Problem Sheet 1

FYSC22

Author
Max Eriksson
maxerikss@gmail.com

Lund University
Department of Physics

April 3, 2024

1 First Exercise

(a)

Solution: We know that

$$\left.\frac{\partial M(A,Z)}{\partial Z}\right|_A = 0 \tag{1}$$

for the minimum mass and from the lecture notes this gives

$$Z_{\text{min}}(A) \approx \frac{A}{1.98 + 0.015A^{2/3}}. \tag{2}$$

We now want to find for which A we have $Z_{\text{min}}(A) = 28$ and $N = \text{even}$, where N is the number of neutrons. Using the code found in App. A.1, the most stable nickel nuclide is ^{62}Ni.

Answer: The most stable nickel nuclide is ^{62}Ni.

(b)

Solution: From the lecture it is known that the neutron drip line appears at

$$S_{\text{n}} = B(A,Z) - B(A-1,Z) = 0 \tag{3}$$

and the proton drip line appears at

$$S_{\text{p}} = B(A,Z) - B(A-1,Z-1) = 0, \tag{4}$$

where $B(A,Z)$ is the binding energy which can be calculated from the Weizsäcker mass formula where it is given as

$$B(A,Z) = a_{\text{v}}A - a_{\text{s}}A^{2/3} - a_{\text{c}}\frac{Z^2}{A^{1/3}} - a_{\text{ass}}\frac{(N-Z)^2}{A} + a_{\text{p}}A^{-1/2}, \tag{5}$$

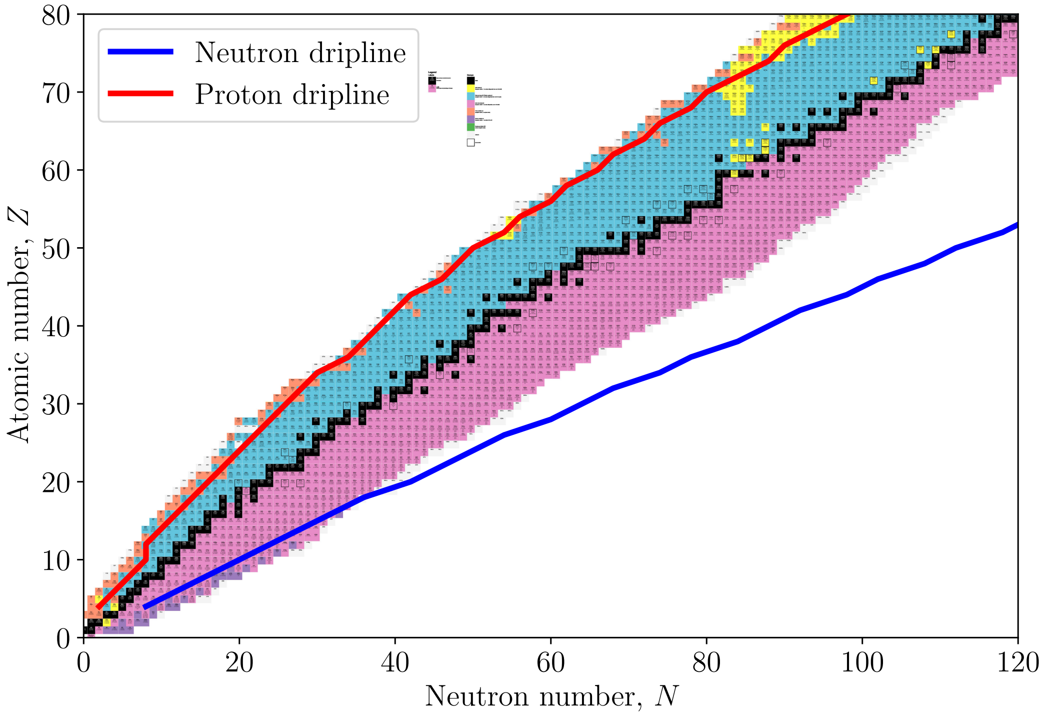

Figure 1: The neutron and proton drip lines plotted for $Z \in [4, 80]$

where $N = A - Z$ and the values for the constants are $a_v = 15.9$ MeV, $a_s = 18.4$ MeV, $a_c = 0.71$ MeV, $a_{ass} = 23.2$ MeV, and $a_p = 11.5$ MeV. Implementing these equations in Python gives that the neutron drip line appears at ^{88}Ni while the proton drip line appears at ^{52}Ni. The code can be seen in App. A.2. For fun, the code was run for $Z \in [4, 80]$ and the result was plotted against the nuclide chart. Both analyzing the actual chart as well as looking at Fig. 1 the calculation seems to give a bit too high of an A, but is reasonably accurate.

Answer: The even-even Nickel nuclide that denotes the neutron drip line is ^{88}Ni and the proton drip line is ^{52}Ni

2 Second Exercise

(a)

Solution: Consulting Feynman's lectures of physics the energy stored in a uniformly charged sphere is

$$E = \frac{3}{5}\frac{Q^2}{4\pi\epsilon_0 R} \tag{6}$$

where Q is the total charge, ϵ is the vacuum permittivity, and R is the radius of the sphere. Since $R = r_0 A^{1/3}$ the equation may be written as

$$E = \frac{3}{5}\frac{Q^2}{4\pi\epsilon_0 A^{1/3}}\frac{1}{r_0}. \tag{7}$$

The energy difference

$$\Delta E = [M(A,Z) - M(A,Z-1)]c^2 \tag{8}$$

may then be written as

$$\Delta E = \frac{3}{5}\frac{Z^2e^2}{4\pi\epsilon_0 A^{1/3}}\frac{1}{r_0} - \frac{3}{5}\frac{(Z-1)^2e^2}{4\pi\epsilon_0 A^{1/3}}\frac{1}{r_0} = \frac{3}{5}\frac{e^2}{4\pi\epsilon_0 A^{1/3}}\frac{1}{r_0}[Z^2 - (Z-1)^2] \tag{9}$$

$$= \frac{3}{5}\frac{(2Z-1)e^2}{4\pi\epsilon_0 A^{1/3}}\frac{1}{r_0}. \tag{10}$$

Since $N = Z - 1$ we get that $A = 2Z - 1$ which then makes

$$\Delta E = \frac{3e^2A^{2/3}}{20\pi\epsilon_0}\frac{1}{r_0}. \tag{11}$$

Answer: The relationship between the energy difference and $1/r_0$ is

$$\Delta E = \frac{3e^2A^{2/3}}{20\pi\epsilon_0}\frac{1}{r_0}. \tag{12}$$

(b)

${}^{A}_{Z}$X	Mass excess [µu]	ΔE [MeV]	${}^{A}_{Z}$X	Mass excess [µu]	ΔE [MeV]
${}^{11}_{5}$B	9305	933.48	${}^{15}_{7}$N	109	934.25
${}^{11}_{6}$C	11 434		${}^{15}_{8}$O	3065	
${}^{19}_{9}$F	−1597	934.74	${}^{23}_{11}$Na	−10 230	935.56
${}^{19}_{10}$Ne	1880		${}^{23}_{12}$Mg	−5875	
${}^{29}_{14}$Si	−23 505	936.44	${}^{35}_{17}$Cl	−31 147	937.47
${}^{29}_{15}$P	−18 199		${}^{35}_{18}$Ar	−24 743	
${}^{41}_{20}$Ca	−37 722	938.00	${}^{45}_{22}$Ti	−41 876	938.63
${}^{41}_{21}$Sc	−30 749		${}^{45}_{23}$V	−34 218	

Table 1: The error is generally less than 1 µu, so this is the uncertainty that will be used in the error calculations. The calculation of ΔE according to Eq. (8) and the conversion to MeV is done in the code in App. A.3.

(c) & (d)

Solution: Rewriting Eq. (12) with $X = (3e^2 A^{2/3})/(20\pi\epsilon_0)$ as

$$\Delta E = \frac{1}{r_0} X, \tag{13}$$

then by calculating ΔE from Tab. 1 and Eq. (8) and doing a linear regression and plotting the result gives $r_0 = 1.28\,\mathrm{fm} \pm 0.06\,\mathrm{fm}$ with a 95.4 % confidence rating. From the lectures we know that $r_0 \approx 1.2\,\mathrm{fm}$, and the calculated value seems reasonable. However, the y-intercept is not at $y = 0$ which is a bit weird.

Answer: The value of r_0 is $r_0 = 1.28\,\mathrm{fm} \pm 0.06\,\mathrm{fm}$

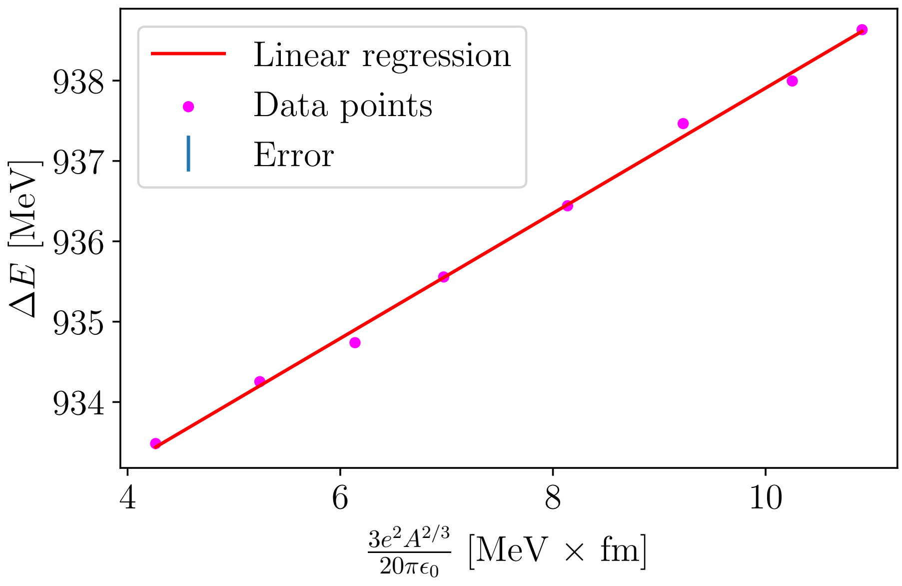

Figure 2: Data points and a linear regression. The error bars are not visible since they are smaller than the scatter-markers. For the code see App. A.3.

3 Third Exercise

(a)

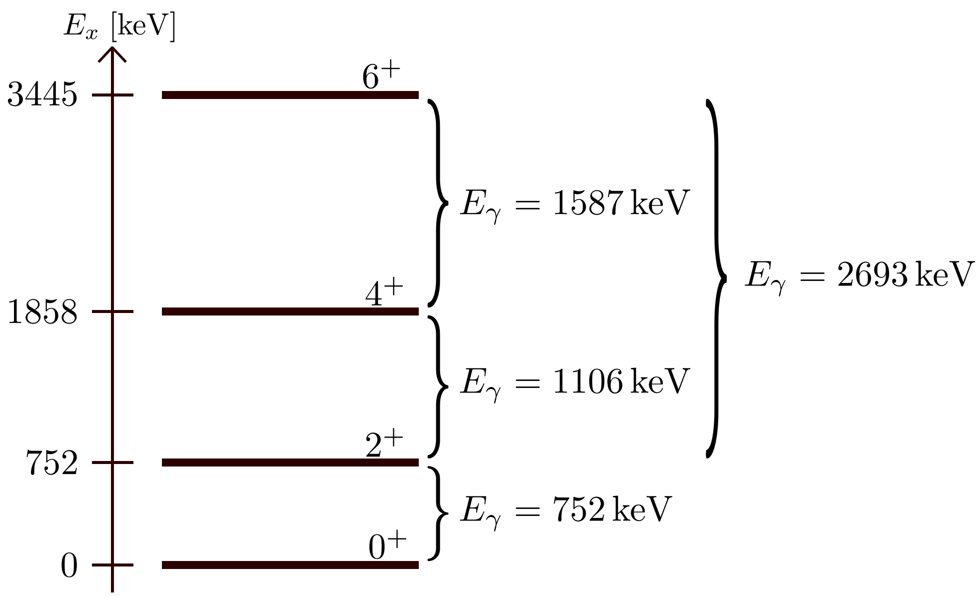

Figure 3: The level sequence of $^{48}_{24}$Cr.

(b)

Solution: The energy for a rotational state is given by

$$E = \frac{\hbar^2 I(I+1)}{2\mathcal{I}}, \tag{14}$$

then we get that

$$\begin{aligned} E_\gamma = E(I+2) - E(I) &= \frac{\hbar^2 (I+2)(I+3)}{2\mathcal{I}} - \frac{\hbar^2 I(I+1)}{2\mathcal{I}} && (15) \\ &= \frac{\hbar^2}{2\mathcal{I}}[(I+2)(I+3) - I(I+1)] = \frac{\hbar^2}{2\mathcal{I}}[(I^2+5I+6) - (I^2+I)] && (16) \\ &= \frac{\hbar^2(4I+6)}{2\mathcal{I}} = \frac{\hbar^2(2I+3)}{\mathcal{I}} && (17) \end{aligned}$$

Thus,

$$E_\gamma = \frac{\hbar^2(2I+3)}{\mathcal{I}}. \tag{18}$$

(c)

Solution: Solving for $\mathcal{I}$ in Eq. (18) we obtain

$$\mathcal{I} = \frac{\hbar^2(2I+3)}{E_\gamma}. \tag{19}$$

Using the values from Fig. 3 we get for $I = 0, 2, 4$ the moment of inertias

$$\mathcal{I}_0 = 3.99\,\hbar^2\ \mathrm{MeV}^{-1} \tag{20}$$

$$\mathcal{I}_2 = 6.33\,\hbar^2\ \mathrm{MeV}^{-1} \tag{21}$$

$$\mathcal{I}_4 = 6.93\,\hbar^2\ \mathrm{MeV}^{-1}, \tag{22}$$

which gives an average moment of inertia $\langle\mathcal{I}\rangle = 5.75\,\hbar^2\ \mathrm{MeV}^{-1}$.

Answer: The average moment of inertia for the rotational band of $^{48}_{24}\mathrm{Cr}$ is $\langle\mathcal{I}\rangle = 5.75\,\hbar^2\ \mathrm{MeV}^{-1}$.

(d)

Solution: From the lectures we know that the theoretical moment of inertia for a rigid body nuclei is

$$\mathcal{I}_{\text{rigid}} \approx \frac{2}{5} M r_0^2 A^{2/3}(1 + 0.31\beta_2), \tag{23}$$

where M is the mass, A is the atomic number, $r_0 \approx 1.2\,\text{fm}$ and $\beta_2 \approx 0.35$ is the quadrupole deformation parameter.

$$\mathcal{I}_{\text{rigid}} \approx 128.80\,\hbar^2\ \text{MeV}^{-1} \tag{24}$$

I think I do the conversion wrong to get the answer in the correct units. I don't have time to fix it :)

4 Fourth Exercise

(a)

Solution: The reaction that is seen is

$$n + {}^{235}_{92}\mathrm{U} \longrightarrow {}^{236}_{92}\mathrm{U}^* \longrightarrow \text{fission} + \nu n. \tag{25}$$

The activity is given as

$$A(t) = A_0 e^{-\lambda t}, \tag{26}$$

where λ is the decay probability of ${}^{236}_{92}\mathrm{U}^*$. However, there is only a $p = 84.0(2)\,\%$ probability that ${}^{92}_{235}\mathrm{U}$ absorbs a neutron and turns into ${}^{236}_{92}\mathrm{U}^*$ so this factor needs to be accounted for. Since the power output is constant at $2.2\,\mathrm{GW} = 1.37 \times 10^{22}\,\mathrm{MeV\,s^{-1}}$ and every fissile event is $190(5)\,\mathrm{MeV}$. However, the thermal efficiency of a nuclear power plant is $35(2)\,\%$ so we can calculate the base activity.

$$A_0 = \frac{\text{Power}}{\text{Energy per Fission event} \times \text{Efficiency}} \tag{27}$$

The total number of fission events during one year can then be calculated with

$$N^{\text{fission}}_{\text{year}} = A_0 \times T_{\text{year}} \tag{28}$$

and taking all used ${}^{235}_{92}\mathrm{U}$ into account we get

$$N^{\text{total}-235}_{\text{year}} = \frac{N^{\text{fission}}_{\text{year}}}{p} = 7.8(5) \times 10^{27}. \tag{29}$$

From the course literature we know that the mass per nuclide of ${}^{236}_{92}\mathrm{U}$ is $m({}^{236}_{92}\mathrm{U}) = 236.045\,562\,\mathrm{u} = 3.9 \times 10^{-25}\,\mathrm{kg}$. This gives the total mass ${}^{235}_{92}\mathrm{U}$ consumed each year is $M({}^{235}_{92}\mathrm{U/yr}) = N^{\text{total}-235}_{\text{year}} \times m({}^{236}_{92}\mathrm{U}) = 3040(190)\,\mathrm{kg}$. Looking up data on the internet one can find that for $1\,\mathrm{GW}$ of energy generation for a year about one tonne of ${}^{235}_{92}\mathrm{U}$ is needed and the calculated value used here is a bit too high. However, some energy is also released from the daughter nuclide's decay as well. The code used for the calculation can be seen in App. A.4

Answer: The total mass ${}^{235}_{92}$U consumed each year for a constant power generation of 2.2 GW is 3040(190) kg

(b)

Solution: The probability p_f for fission of ${}^{235}_{92}$U can be calculated as

$$p_\mathrm{f} = \frac{\sigma_f({}^{235}_{92}\mathrm{U})}{\sigma_a({}^{235}_{92}\mathrm{U}) + \sigma_a({}^{238}_{92}\mathrm{U})}\mathrm{NA}({}^{235}_{92}\mathrm{U}) \tag{30}$$

where σ_f is the cross-section for fission, σ_a is the cross-section for capture of neutrons and NA is the natural abundance. The probability p_c for capture for ${}^{238}_{92}$U can be calculated as

$$p_\mathrm{c} = \frac{\sigma_c({}^{238}_{92}\mathrm{U})}{\sigma_a({}^{235}_{92}\mathrm{U}) + \sigma_a({}^{238}_{92}\mathrm{U})}\mathrm{NA}({}^{238}_{92}\mathrm{U}) \tag{31}$$

The ratio of ${}^{238}_{92}$U capture and ${}^{235}_{92}$U fission is then

$$\xi = \frac{p_\mathrm{c}}{p_\mathrm{f}} = \frac{\sigma_c({}^{238}_{92}\mathrm{U})}{\sigma_f({}^{235}_{92}\mathrm{U})}\frac{\mathrm{NA}({}^{238}_{92}\mathrm{U})}{\mathrm{NA}({}^{235}_{92}\mathrm{U})} = R \times \frac{\mathrm{NA}({}^{238}_{92}\mathrm{U})}{\mathrm{NA}({}^{235}_{92}\mathrm{U})} \tag{32}$$

The natural abundance of ${}^{235}_{92}$U and ${}^{238}_{92}$U is $\mathrm{NA}({}^{235}_{92}\mathrm{U}) = 0.72\,\%$ and $\mathrm{NA}({}^{238}_{92}\mathrm{U}) = 99.28\,\%$ respectively. Thus

$$\xi = 0.552(14) \tag{33}$$

so the number of ${}^{238}_{92}$U nuclides which capture neutrons and turn in to ${}^{239}_{94}$Pu is then

$$N^\text{capture-238}_\text{year} = N^\text{total-235}_\text{year} \times \xi = 4.3(3) \times 10^{27} \tag{34}$$

and the mass per ${}^{239}_{94}$Pu nuclide is $m({}^{239}_{94}\mathrm{Pu}) = 239.052\,157\,\mathrm{u}$ which gives a total mass during a year to $M({}^{239}_{94}\mathrm{Pu}) = 1710(120)\,\mathrm{kg}$. The calculations can be seen in A.5.

Answer: The total mass ${}^{239}_{94}$Pu produced each year is 1710(120) kg

(c)

Solution: The activity is calculated as

$$A(t) = A_0 e^{-\lambda t} = \lambda N_0 e^{-\lambda t} \tag{35}$$

and λ can be calculated as $\lambda = \ln 2/T_{1/2}$ and the total number of ${}^{254}_{98}\mathrm{Cf}$ nuclides N_0 can be calculated as $N_0 =$ Total mass/Mass per nuclide, and the nuclide mass is $m({}^{254}_{98}\mathrm{Cf}) =$ 254.087 317 u. Since the amount of ${}^{254}_{98}\mathrm{Cf}$ will decrease over time, we can calculate for $t = 0$ and the power is $P = 11.3(3)\,\mathrm{mW}$, and the efficiency of the reactor give $P_{\mathrm{eff}} = 4.0(2)\,\mathrm{mW}$. The power is very small, but the mass is also incredibly little, so even if it is more active than the uranium the small amount doesn't produce any real energy. The calculations can be seen in App. A.6

Answer: The total power generated in the reactor from 1.0 µg of ${}^{254}_{98}\mathrm{Cf}$ is 4.0(2) mW.

(d)

Solution: Rise in temperature for a metal can be calculated as

$$\Delta T = \frac{E}{M({}^{254}_{98}\mathrm{Cf}) \times C}. \tag{36}$$

Since one minute is much less than 60 days, the effect can be though of as constant. The specific heat capacity for ${}^{254}_{98}\mathrm{Cf}$ is unknown but using the closest nuclide with a known value, Americium, we get $C = 110\,\mathrm{J\,kg^{-1}\,K^{-1}}$. This gives the temperature change in one minute as $\Delta T = 6.18(24)\,\mathrm{MK}$. This value seems unreasonably high. The reason is most likely that we have neglected the heat exchange with the environment The calculations can be seen in App. A.6.

Answer: The theoretical change of temperature in one minute for the ${}^{254}_{98}\mathrm{Cf}$ source without heat exchange is 6.18(24) MK

A Code

A.1 1a

```python
import numpy as np

def Zmin(A: int) -> float:
  """
  Calculation of Zmin from mass number

  parameters:
    A; int; The mass number.
  returns:
    Zmin; float; The minimum Z
  """
  return A/(1.98 + 0.015 * A**(2/3))

def find_ee_A(Z: int) -> int:
  """
  Calculation of the mass number of the most stable even-even nuclide of given atomic number

  parameters:
    Z; int; The atomic number for which the most stable nuclide is to be found
  returns:
    A; int; The mass number of the most stable even-even nuclide with atomic number Z
  """
  A_list = np.arange(Z, Z +200, 2)
  idx = np.argmin(np.abs((Zmin(A_list) - Z)))
  return A_list[idx]

print(f"The most stable nuclide is nickel-{find_ee_A(28)}")
```

A.2 1b

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc

rc('font',**{'family':'serif','serif':['Computer Modern'], 'size':'16'})
rc('text', usetex=True)


def B(A: int, Z: int) -> float:
  """
  Implementation of the Weizsacker mass formula from the lecture,
  for even-even nuclides

  parameters:
    A; int; The mass number
    Z; int; The atomic number

  returns:
    B; float; Binding energy [MeV]
  """
  N = A - Z
```

```
    volume_term = 15.9 * A
    surface_term = 18.4 * A**(2/3)
    coulomb_term = 0.71 * Z**2 * A**(-1/3)
    assymetry_term = 23.2 * (N - Z)**2 / A
    pairing_term = 11.5 * A**(-1/2)
    return volume_term - surface_term - coulomb_term - assymetry_term +
     pairing_term

def Sn(A: int, Z: int) -> float:
    """
    Finding the neutron seperation energy [MeV]

    parameters:
    A; int; The mass number
    Z; int; The atomic number

    returns:
      Sn; float; The neutron seperation energy [MeV]
    """
    return B(A, Z) - B(A - 1, Z)

def Sp(A: int, Z: int) -> float:
    """
    Finding the proton seperation energy [MeV]

    parameters:
    A; int; The mass number
    Z; int; The atomic number

    returns:
      Sp; float; The proton seperation energy [MeV]
    """
    return B(A, Z) - B(A - 1, Z - 1)

def find_driplines(Z: int, pr: bool = False) -> float:
    """
    Finding the neutron and proton driplines for an even-even nuclide with
      a given atomic number

    parameters:
      Z; int; the atomic number

    returns:
      Driplines; tuple; A tuple containging the mass numbers
                 on the form (neutron dripline, proton dripline)
    """

    A_list = np.arange(Z+2, Z + 200, 2)

    Sn_list = Sn(A_list, Z)
    idx_N = np.argmin(np.abs(Sn_list))
    N_dripline = A_list[idx_N]

    Sp_list = Sp(A_list, Z)
    idx_P = np.argmin(np.abs(Sp_list))
    P_dripline = A_list[idx_P]

    if pr:
```

```
    print(f"The neutron dripline is nickel-{N_dripline} and the proton
   dripline is nickel-{P_dripline}")

  return (N_dripline, P_dripline)


def plot():
  """
  Plotting the drip lines
  """
  Z_list = np.arange(4, 82, 2)
  A_dripline = np.vectorize(find_driplines)

  N_neutron = A_dripline(Z_list)[0] - Z_list
  N_proton = A_dripline(Z_list)[1] - Z_list

  fig, ax = plt.subplots(1,1)
  fig.set_figheight(4*1.5)
  fig.set_figwidth(4*(11125/7438)*1.5)

  ax.plot(N_neutron, Z_list, c='b', lw=3, label='Neutron dripline')
  ax.plot(N_proton, Z_list, c='r', lw=3, label='Proton dripline')

  img = plt.imread('./NuclideMap.jpg')
  ax.imshow(img, extent=[-0.5, 176.5, -0.5, 117.5])

  ax.set_xlim(0, 120)
  ax.set_ylim(0, 80)

  ax.set_ylabel(r'Atomic number, $Z$')
  ax.set_xlabel(r'Neutron number, $N$')

  ax.legend()

  plt.savefig('driplines.pdf', dpi=500)

plot()
```

A.3 2cd

```
import numpy as np
import scipy.constants as c
from scipy.stats import linregress as linreg
import matplotlib.pyplot as plt
import uncertainties as u
from uncertainties import unumpy as unp
from matplotlib import rc

rc('font',**{'family':'serif','serif':['Computer Modern'], 'size':'16'})
rc('text', usetex=True)


nuclide_data = np.array([
  ["B",  11, 5,  u.ufloat(9305  , 1)],
  ["C",  11, 6,  u.ufloat(11434 , 1)],
  ["N",  15, 7,  u.ufloat(109   , 1)],
  ["O",  15, 8,  u.ufloat(3065  , 1)],
  ["F",  19, 9,  u.ufloat(-1597 , 1)],
```

```
  ["Ne", 19, 10, u.ufloat(1880  , 1)],
  ["Na", 23, 11, u.ufloat(-10230, 1)],
  ["Mg", 23, 12, u.ufloat(-5875 , 1)],
  ["Si", 29, 14, u.ufloat(-23505, 1)],
  ["P",  29, 15, u.ufloat(-18199, 1)],
  ["Cl", 35, 17, u.ufloat(-31147, 1)],
  ["Ar", 35, 18, u.ufloat(-24743, 1)],
  ["Ca", 41, 20, u.ufloat(-37722, 1)],
  ["Sc", 41, 21, u.ufloat(-30749, 1)],
  ["Ti", 45, 22, u.ufloat(-41876, 1)],
  ["V",  45, 23, u.ufloat(-34218, 1)]
])

def mass_diff(data: np.ndarray) -> np.ndarray:
  """
  Calculates the mass difference

  parameters:
    data; numpy array; the input data
  returns:
    mass_diff_total; numpy array, the mass difference and the error
    mass_diff; numpy array; the mass difference
    error; numpy array; the error after the calculations
  """
  idx1 = np.arange(0, len(data), 2)
  idx2 = np.arange(1, len(data), 2)
  data1 = data[idx1]
  data2 = data[idx2]
  mass_diff_total = (data2[:,3] - data1[:,3]) * 0.0009315 # MeV / u
  mass_diff = unp.nominal_values(mass_diff_total)
  error = unp.std_devs(mass_diff_total)
  return mass_diff_total, mass_diff, error

def plot(X, Y, error):
  """
  Plotting
  """
  fig, ax = plt.subplots(1,1)
  fig.set_figwidth(6)
  fig.set_figheight(4)

  ax.set_xlabel(r'$\frac{3 e^2 A^{2/3}}{20 \pi \epsilon_0}$ [MeV $\
    times$ fm]')
  ax.set_ylabel(r'$\Delta E$ [MeV]')

  scale = 1.602e-13*1e-15 # Scaling the X data/axis to MeV * fm
  reg = linreg(X, Y)
  x_reg = np.linspace(X[0], X[-1], 200)
  y_reg = (x_reg*reg[0] + reg[1])

  print(reg)
  slope = 1/(reg[0]*scale)
  err = 2*(reg[4]/reg[0]) * 1/(reg[0]*scale) # calculating 2 standard
    deviations so approx 95% confidence
  print(f'r0 = {round(slope,2)} fm +/- {round(err,2)} fm')

  ax.plot(x_reg/scale, y_reg, c='r', label='Linear regression')
  ax.scatter(X/scale, Y, c='fuchsia', s=16, zorder=3, label='Data points
```

```
')
    ax.errorbar(X/scale, Y, yerr=error, ls='None', zorder=1, label='Error')

    ax.legend()
    plt.tight_layout()
    plt.savefig('r0.pdf')


mdt, Dm, error = mass_diff(nuclide_data)
A = np.array([11, 15, 19, 23, 29, 35, 41, 45])
x_list = 3*(1.6e-19)**2 * A**(2/3) / (20*np.pi*8.85e-12)

plot(x_list, Dm, error)
```

A.4 4a

```
import uncertainties as u
from uncertainties import unumpy as unp


def kgYear(P: float):
    """
    Calculates the kg of U-235 used per year to generate a constant power of P

    parameters:
        P; float; The total Power
    return:
        prints the total mass used per year.
    """
    eff = u.ufloat(0.35, 0.02) #efficiency of nuclear power plants
    power = P/1.602e-13 # converting from watt to MeV/s
    E_fission = u.ufloat(190, 5) * eff # energy per fission events in MeV
    year = 31556952 #seconds in a year
    p = u.ufloat(0.84, 0.002) # probability that U-236 decays with fission
    activity = power/E_fission
    mass_per_N = (236 + 45562e-6)*1.66e-27 #mass per nuclide in kg
    N = activity * year/p # Total number of decays in a year
    total_mass = N*mass_per_N
    return print(f"The total number of nuclides needed is {N} and the mass is {total_mass}")

kgYear(2.2e9)
```

A.5 4b

```
import uncertainties as u

def xi(R: float):
    """
    Calculates xi as seen in the assignment

    parameters:
        R; float; The  relative neutron capture probability

    returns;
        xi; float; the ratio
```

```
    """
    U235 = 0.0072
    U238 = 0.9928
    return R * U238/U235

def Ncapture(xi: float, Nfission: float):
    """
    Calculates the number of U238 turning into Pu239
    parameters:
      xi; float; The ratio against fission
      Nfission; float; the number of U235

    returns;
      Ncapture; float; the number of Pu239
    """
    return Nfission * xi

N = Ncapture(xi(u.ufloat(4e-3, 0.1e-3)), u.ufloat(7.8e27, 0.5e27))

print(f"The total mass of Pu-239 {N * 239.052157 * 1.66e-27} kg" )
```

A.6 4cd

```
import uncertainties as u
import numpy as np

def power(mass: float):
    """
    Calculates the power from a given mass Cf254

    parameters:
      mass; float; the mass of Cf254

    return:
      power; float; The total power released from Fission
    """
    nuclide_mass = 254.087317 * 1.66e-27 #mass in kg/nuclide
    N_0 = mass/nuclide_mass #total number
    T12 = u.ufloat(60.5, 0.2) * 86400 #half life in seconds
    l = np.log(2)/(T12)
    A_0 = N_0 * l
    E_per_event = u.ufloat(225, 5) * 1.602e-13
    print(N_0)
    return A_0 * E_per_event  # power in W

P = power(1e-9)
Peff = P * u.ufloat(0.35, 0.02)

print(f" The power is {Peff*1e3} mW")

energy_minute = P * 60
print(f"The temperature change in one minute is {energy_minute/(1e-9 *
    110)}")
```