Nuclear Shell Model

Are nuclei like atoms? Do the nucleons 'fill' single-particle orbitals inside an effective nuclear potential? Are there 'noble gas' nuclei, with particularly low energies? Are there shells like seen in the periodic table?

We shall discover that the binding energy for nucleons has large contributions that depend on other interesting quantum effects. But, if we subtract these other effects, we can see clear evidence of a shell structure. The protons and neutrons, distinguishable particles, each fill up their own shells. Low energies arise at 'magic' nucleon numbers, when the protons or neutrons exactly fill up a shell.

We first load experimental mass data for nuclei. If you are on-line, you can pull this directly from the Web; otherwise you can load a file.

In [ ]:
```
# massTable = open("mass.mas114.txt")  # If offline; may have to change direct
ory to find file

import urllib2                          # Access via Web
massTable = urllib2.urlopen('http://amdc.in2p3.fr/masstables/Ame2011int/mass.m
as114')

wholeFile = massTable.readlines()
massTable.close()
```

We now unpack the data. If you examine the file, they tell us the first 39 lines are header.

In [ ]:
```
data = wholeFile[39:]
```

We now pull out the data we need: the number of neutrons and protons for each isotope, and the 'mass excess' in MeV. Many of the lines are 'theoretical guesses' for masses; these have # signs instead of decimal points. We remove them. We also keep the element name (for convenience).

This is somewhat tedious, and I'm probably not doing it elegantly. The data is in Fortran format, with certain ranges of characters for each column. In the end, we have four lists: N, Z, mass excess, and the name of the nth element on the list will be neutrons[n], protons[n], massExcess[n], and elementName[n].

In [ ]:
```
neutrons = []
protons = []
elementName = []
massExcess = []
for line in data:
    if '#' not in line[28:41]:
        N = int(line[4:9])
        Z = int(line[9:14])
        name = line[20:23]
        dM = float(line[28:41])/1000.   # keV to MeV
        neutrons += [N]
        protons += [Z]
        elementName += [name]
        massExcess += [dM]
```

We will be comparing with the 'semi-empirical mass formula'. To do this, we must convert from 'mass excess' to 'binding energy'. The binding energy is the mass of N neutrons and Z protons minus the nuclear mass. The 'mass excess' is the atomic mass minus (N+Z) atomic mass units, where an atomic mass unit is 1/12 of a carbon 12 atom. So to get the nuclear mass, we add one amu per proton and subtract one electron mass per proton from the atomic mass excess. We then subtract this from the proton and neutron masses to get the binding energy.

Let's set up an array of binding energies; we'll leave them zero where there is missing data. We'll set up a matrix of names too, for convenience.

In [ ]:
```
# all in MeV
mP =
mE =
mN =
amu =

bindingEnergy = zeros((max(protons)+1, max(neutrons)+1))
names = [[None for i in range(max(neutrons)+1)] for j in range(max(neutrons)+1
)]
for dat in range(len(massExcess)):
        Z = protons[dat]
        N = neutrons[dat]
        A =
        nuclearMass =
        bindingEnergy[Z][N] =
        names[Z][N] = str(A) + elementName[dat]
```

Here's the 'semi-empirical mass formula', based on a 'liquid drop' model plus corrections for Coulomb interactions, Pauli exclusion, and spin.

- The volume term is the energy gained by A=(Z+n) nucleons gains as it 'condenses' into the nuclear fluid: aV * A
- The surface term is like the surface tension of the droplet: -aS * A^(2/3)
- The Coulomb term is the electrostatic energy needed to push Z protons into a sphere of radius proportional to A^(1/3); it is why heavy nuclei have fewer protons than neutrons: -aC Z^2 / A^(1/3)
- When there are more neutrons than protons, the Fermi energy of the neutrons is bigger than the protons. This gives a Pauli exclusion term to the nuclear energy: -aA * (N-Z)^2 / A.
- Finally, two protons or neutrons can get added to each orbital. Hence nuclei where both Z and N are even are more stable than average, and when both Z and N are odd they are less stable. This is accounted by a factor δ.

In [ ]:
```
def SemiEmpiricalMassFormula(Z,N):
    # Gives mass of nucleus in eV, from Rohlf (Wikipedia Semi-empirical_mass_f
ormula)
    aV = 15.75
    aS = 17.8
    aC = 0.711
    aA = 23.7
    aP = 11.18
    #
    A =
    if A == 0: return 0.
    volumeTerm =
    surfaceTerm =
    coulombTerm =
    PauliTerm =
    EB = volumeTerm + surfaceTerm + coulombTerm + PauliTerm
    delta =
    if Z%2 == 0 and N%2 == 0:
        EB += delta
    if Z%2 == 1 and N%2 == 1:
        EB -= delta
    return EB

(SemiEmpiricalMassFormula(26,30), bindingEnergy[26][30])
```

We test the semi-empirical mass formula with 56Fe. It should be within a percent of the right answer. Iron 56 has Z=26 and N=30; it is the nucleus with the largest binding energy. (A 'dead' white dwarf star where fusion has finished, but which is too small to collapse into a neutron star, will be made of 56Fe.) To debug your semi-empirical mass formula, I got the following values for the individual terms for 56Fe:

- volumeTerm = 882.0 MeV
- surfaceTerm = -260.54 MeV
- coulombTerm = -125.628 MeV
- PauliTerm = -6.77 MeV
- delta = 1.49 MeV

You can als check this formula against one of the on-line versions, such as http://hyperphysics.phy-astr.gsu.edu/hbase/nuclear/liqdrop.html.

We use the semi-empirical mass formula to build a matrix of results

In [ ]:
```
SEMFMatrix = array([[SemiEmpiricalMassFormula(Z,N) for N in range(max(neutrons
)+1)] for Z in range(max(protons)+1)])
```

We can use 'imshow' to plot our results:

In [ ]:
```
imshow(SEMFMatrix)
```

```
In [ ]: imshow(bindingEnergy)
```

We provide a plotting routine for giving the difference between the true binding energy and that given by the semi-empirical mass formula. Negative values (blue) indicate extra binding (i.e., regions of magic stability). Notice we use "masked arrays" to avoid plotting values where there is no experimental data.

```
In [ ]: def PlotDifference(minZ = 0, minN = 0, maxZ = max(protons)+1, \
                           maxN = max(neutrons)+1, vminMin = -100):
            # Masked array ma.array only plots where bindingEnergy != 0
            toShow = ma.array(SEMFMatrix-bindingEnergy, \
                                  mask=(bindingEnergy==0))[minZ:maxZ,minN:maxN]
            vmin = max(ma.min(toShow.flatten()), vminMin)
            vmax = -vmin    # To make perfect agreement = white, center vertical scale
            figure()
            # imshow transposes axes and flips y-axis from traditional direction
            imshow(toShow, interpolation='nearest', vmin=vmin, vmax=vmax, \
                   extent=(minN-0.5,maxN-0.5,maxZ-0.5,minZ-0.5), cmap=cm.coolwarm)
            colorbar()
            xlabel('neutrons')
            ylabel('protons')
```

Plotting the whole range is washed out, so use vminMin = -5 to make it more vivid.

Zoom in on 0 < Z, N < 22 to see if there is any evidence of magic for small nuclei.

```
In [ ]: PlotDifference(vminMin=-5)

        PlotDifference(0,0,22,22)
```

Do you observe especially stable nuclei along 'magic numbers' of protons or neutrons? How do these compare with those quoted by Wikipedia, [2, 8, 20, 28, 50, 82, 126]?