NUEN 629: Homework #5

Thursday, December 3, 2015

Ryan G. McClarren

James B. Tompkins

Problem 1

The requested activation and decay chains may be found in Cadwallader and Longhurst's 1999 report (Cadwallader, L. & Longhurst, G. (1999). FLiBe Use in Fusion Reactors: An Initial Safety Assessment. *Idaho National Engineering and Environmental Laboratory.*). The depletion calculations performed in subsequent problems primarily involves these activation and decay chains, but additional chains are considered for the listed products as well.

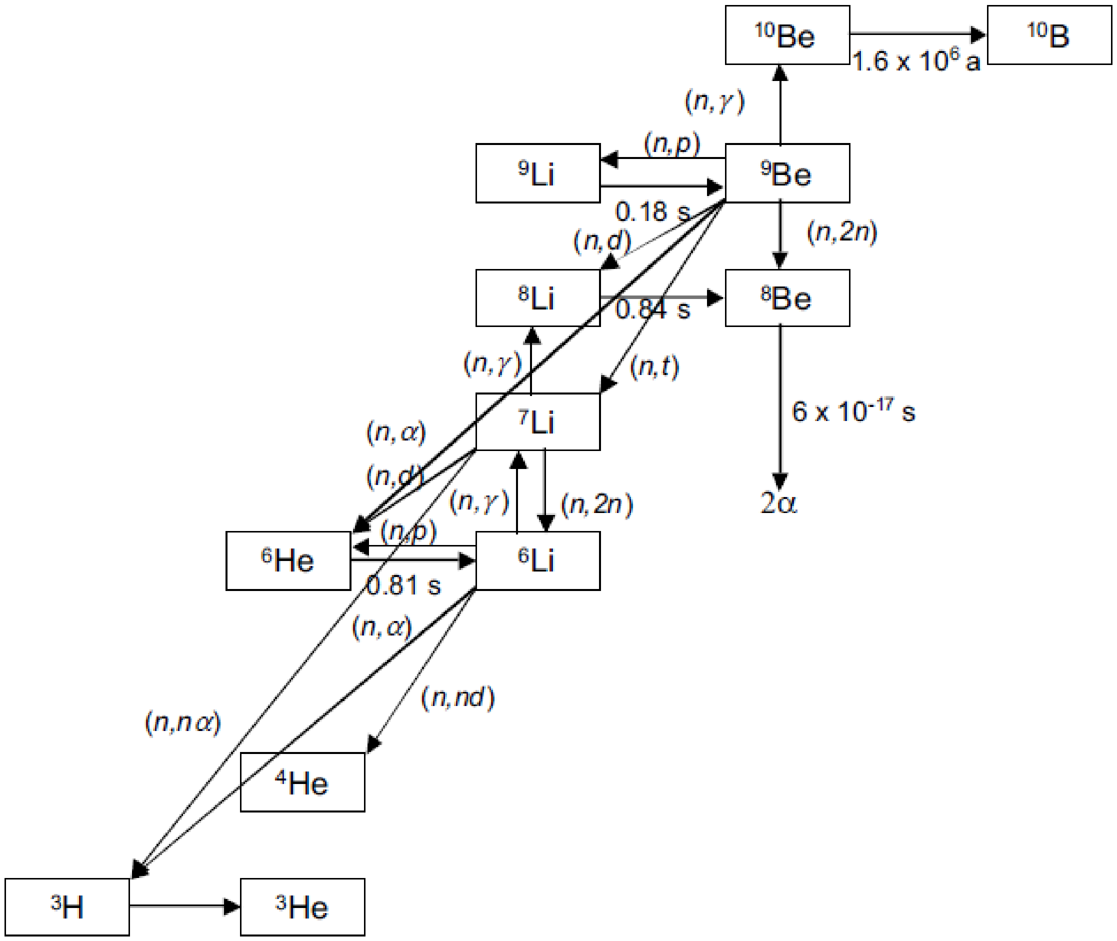

Figure 1: Lithium and beryllium activation and decay chain (Cadwallader & Longhurst)

Problem 1 continued on next page. . .

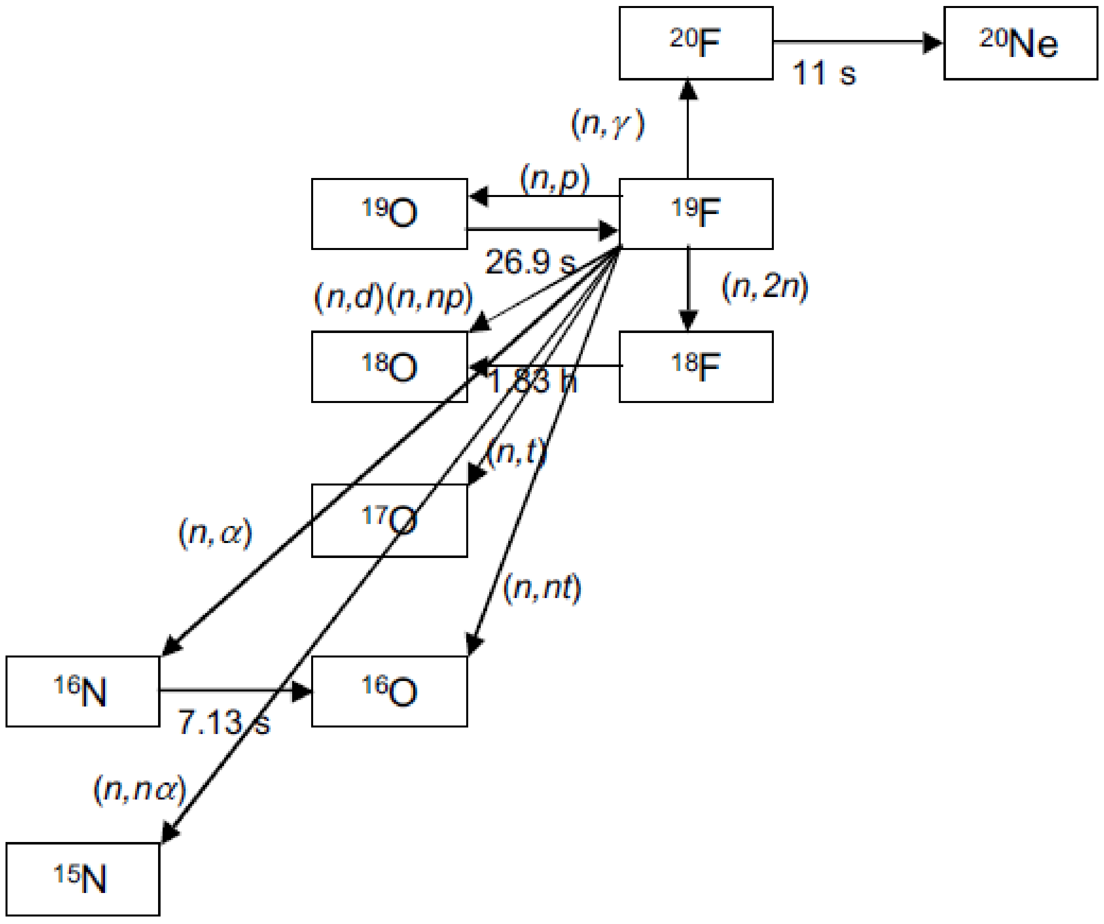

Figure 2: Fluorine activation and decay chain (Cadwallader & Longhurst)

Problem 1 continued on next page. . .

Problem 2

We are asked to solve the given depletion problem by calculating the concentration of tritium present in the FLiBe blanket around a fusion reactor after a two year irradation period. First, the cross sections and half lives of the nuclides involved in the decay chains are found using OECD's Janis software, then creating an activation and decay matrix to use in finding the solution to this problem. The following listing details the backward Euler solution method.

Listing 1: Backward Euler Solution Method

```
# Backward Euler Method
time_iters = range(0,75)
iter_length = 9.74
ts = np.arange(0.0,731.0,9.74)
u_euler = np.zeros(len(nuclides))
u_set_euler = np.zeros((len(time_iters)+1,len(nuclides)))
u_set_euler[0,:] = b


b_new = b
for i_iter in time_iters:
    i_iter += 1
    u_euler = np.dot(np.linalg.inv(np.identity(len(nuclides)) - \
                A*iter_length),b_new)
    #print(u_euler)
    u_euler = np.absolute(u_euler)
    u_set_euler[i_iter,:] = u_euler
    b_new = u_euler
```

The calculated tritium concentration in the FLiBe blanket after two years of irradiation is 4.291837 x 10^5 kg tritium per kg FLiBe original material. Thirty-three isotopes were tracked in this simulation, and results over time for the fourteen nuclides with largest concentration at the end of the irradation cycle are given in Figures 3 and 4.

Problem 2 continued on next page. . .

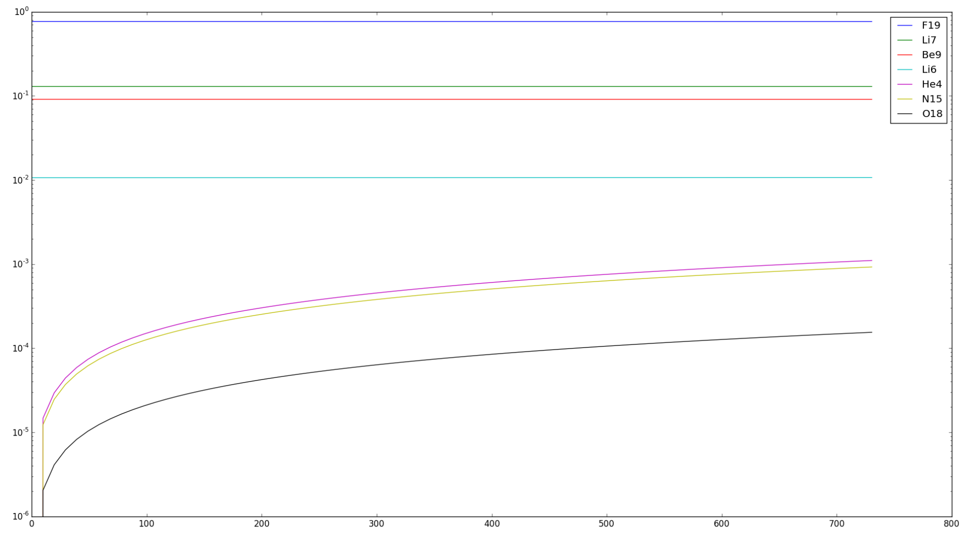

Figure 3: Nuclides with largest concentration after the two year irradiation period from backward Euler method.

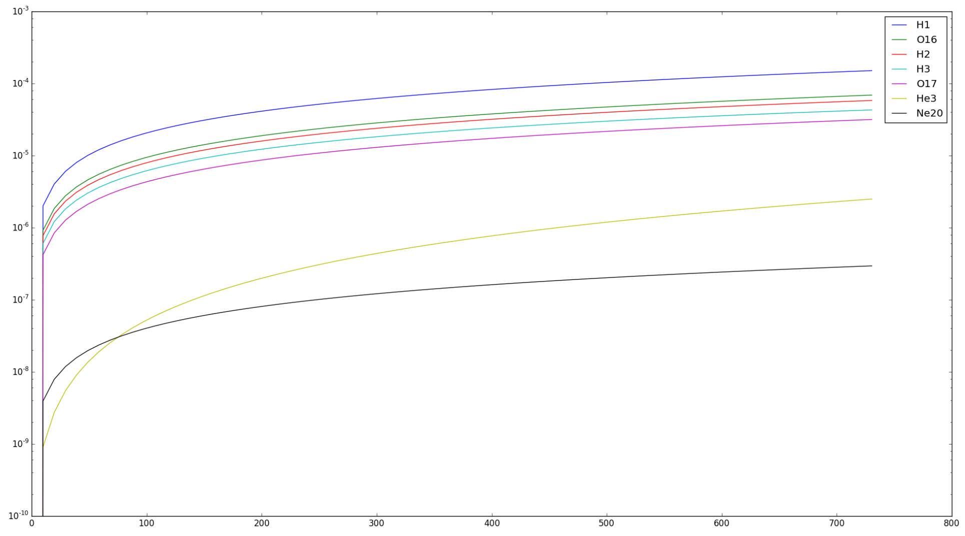

Figure 4: Nuclides with next largest concentrations after irradiation from backward Euler method.

Problem 2 continued on next page. . .

Next, the matrix exponential method is used to solve the depletion problem.

Listing 2: Matrix Exponential Solution Method

```
# Matrix Exponential Method
ts = np.arange(0., 730.51, 1.2175)
print(len(ts))
t_iter = 0
u_set_matexp = np.zeros((len(ts),len(nuclides)))


for t in ts:
    u_matexp = np.zeros(len(nuclides))
    temp = sp.linalg.expm(A*t)
    u_matexp = np.dot(temp,b)
    u_set_matexp[t_iter,:] = u_matexp
    t_iter += 1
```

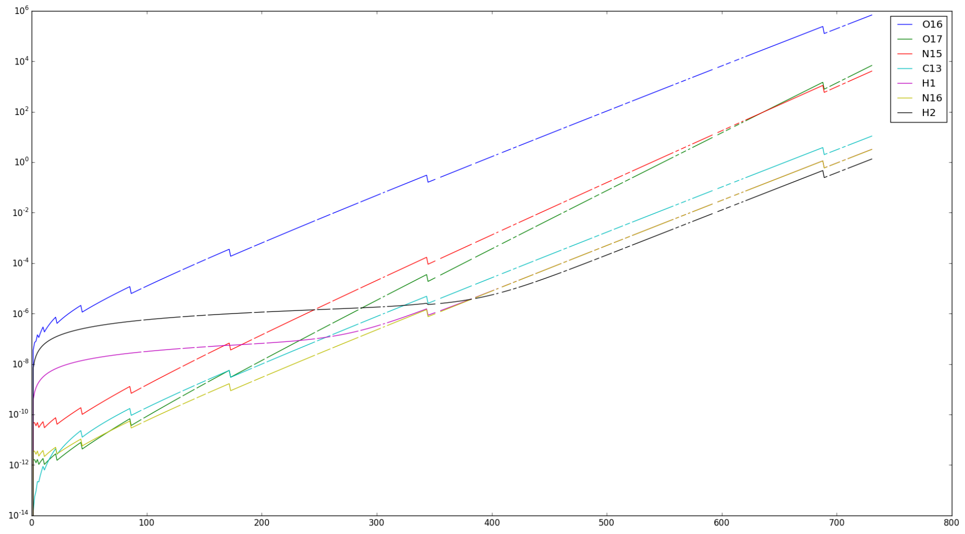

Figure 5: Nuclides with largest concentration from matrix exponential method.

This method gives a tritium concentration of 0.0078297 kg tritium per kg FLiBe, a significant increase from the previous estimation. The concentrations of all isotopes increases and goes above 1 suggesting that there is a problem with how this method is solving the problem (more likely, just how I have set up the problem). The kinks in the solution lend further evidence to the questionability of this method's implementation. Two solutions to fixing this implementation are to neglect some of the fast decaying (Be-8) isotopes and take shorter time steps.

The quadrature approximated matrix exponential method attempts to approximate the matrix exponential solution by the best rational method producing quadrature points from residue calculus.

Listing 3: Quadrature Approximated Matrix Exponential Solution Method

```
# Matrix Exponential via Quadrature Method
ts = np.arange(0.0,730.51,9.74)
t_iter = 0
N = 32
#u_quad = np.zeros(len(nuclides))
u_set_quad = np.zeros((len(ts),len(nuclides)))

for t in ts:
    theta = np.pi*np.arange(1,N,2)/N
    z = N*(.1309 - 0.1194*theta**2 + .2500j*theta)
    w = N*(- 2*0.1194*theta + .2500j)
    c = 1.0j/N*np.exp(z)*w
    #plt.plot(np.real(z),np.imag(z),'o-')
    #plt.savefig('hw5.pdf')
    #plt.close()
    u_quad = np.zeros(len(nuclides))
    for k in range(int(N/2)):
        n,code = splinalg.gmres(z[k]*sparse.identity(len(nuclides))  - A*t,
                  b, tol=1e-12, maxiter=2000)
        if (code):
            print(code)
        u_quad = u_quad- c[k]*n
    u_quad = 2*np.real(u_quad)
    u_quad = np.absolute(u_quad)
    u_set_quad[t_iter,:] = u_quad
    t_iter += 1
    #print(u_quad)
```

This method returns results for tritium concentration consistent with the backward Euler method (4.312772 x 10^5 kg tritium per kg FLiBe) and maintains the dominance of the four natural isotopes present, but results for built up isotopes are very erratic. Though the concentration solutions seem to oscillate around mean results similar to the backward Euler method, the error clearly dominates this implementation. Suggestions for making this method work are the same listed for matrix exponential method.

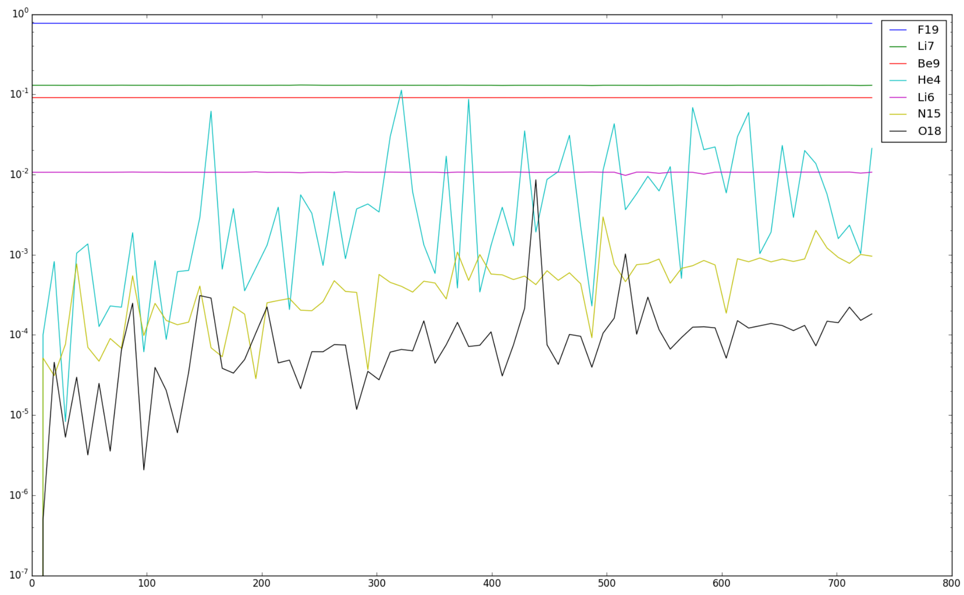

Figure 6: Nuclides with largest concentration from quadrature matrix exponential method.

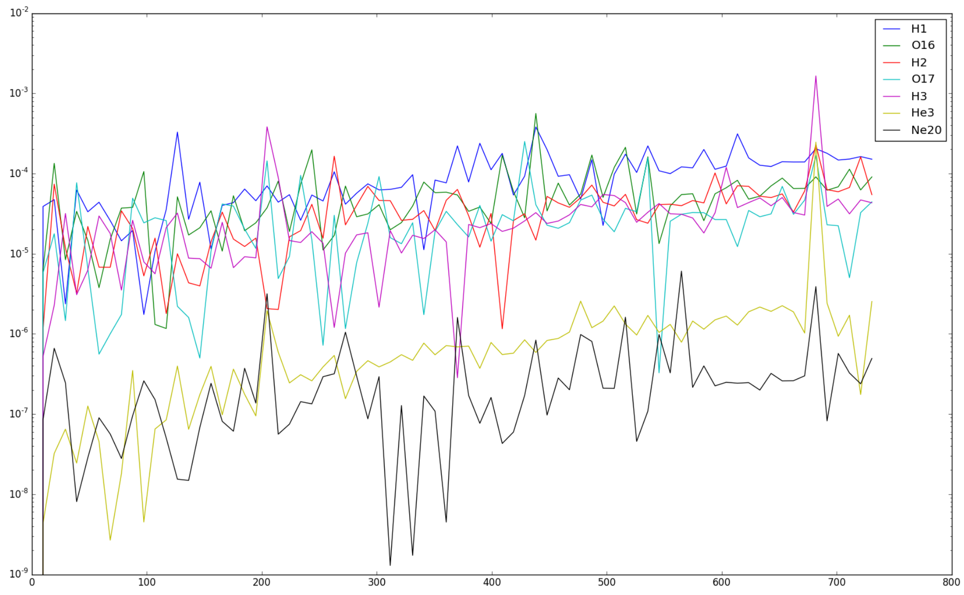

Figure 7: Nuclides with next largest concentrations from quadrature matrix exponential method.

Problem 2 continued on next page. . .

Problem 3

Listing 4: Decay of Activation Products

```python
A_d = np.zeros((len(nuclides),len(nuclides)))

for isotope in nuclides:
    row = nuclides[isotope]
    row_betanegdecay =  betanegdecay(nuclides, isotope)
    row_betaposdecay =  betaposdecay(nuclides, isotope)
    row_2alphadecay =   twoalphadecay(nuclides, isotope)
    if row_betanegdecay[0] >= 0:
        row_lambda = log(2)*60*60*24/row_betanegdecay[1] # [days^-1]
    elif row_betaposdecay[0] >= 0:
        row_lambda = log(2)*60*60*24/row_betaposdecay[1] # [days^-1]
    elif row_2alphadecay[0] >= 0:
        row_lambda = log(2)*60*60*24/row_2alphadecay[1] # [days^-1]
    else:
        row_lambda = 0.0
    # Diagonal Assignment
    A_d[row,row] = -row_lambda
    # Off Diagonal Assignment
    if row_betanegdecay[0] >= 0:
        A_d[row_betanegdecay[0],row] = log(2)*60*60*24/row_betanegdecay[1]
    if row_betaposdecay[0] >= 0:
        A_d[row_betaposdecay[0],row] = log(2)*60*60*24/row_betaposdecay[1]
    if row_2alphadecay[0] >= 0:
        A_d[row_2alphadecay[0],row] = log(2)*60*60*24/row_2alphadecay[1]


b_d = u_euler

time_iters = range(0,500)
iter_length = 16524.
ts = np.arange(0.0,8.2620000001e6,16524.)
u_decay = np.zeros(len(nuclides))
b_new = b_d
Activities = np.zeros(len(ts))
Activities[0] = sum((6.0221409e23/atom_mass)*b_d*decay_consts)
print(Activities[0])
#u_set_decay = np.zeros((len(time_iters)+1,len(nuclides)))


for t_iter in time_iters:
    t_iter += 1
    u_decay = np.dot(np.linalg.inv(np.identity(len(nuclides)) - \
              A_d*iter_length), b_new)
    #print(u_decay)
    u_decay = np.absolute(u_decay)
    Activities[t_iter] = sum((6.0221409e23/atom_mass)*u_decay*decay_consts)
    print(Activities[t_iter])
    b_new = u_decay
```

Problem 3 continued on next page. . .

Using the results from the backward Euler solution as initial quantities of materials and constructing a new matrix based solely on the decays of tracked nuclides, the backward Euler method is applied to find how long it will take for the discharged FLiBe blanket to decay to below the activity of Brazil nuts (444 Bq/kg). The listing above demonstrates constructing the new decay matrix and solution method. It ends up taking about 22,620 years for the isotopes present in the blanket to reach an activity of 420.15028 Bq/kg demonstrating the importance of considering long lived decay products even in fuels considered for "clean" reactors utilizing fusion.

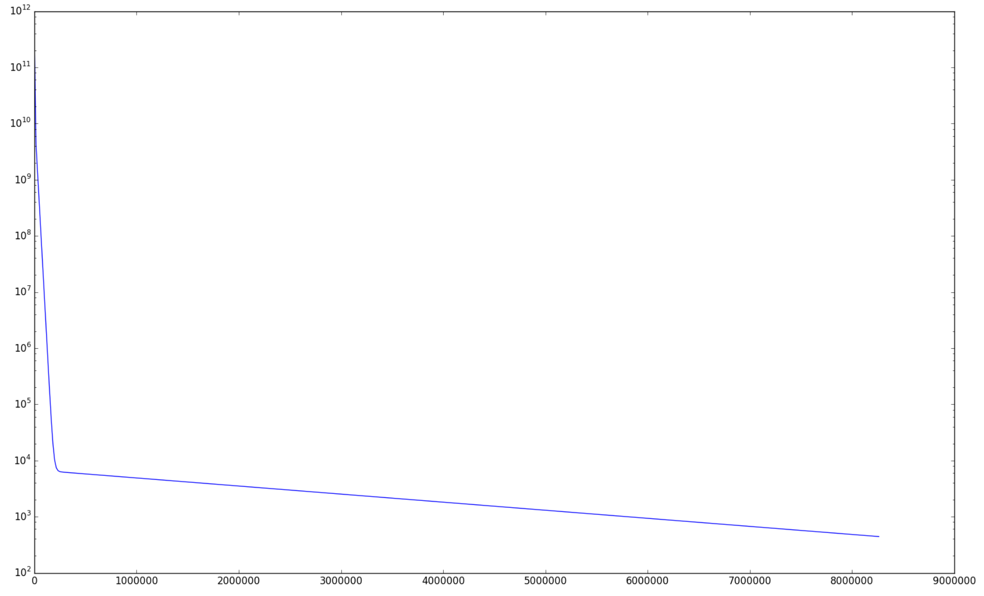

Figure 8: Activity of FLiBe blanket plotted against days of decay.