Stucture Exercise 1

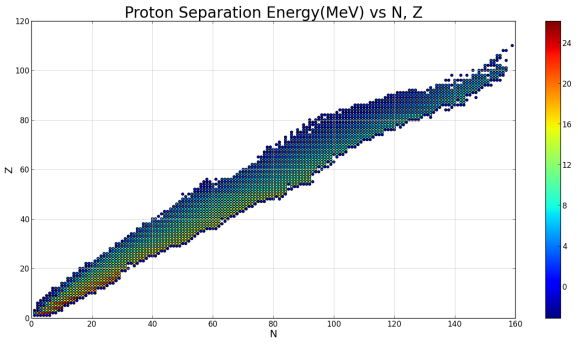
Charles Loelius

January 21, 2014

1 Separation Energies

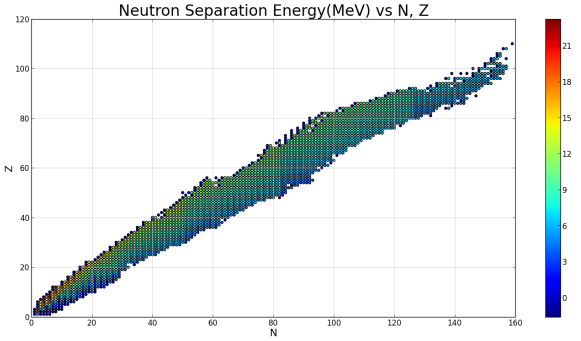
I demonstrate below two plots generated from a script which can be found on github, and which will be printed below.

First, for the proton separation energy:



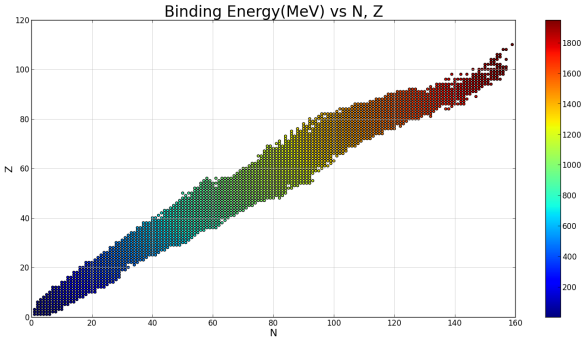
We note that, as might be expected, the largest separation energies are found at high N, low Z parts of the range, while the high Z, low N regions have very small such energy. This makes sense, as in the former case we would expect to be near the neutron dripline, where the protons are vital for stability, wheras in the latter case we'd expect to be near the proton dripline and so expect it to be very easy to remove a proton(and so have low proton separation energy). The only exception is in the very borders near the high end of the neutron to proton ratio, where the separation energies become very small, which may be an effect of the extreme instability in that range.

Next, for the neutron separation energies.



We see as we would expect from the previous discussion something very much like an inverse of the proton separation energies, with the largest values for cases of low N and high Z, and the smallest values in cases of high N and low Z, again representing the relative instability of high N nuclei, and their relative propensity towards shedding neutrons, versus high Z nuclei which require those neutrons to be (more) stable.

Finally we plot the binding energies here:

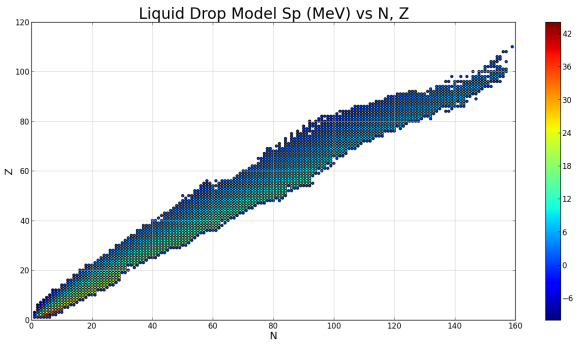


We see as we would trivially expect an increase in binding energy totals with increasing A.

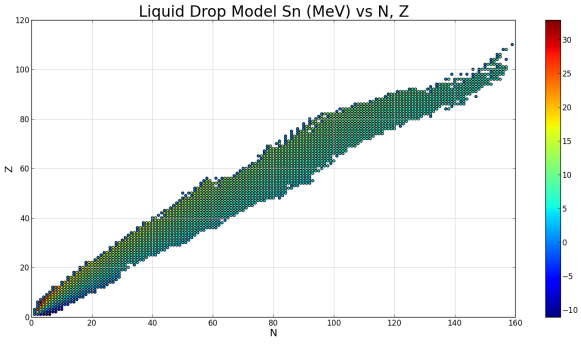
2 Liquid Drop Model Results

I compare the results from the previous section to the liquid drop model using similar plots. I chose not to plot these on the same graphs because of the dificulty of comparsion. In addition, I have shown a "relative" comparison between the liquid drop results and the empirical results. I begin with the liquid drop model's results for the proton and

neutron separation energies. For the proton separation energies we have:



And for the neutron separation energies we also have:

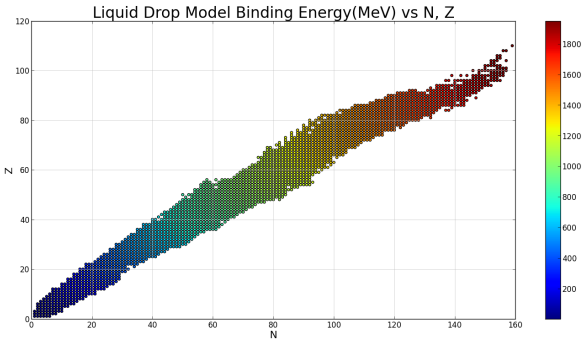


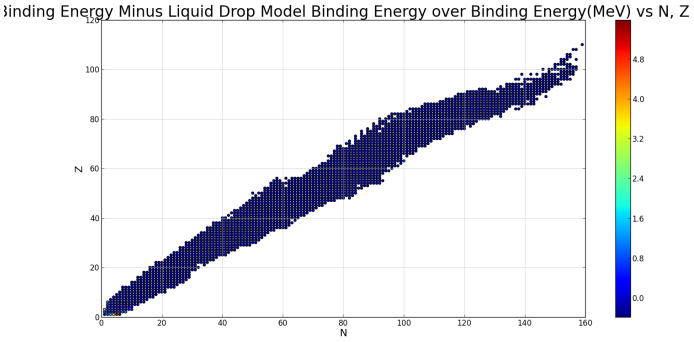
We see that this has the same features when it comes to the high Z/low N and low Z/high N limits as the empirical values, suggesting the equation is at least roughly correct. We will see in the binding energies that this is in fact a much better approximation than expected.

First we look at the actual binding energies, which we see have the same structure as in the empirical results.

We then follow this with a look at the relative difference between the binding energies.

From this we see that the relative differences are throughout in the range of less than 50% and are in general much closer, with the only exceptions being the extremely low Z and N cases, where we have differences ranging up to 400%.

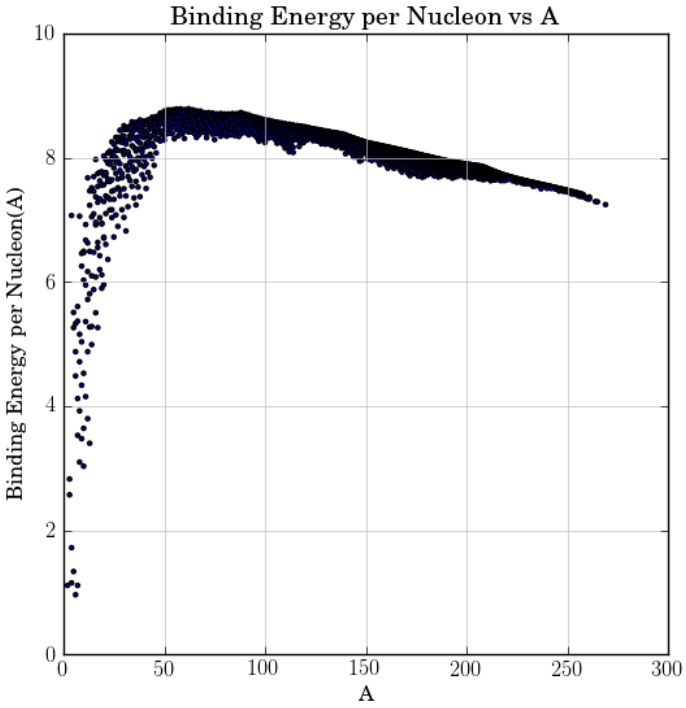




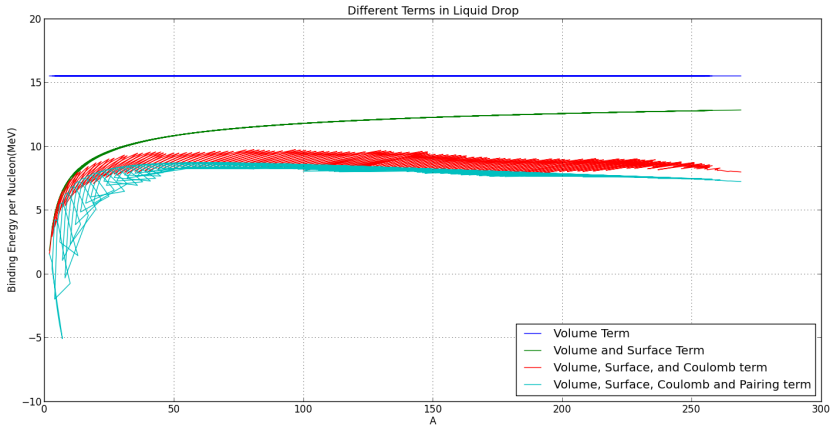
3 Comparison of Terms in Data and Emperical Mass Equation

We plot below the experimentally observed binding energy per nucleon vs A.

Next we consider each of the terms in the Empirical Mass formula. These are shown below, with binding energy per nucleon shown. This will allow us to see the effects of each term.

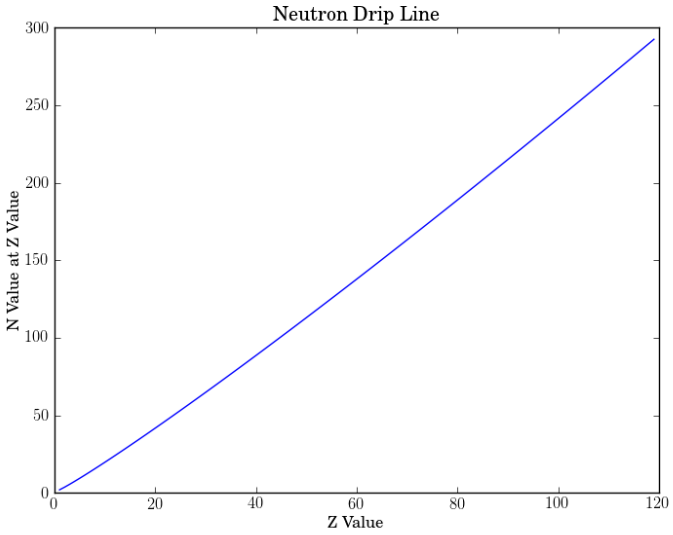


We see as expected that the volume term provides a constant stability, which is counteracted by the coulomb and surface terms, and that indeed even the surface term alone seems to be enough to begin a strong structure in the system, but that we do not produce a curve with a peak until either the coulomb or coulomb and pairing terms enter into the equation(it is hard to judge the coulomb term because of the fact that there is a wide variety in the energy spread and so how and if it peaks is difficult to say.) This suggest the vital importance of all of these terms, at least at a prima facie level, in order to explain nuclei, as the curve of binding energy is vital for explaining astrophysical abundances, and without the maxima would utterly change the composition of the cosmos.



4 Neutron Drip Line

I use a separate program to calculate the neutron drip line by finding the zero's of the separation function using sympy. The point where the neutron separation energy becomes negative is the point where we anticipate finding the neutron dripline. Doing so we find the following plot:



Investigating we see that in the range of 8 to 9 protons we would expect the neutron drip line at approximately N=16 and 19 respectively. Indeed, the empirical value of the neutron drip line appears to be 16 for Oxygen(as can be found at `https://groups.nscl.msu.edu/theory/content/oxygen-isotopes-beyond-neutron-dripline`), and

we also find that for fluorine the dripline is at N=22(`http://cerncourier.com/cws/article/cern/31868`). We thus see that the Emperical Mass Formula is very close for the oxygen isotope but seems to becomes slightly off even as close by as flourine. This suggests the possibility that for any higher z isotopic chains the emperical mass formula may not work for finding the neutron drip line.

5 Script for Generating Binding Energies

```
import numpy
import matplotlib.pyplot as plt
from matplotlib import cm
def MakeZNPlot(Z,N,X,TITLE):
    fig = plt.figure(figsize=(6,6))
    ax = fig.add_subplot(111)
    ax.set_title(TITLE,fontsize=24)
    ax.set_xlabel("N",fontsize=16)
    ax.set_ylabel("Z",fontsize=16)
    ax.set_ylim([0,120])
    ax.set_xlim([0,160])

    ax.grid(True,linestyle='-',color='0.75')


    # scatter with colormap mapping to z value
    a=ax.scatter(N,Z,s=15,c=X, marker = 'o', cmap = cm.jet );
    plt.colorbar(a)
    plt.show()
    plt.savefig(TITLE+".png")
def MakeAPlot(A,X,TITLE,YLAB):
    fig = plt.figure(figsize=(6,6))
    ax = fig.add_subplot(111)
    ax.set_title(TITLE,fontsize=14)
    ax.set_xlabel("A",fontsize=12)
    ax.set_ylabel(YLAB,fontsize=12)
    ax.grid(True,linestyle='-',color='0.75')
    ax.set_xlim([0,300])


    # scatter with colormap mapping to z value
    ax.scatter(A,X,s=5, marker = 'o' );
    plt.savefig(TITLE+".png")
    plt.show()
def MakeAPlots(A,X,TITLE,YLAB,names):
    fig = plt.figure(figsize=(6,6))
    ax = fig.add_subplot(111)
    ax.set_xlim([0,300])
    ax.set_title(TITLE,fontsize=14)
    ax.set_xlabel("A",fontsize=12)
    ax.set_ylabel(YLAB,fontsize=12)
    ax.grid(True)
    for i,x in enumerate(X):
        plt.plot(A,x,label=names[i])
    plt.legend(loc=4)
    plt.savefig(TITLE+".png")
```

```
    plt.show()
def WBBE(A,Z,a1,a2,a3,a4):
        N=A-Z
        return a1*A-a2*A**(2.0/3.0)-a3*Z**2/(A**(1/3))-a4*(N-Z)**2/A
class Nucleus:
        Z=0
        N=0
        A=0
        Sn=0
        Sp=0
        SnWB=0
        SpWB=0
        BE=0
        BEEMP=0#BE is the binding energy from Weisaker-Bette
        alpha1=15.49#Energy in MeV
        alpha2=17.23
        alpha3=.697
        alpha4= 22.6
#         def __init__(self,Z,A,BE):
#             self.Z=Z
#             self.A=A
#             self.N=A-Z
#
    self.BEEMP=WBBE(self.A,self.Z,self.alpha1,self.alpha2,self.alpha3,self.alpha4)
#             self.BE=BE
        def
            __init__(self,Z,A,BE,alpha1=15.49,alpha2=17.23,alpha3=.697,alpha4=22.6):
            self.Z=Z
            self.A=A
            self.N=A-Z
            self.alpha1=alpha1
            self.alpha2=alpha2
            self.alpha3=alpha3
            self.alpha4=alpha4
            self.BEEMP=WBBE(self.A,self.Z,self.alpha1,self.alpha2,self.alpha3,self.alpha4)
            self.BE=BE

        def  GetSeparationEnergies(self,l):
            if(A==2):
                self.Sp=self.BE
                self.Sn=self.BE
            else:
                for nuc in l:
                    if(nuc.N==self.N):
                        if(nuc.Z==int(self.Z-1)):
                            self.Sp=self.BE-nuc.BE
                            self.SpWB=self.BEEMP-nuc.BEEMP
                    if(nuc.Z==self.Z):
                            if(nuc.N==int(self.N)-1):
```

```
                                    self.Sn=self.BE-nuc.BE
                                    self.SnWB=self.BEEMP-nuc.BEEMP
fileform=numpy.loadtxt("C:\\Users\\Charles\\Downloads\\mass\\mass\\aud11.dat",skiprows=2)
nuclei=[]
nucleialpha1=[]
nucleialpha12=[]
nucleialpha123=[]
nucleialpha1234=[]

for line in fileform:#Read in files here
        Z=line[0]
        A=line[1]
        BE=line[2]
        nuclei.append(Nucleus(Z,A,BE))
        nucleialpha1.append(Nucleus(Z,A,BE,15.49,0,0,0))
        nucleialpha12.append(Nucleus(Z,A,BE,15.49,17.23,0,0))
        nucleialpha123.append(Nucleus(Z,A,BE,15.49,17.23,.697,0))
        nucleialpha1234.append(Nucleus(Z,A,BE,15.49,17.23,.697,22.6))

for n in nuclei:#perform first calculations
    n.GetSeparationEnergies(nuclei)
Ns=[]
Zs=[]
BEs=[]
WBBEs=[]
Sns=[]
Sps=[]
WBSns=[]
WBSps=[]
As=[]
BEPN=[]
BEREL=[]
WBBEsalpha1=[]
WBBEsalpha12=[]
WBBEsalpha123=[]
WBBEsalpha1234=[]

for n in nuclei:#Prepare for plotting
    Ns.append(n.N)
    Zs.append(n.Z)
    BEs.append(n.BE)
    WBBEs.append(n.BEEMP)
    As.append(int(n.N+n.Z))
    Sns.append(n.Sn)
    Sps.append(n.Sp)
    BEPN.append(n.BE/n.A)
    BEREL.append((n.BE-n.BEEMP)/n.BE)
    WBSns.append(n.SnWB)
    WBSps.append(n.SpWB)
```

```
for i in range(len(nuclei)):
    WBBEsalpha1.append(nucleialpha1[i].BEEMP/nucleialpha1[i].A)
    WBBEsalpha12.append(nucleialpha12[i].BEEMP/nucleialpha12[i].A)
    WBBEsalpha123.append(nucleialpha123[i].BEEMP/nucleialpha123[i].A)
    WBBEsalpha1234.append(nucleialpha1234[i].BEEMP/nucleialpha1234[i].A)

MakeZNPlot(Zs,Ns,Sps,"Proton Separation Energy(MeV) vs N, Z")
#MakeZNPlot(Zs,Ns,As)
MakeZNPlot(Zs,Ns,Sns,"Neutron Separation Energy(MeV) vs N, Z")
MakeZNPlot(Zs,Ns,BEs,"Binding Energy(MeV) vs N, Z")
MakeZNPlot(Zs,Ns,BEs,"Liquid Drop Model Binding Energy(MeV) vs N, Z")
MakeZNPlot(Zs,Ns,WBSns,"Liquid Drop Model Sn (MeV) vs N, Z")
MakeZNPlot(Zs,Ns,WBSps,"Liquid Drop Model Sp (MeV) vs N, Z")
MakeZNPlot(Zs,Ns,BEREL,"Binding Energy Minus Liquid Drop Model Binding Energy
    over Binding Energy(MeV) vs N, Z")
MakeAPlot(As,BEs,"Binding Energy(MeV) vs A","Binding Energy")
WBs=[]
#WBs.append(WBBEs)
WBs.append(WBBEsalpha1)
WBs.append(WBBEsalpha12)
WBs.append(WBBEsalpha123)
WBs.append(WBBEsalpha1234)
names=["Volume Term", "Volume and Surface Term", "Volume, Surface, and Coulomb
    term", "Volume, Surface, Coulomb and Pairing term"]
MakeAPlots(As,WBs,"Different Terms in Liquid Drop","Binding Energy per
    Nucleon(MeV)",names)
#print(nuclei[20].Sp)
#print(nuclei[20].Sn)
```

6 Script for Generating Neutron Drip Line

```
import matplotlib.pyplot as plt
from sympy import *
def WBE(Z,N):
    "Returns a sympy version of Emperical Mass Formula"
   # Z=symbols('Z')
   # N=symbols('N')
    A=N+Z
    a1=15.49
    a2=17.23
    a3=.697
    a4=22.6
    equation=(a1*A-a2*A**(2.0/3.0)-a4*(N-Z)**2/A-a3*Z**2/A**(1/3.0))

    return equation

#def GetNeutronDrip(A,Z):
    #N=A-Z
Z=Symbol('Z')
N=Symbol('N')
Zs=list(range(1,120))
NeutronDrips=[]
for z in Zs:
    try:
        NeutronDrips.append(mpmath.findroot(lambda N: WBE(z,N),z*5))
    except:
        NeutronDrips.append(mpmath.findroot(lambda N: WBE(z,N),z*8))
    print(z)
NeutronMax=[]


Nn=list(range(100))
plt.plot(Zs,NeutronDrips)
plt.title("Neutron Drip Line")
plt.xlabel("Z Value")
plt.ylabel("N Value at Z Value")
WBs=[]
for i in Nn:
    WBs.append(WBE(8,i))
#plt.plot(Nn,WBs)
plt.show()
#print(WBE(Z,N))
#print(solve(WBE(Z,N),Z))
#print(mpmath.findroot(lambda N: WBE(8,N),8))
#mpmath.plot(lambda N: WBE(8,N),[0,20])
#mpmath.plot(lambda x: mpmath.exp(x)*mpmath.li(x), [1, 4])
```