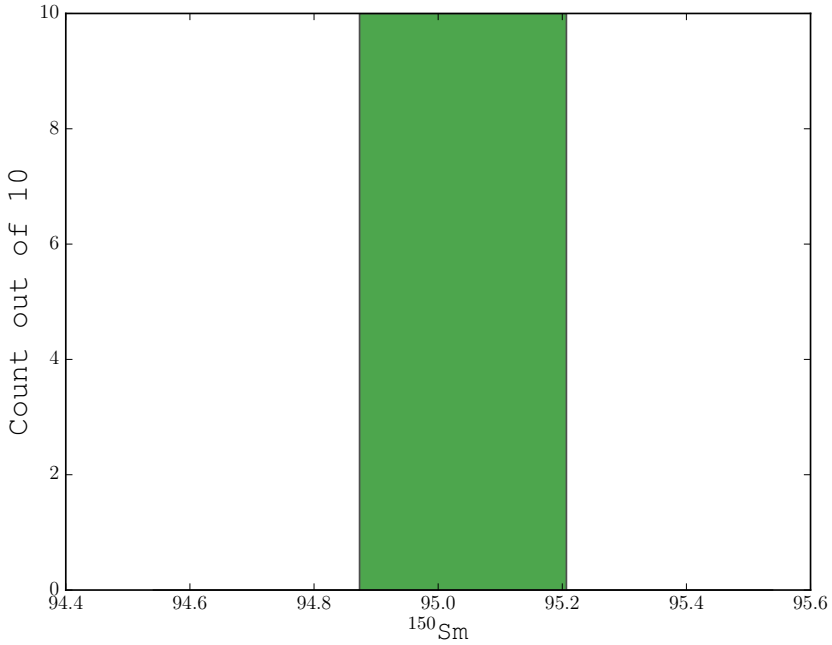

Figure 25: Histogram of QOI ^{150}Sm

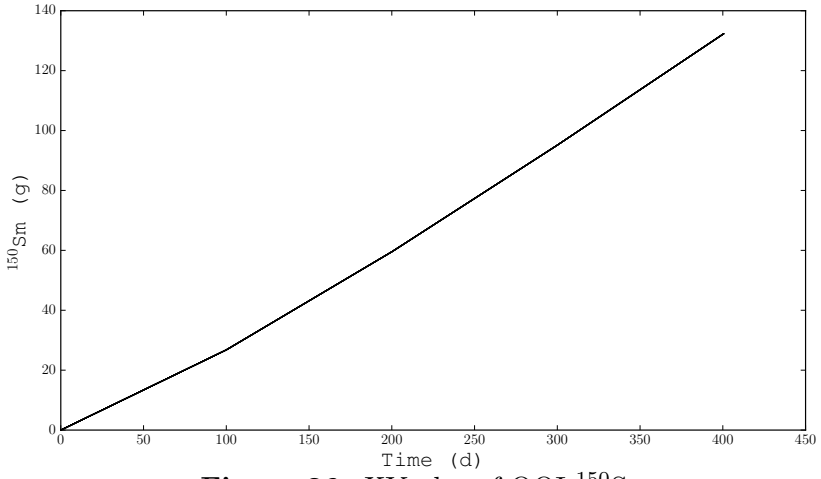

Figure 26: XY plot of QOI ^{150}Sm

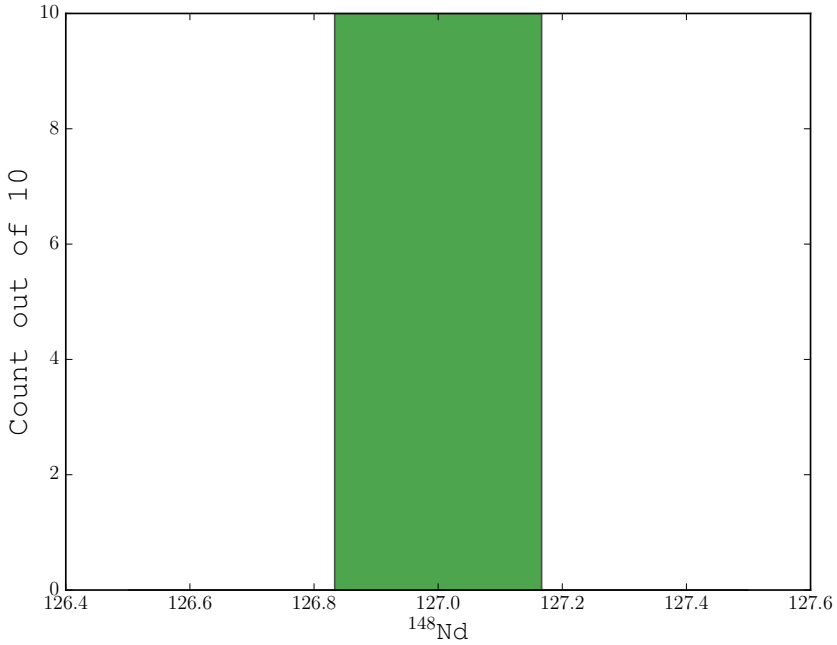

Figure 27: Histogram of QOI ^{148}Nd

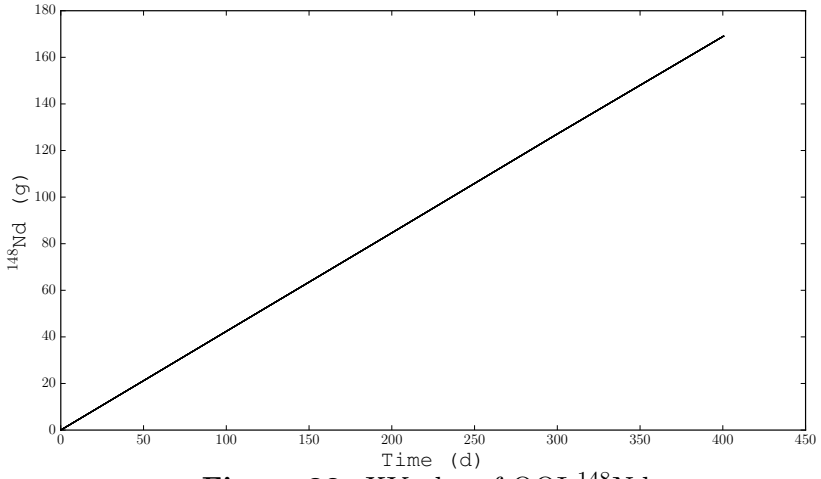

Figure 28: XY plot of QOI ^{148}Nd

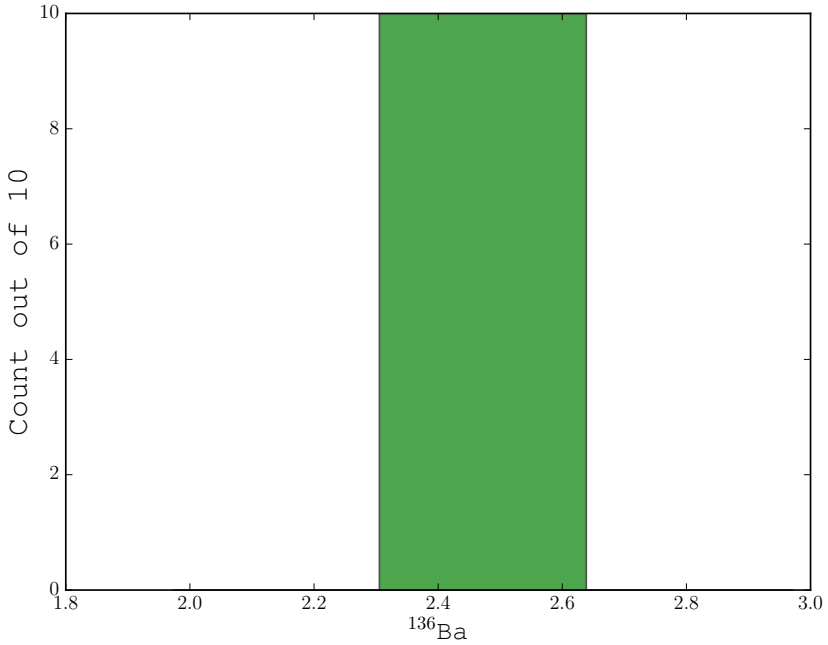

Figure 29: Histogram of QOI ^{136}Ba

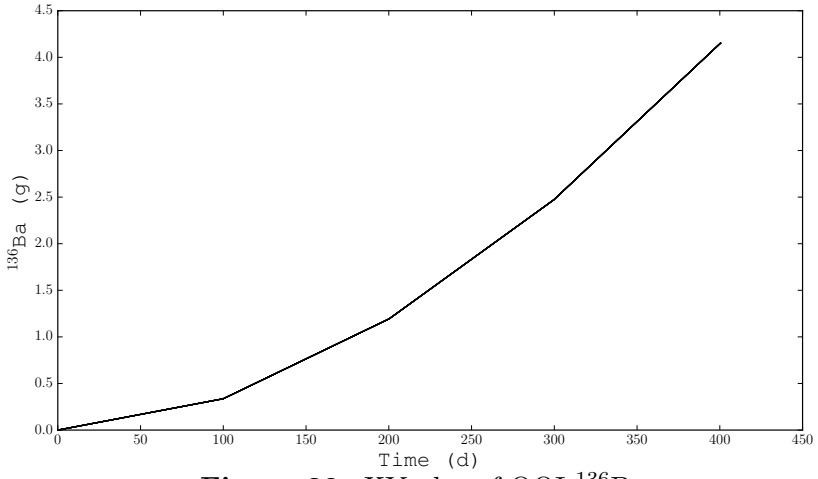

Figure 30: XY plot of QOI ^{136}Ba

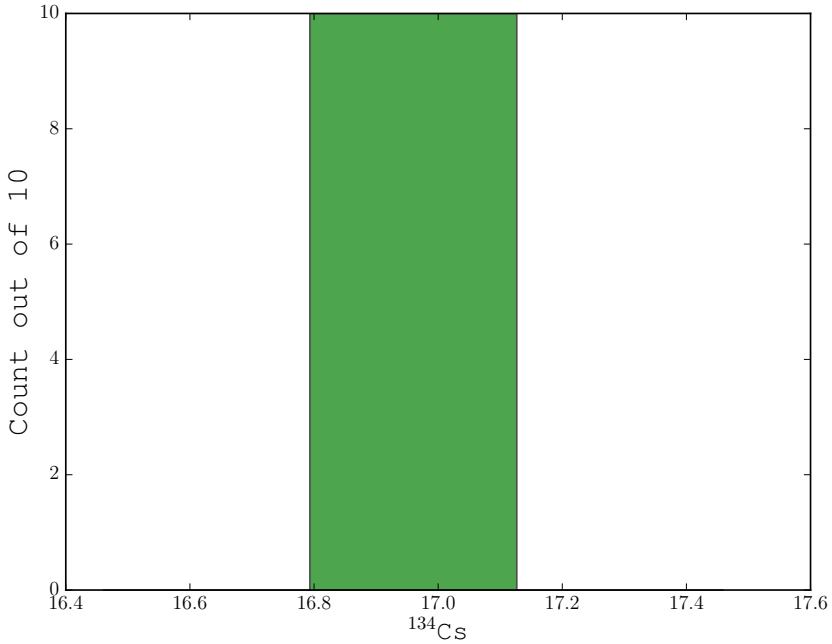

Figure 31: Histogram of QOI ^{134}Cs

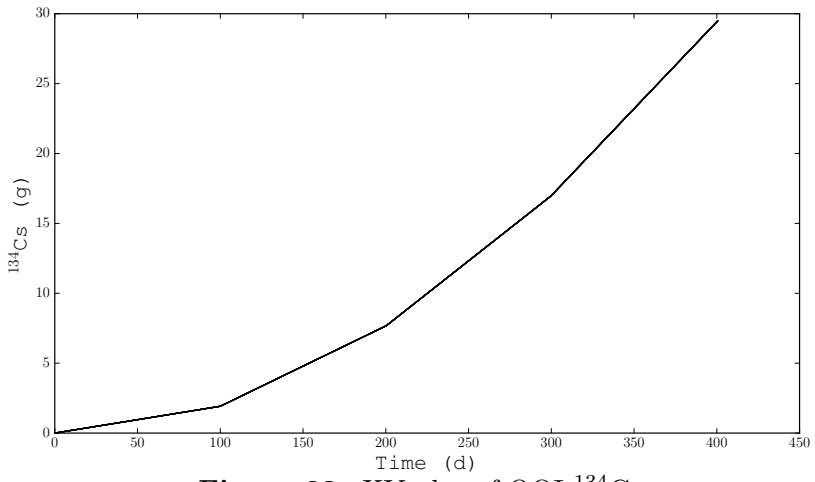

Figure 32: XY plot of QOI ^{134}Cs

I apologize for the wall of graphs. I left some out.

IV Conclusions

There was an unexpectedly large amount of error in the one group cross sections, but currently there is no variation in the output, even when I change the cross section by a factor of over 1000, I don't think ORIGEN is taking up what I am laying down.

References

[1] Allen G Croff. User's manual for the origen2 computer code. Technical report, Oak Ridge National Lab., 1980.