Nuclear Models

Morten Hjorth-Jensen

National Superconducting Cyclotron Laboratory and Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA

NS3 Nuclear Physics Summer School 2019

Stability of matter

To understand why matter is stable, and thereby shed light on the limits of nuclear stability, is one of the overarching aims and intellectual challenges of basic research in nuclear physics. To relate the stability of matter to the underlying fundamental forces and particles of nature as manifested in nuclear matter, is central to present and planned rare isotope facilities.

Important properties of nuclear systems which can reveal information about these topics are for example masses, and thereby binding energies, and density distributions of nuclei. These are quantities which convey important information on the shell structure of nuclei, with their pertinent magic numbers and shell closures or the eventual disappearence of the latter away from the valley of stability.

Motivation and the many scales of the nuclear many-body problem

To understand nuclei and nuclear matter theoretically means

- Understand how the laws of motion and the fundamental forces at play govern the **nuclear world**
- Quantum Chromodynamics with quarks and gluons is the model for the strong force
- The degrees of freedom from quarks and gluons are too complicated for an *effective* theoretical study of nuclei and nuclear matter
- Effective degrees of freedom (*dof*) in terms of protons, neutrons and selected mesons like pions. These *dof* lead to an effective field theory for the nuclear forces

- Solve the nuclear many-body problem using various mathematical algorithms, large scale diagonalization, renormalization techniques, quantum computing, machine learning, lattice quantum chromodynamics etc etc. The math spans from eigenvalue problems to stochastic modeling.

Where to find the material

The Jupyter Notebook is at the link https://github.com/mhjensenseminars/EducationalSeminars/tree/master/doc/pub/nuclearmodels/ipynb

The material in various formats (jupyter notebooks, various html and pdf formats) is at http://mhjensenseminars.github.io/EducationalSeminars/doc/web/overview.html

The main Github address for all material is at https://github.com/mhjensenseminars/EducationalSeminars

Feel free to use this material as you wish.

The nuclear many-body problem, where do we start?

During this summer school you will meet several of the above theoretical approaches to understand nuclei from a bottom-up type of approach (lectures of Brown, Lee and Nunes).

The aim here is provide you with the starting point for modeling nuclei and nuclear matter.

Where we do start from? Our main ansatz for a many-body wave function starts with a product of single-particle wave function.

- What is the rationale for doing so?
- And what kind of data do we have which motivates this? Hint: Think of the periodic system of atoms

Drip lines

Neutron (or proton)-rich nuclei are particularly interesting. As a particular chain of isotopes (isotones) becomes more and more neutron (proton) rich, one reaches finally the limit of stability, the so-called dripline, where one additional neutron (proton) makes the next isotope (isotone) unstable with respect to the previous ones.

The appearence or not of magic numbers and shell structures, the formation of for example neutron skins and halos can thence be probed via investigations of quantities like the binding energy or the charge radii and neutron rms radii of neutron-rich nuclei. These quantities have in turn important consequences for theoretical models of nuclear structure and their application in astrophysics.

FRIB limits for the chain of calcium isotopes

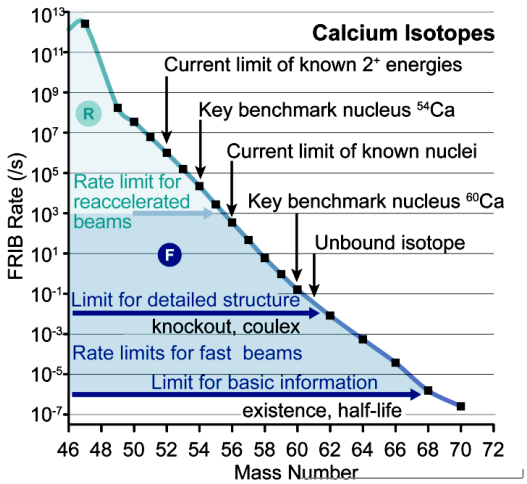

Figure 1: Expected experimental information on the calcium isotopes that can be obtained at FRIB. The limits for detailed spectroscopic information are around $A \sim 60$.

More on Neutron-rich nuclei

The neutron radius of ^{208}Pb, recently extracted from the PREX experiment at Jefferson Laboratory can be used to constrain the equation of state of neutron matter.

A related quantity to the neutron rms radius $r_n^{\rm rms} = \langle r^2 \rangle_n^{1/2}$ is the neutron skin $r_{\rm skin} = r_n^{\rm rms} - r_p^{\rm rms}$, where $r_p^{\rm rms}$ is the corresponding proton rms radius. There are several properties which relate the thickness of the neutron skin to quantities in nuclei and nuclear matter, such as the symmetry energy at the saturation point for nuclear matter, the slope of the equation of state for neutron matter or the low-energy electric dipole strength due to the pigmy dipole resonance.

Mean-field picture

With a mean- or average-field picture we mean that a given nucleon (either a proton or a neutron) moves in an average potential field which is set up by all other nucleons in the system. Consider for example a nucleus like ^{17}O with nine neutrons and eight protons.

Many properties of this nucleus can be interpreted in terms of a picture where we can view it as one neutron on top of ^{16}O. We infer from data and our theoretical interpretations that this additional neutron behaves almost as an individual neutron which *sees* an average interaction set up by the remaining 16 nucleons in ^{16}O. A nucleus like ^{16}O is an example of what we will denote as a good closed-shell nucleus. More later.

Mean-field picture, which potential do we opt for?

A simple potential model which enjoys quite some popularity in nuclear physics, is the **three-dimensional harmonic oscillator**. This potential model captures some of the physics of deeply bound single-particle states but fails in reproducing the less bound single-particle states.

Mean-field picture, more realistic model

A parameterized, and more realistic, potential model which is widely used in nuclear physics, is the so-called **Woods-Saxon** potential. Both the harmonic oscillator and the Woods-Saxon potential models define computational problems that can easily be solved (see below), resulting (with the appropriate parameters) in a rather good reproduction of experiment for nuclei which can be approximated as one nucleon on top (or one nucleon removed) of a so-called closed-shell system.

Too simple?

To be able to interpret a nucleus in such a way requires at least that we are capable of parametrizing the abovementioned interactions in order to reproduce say the excitation spectrum of a nucleus like ^{17}O.

With such a parametrized interaction we are able to solve Schroedinger's equation for the motion of one nucleon in a given field. A nucleus is however a true and complicated many-nucleon system, with extremely many degrees of freedom and complicated correlations, rendering the ideal solution of the many-nucleon Schroedinger equation an impossible enterprise. It is much easier to solve a single-particle problem with say a Woods-Saxon potential.

Motivation, better mean-fields

An improvement to these simpler single-nucleon potentials is given by the Hartree-Fock method, where the variational principle is used to define a mean-field which the nucleons move in. There are many different classes of mean-field methods.

An important difference between these methods and the simpler parametrized mean-field potentials like the harmonic oscillator and the Woods-Saxon potentials, is that the resulting equations contain information about the nuclear forces

present in our models for solving Schroedinger's equation. Hartree-Fock and other mean-field methods like density functional theory can be considered as essential *kitchen items* in our attempts in solving the nuclear many-body problem.

Aims here

The aim here is to present some of the experimental data we use to motivate our ansatz for a many-body wave function. In particular, we will focus on separation energies and shell-gap energies and use these to build a picture of nuclei in terms of (from a philosophical stand we would call this a reductionist approach) a single-particle picture.

The harmonic oscillator will serve as an excellent starting point in building nuclei from the bottom and up. Here we will neglect a theoretical description in terms of the underlying nuclear forces (see Brown's and Hergert's lectures).

We want to develop our physics intuition of nuclear systems using a theoretical approach where we describe data in terms of the motion of individual nucleons and their mutual interactions.

Do we understand the physics of dripline systems?

Our first approach in analyzing data theoretically, is to see if we can use experimental information to

- Extract information about a *so-called* single-particle behavior
- And interpret such a behavior in terms of the underlying forces and microscopic physics

The next step is to see if we could use these interpretations to say something about shell closures and magic numbers. Since we focus on single-particle properties, a quantity we can extract from experiment is the separation energy for protons and neutrons. Before we proceed, we need to define quantities like masses and binding energies. Two excellent reviews on recent trends in the determination of nuclear masses can be found in the articles of Lunney and co-workers and Blaum and co-workers

Masses and Binding energies

A basic quantity which can be measured for the ground states of nuclei is the atomic mass $M(N, Z)$ of the neutral atom with atomic mass number A and charge Z. The number of neutrons is N.

Atomic masses are usually tabulated in terms of the mass excess defined by

$$\Delta M(N,Z) = M(N,Z) - uA,$$

where u is the Atomic Mass Unit

$$u = M(^{12}\mathrm{C})/12 = 931.49386 \text{ MeV}/c^2.$$

Masses and Binding energies

The nucleon masses are

$$m_p = 938.27203(8) \text{ MeV}/c^2 = 1.00727646688(13)u,$$

and

$$m_n = 939.56536(8) \text{ MeV}/c^2 = 1.0086649156(6)u.$$

In the 2003 mass evaluation there are 2127 nuclei measured with an accuracy of 0.2 MeV or better, and 101 nuclei measured with an accuracy of greater than 0.2 MeV. For heavy nuclei one observes several chains of nuclei with a constant $N - Z$ value whose masses are obtained from the energy released in α-decay.

Masses and Binding energies

The nuclear binding energy is defined as the energy required to break up a given nucleus into its constituent parts of N neutrons and Z protons. In terms of the atomic masses $M(N,Z)$ the binding energy is defined by

$$BE(N,Z) = ZM_H c^2 + Nm_n c^2 - M(N,Z)c^2,$$

where M_H is the mass of the hydrogen atom and m_n is the mass of the neutron. In terms of the mass excess the binding energy is given by

$$BE(N,Z) = Z\Delta_H c^2 + N\Delta_n c^2 - \Delta(N,Z)c^2,$$

where $\Delta_H c^2 = 7.2890$ MeV and $\Delta_n c^2 = 8.0713$ MeV.

Liquid drop model as a simple parametrization of binding energies

A popular and physically intuitive model which can be used to parametrize the experimental binding energies as function of A, is the so-called the liquid drop model. The ansatz is based on the following expression

$$BE(N,Z) = a_1 A - a_2 A^{2/3} - a_3 \frac{Z^2}{A^{1/3}} - a_4 \frac{(N-Z)^2}{A},$$

where A stands for the number of nucleons and the a_is are parameters which are determined by a fit to the experimental data.

Liquid drop model as a simple parametrization of binding energies

To arrive at the above expression we have assumed that we can make the following assumptions:

- There is a volume term $a_1 A$ proportional with the number of nucleons (the energy is also an extensive quantity). When an assembly of nucleons of the same size is packed together into the smallest volume, each interior nucleon has a certain number of other nucleons in contact with it. This contribution is proportional to the volume.
- There is a surface energy term $a_2 A^{2/3}$. The assumption here is that a nucleon at the surface of a nucleus interacts with fewer other nucleons than one in the interior of the nucleus and hence its binding energy is less. This surface energy term takes that into account and is therefore negative and is proportional to the surface area.

Liquid drop model as a simple parametrization of binding energies, continues

- There is a Coulomb energy term $a_3 \frac{Z^2}{A^{1/3}}$. The electric repulsion between each pair of protons in a nucleus yields less binding.
- There is an asymmetry term $a_4 \frac{(N-Z)^2}{A}$. This term is associated with the Pauli exclusion principle and reflectd the fact that the proton-neutron interaction is more attractive on the average than the neutron-neutron and proton-proton interactions.

We could also add a so-called pairing term (an important model in nuclear physics), which is a correction term that arises from the tendency of proton pairs and neutron pairs to occur. An even number of particles is more stable than an odd number.

Masses and Binding energies

The following python program reads now in the experimental data on binding energies and performs a least squares fit to the data using a Machine Learning library called **Scikit-Learn** Let us start with reading and organizing our data. We start with the compilation of masses and binding energies from 2016. After having downloaded this file to our own computer, we are now ready to read the file and start structuring our data.

We start with preparing folders for storing our calculations and the data file over masses and binding energies. We import also various modules that we will

find useful in order to present various Machine Learning methods. Here we focus mainly on the functionality of **Scikit-Learn**.

```
# Common imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.linear_model as skl
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import os

# Where to save the figures and data files
PROJECT_ROOT_DIR = "Results"
FIGURE_ID = "Results/FigureFiles"
DATA_ID = "DataFiles/"

if not os.path.exists(PROJECT_ROOT_DIR):
    os.mkdir(PROJECT_ROOT_DIR)

if not os.path.exists(FIGURE_ID):
    os.makedirs(FIGURE_ID)

if not os.path.exists(DATA_ID):
    os.makedirs(DATA_ID)

def image_path(fig_id):
    return os.path.join(FIGURE_ID, fig_id)

def data_path(dat_id):
    return os.path.join(DATA_ID, dat_id)

def save_fig(fig_id):
    plt.savefig(image_path(fig_id) + ".png", format='png')

infile = open(data_path("MassEval2016.dat"),'r')
```

Before we proceed, we define also a function for making our plots. You can obviously avoid this and simply set up various **matplotlib** commands every time you need them. You may however find it convenient to collect all such commands in one function and simply call this function.

```
from pylab import plt, mpl
plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'

def MakePlot(x,y, styles, labels, axlabels):
    plt.figure(figsize=(10,6))
    for i in range(len(x)):
        plt.plot(x[i], y[i], styles[i], label = labels[i])
        plt.xlabel(axlabels[0])
        plt.ylabel(axlabels[1])
    plt.legend(loc=0)
```

Our next step is to read the data on experimental binding energies and reorganize them as functions of the mass number A, the number of protons Z and neutrons N using **pandas**. Before we do this it is always useful (unless you

have a binary file or other types of compressed data) to actually open the file and simply take a look at it!

In particular, the program that outputs the final nuclear masses is written in Fortran with a specific format. It means that we need to figure out the format and which columns contain the data we are interested in. Pandas comes with a function that reads formatted output. After having admired the file, we are now ready to start massaging it with **pandas**. The file begins with some basic format information.

```
"""
This is taken from the data file of the mass 2016 evaluation.
All files are 3436 lines long with 124 character per line.
       Headers are 39 lines long.
   col 1     :  Fortran character control: 1 = page feed  0 = line feed
   format    :  a1,i3,i5,i5,i5,1x,a3,a4,1x,f13.5,f11.5,f11.3,f9.3,1x,a2,f11.3,f9.3,1x,i3,1x,f12.5
   These formats are reflected in the pandas widths variable below, see the statement
   widths=(1,3,5,5,5,1,3,4,1,13,11,11,9,1,2,11,9,1,3,1,12,11,1),
   Pandas has also a variable header, with length 39 in this case.
"""
```

The data we are interested in are in columns 2, 3, 4 and 11, giving us the number of neutrons, protons, mass numbers and binding energies, respectively. We add also for the sake of completeness the element name. The data are in fixed-width formatted lines and we will covert them into the **pandas** DataFrame structure.

```
# Read the experimental data with Pandas
Masses = pd.read_fwf(infile, usecols=(2,3,4,6,11),
              names=('N', 'Z', 'A', 'Element', 'Ebinding'),
              widths=(1,3,5,5,5,1,3,4,1,13,11,11,9,1,2,11,9,1,3,1,12,11,1),
              header=39,
              index_col=False)

# Extrapolated values are indicated by '#' in place of the decimal place, so
# the Ebinding column won't be numeric. Coerce to float and drop these entries.
Masses['Ebinding'] = pd.to_numeric(Masses['Ebinding'], errors='coerce')
Masses = Masses.dropna()
# Convert from keV to MeV.
Masses['Ebinding'] /= 1000

# Group the DataFrame by nucleon number, A.
Masses = Masses.groupby('A')
# Find the rows of the grouped DataFrame with the maximum binding energy.
Masses = Masses.apply(lambda t: t[t.Ebinding==t.Ebinding.max()])
```

We have now read in the data, grouped them according to the variables we are interested in. We see how easy it is to reorganize the data using **pandas**. If we were to do these operations in C/C++ or Fortran, we would have had to write various functions/subroutines which perform the above reorganizations for us. Having reorganized the data, we can now start to make some simple fits using both the functionalities in **numpy** and **Scikit-Learn** afterwards.

Now we define five variables which contain the number of nucleons A, the number of protons Z and the number of neutrons N, the element name and finally the energies themselves.

```
A = Masses['A']
Z = Masses['Z']
N = Masses['N']
Element = Masses['Element']
Energies = Masses['Ebinding']
print(Masses)
```

The next step, and we will define this mathematically later, is to set up the so-called **design matrix**. We will throughout call this matrix $\boldsymbol{X}$. It has dimensionality $p \times n$, where n is the number of data points and p are the so-called predictors. In our case here they are given by the number of polynomials in A we wish to include in the fit.

```
# Now we set up the design matrix X
X = np.zeros((len(A),5))
X[:,0] = 1
X[:,1] = A
X[:,2] = A**(2.0/3.0)
X[:,3] = A**(-1.0/3.0)
X[:,4] = A**(-1.0)
```

With **scikitlearn** we are now ready to use linear regression and fit our data.

```
clf = skl.LinearRegression().fit(X, Energies)
fity = clf.predict(X)
```

Pretty simple! Now we can print measures of how our fit is doing, the coefficients from the fits and plot the final fit together with our data.

```
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(Energies, fity))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(Energies, fity))
# Mean absolute error
print('Mean absolute error: %.2f' % mean_absolute_error(Energies, fity))
print(clf.coef_, clf.intercept_)

Masses['Eapprox']  = fity
# Generate a plot comparing the experimental with the fitted values values.
fig, ax = plt.subplots()
ax.set_xlabel(r'$A = N + Z$')
ax.set_ylabel(r'$E_\mathrm{bind}\,/\mathrm{MeV}$')
ax.plot(Masses['A'], Masses['Ebinding'], alpha=0.7, lw=2,
            label='Ame2016')
ax.plot(Masses['A'], Masses['Eapprox'], alpha=0.7, lw=2, c='m',
            label='Fit')
ax.legend()
save_fig("Masses2016")
plt.show()
```

Q-values and separation energies

We are now interested in interpreting experimental binding energies in terms of a single-particle picture. In order to do so, we consider first energy conservation

for nuclear transformations that include, for example, the fusion of two nuclei a and b into the combined system c

$$^{N_a+Z_a}a + {}^{N_b+Z_b}b \rightarrow {}^{N_c+Z_c}c$$

or the decay of nucleus c into two other nuclei a and b

$$^{N_c+Z_c}c \rightarrow {}^{N_a+Z_a}\ a + {}^{N_b+Z_b}\ b$$

Q-values and separation energies

In general we have the reactions

$$\sum_i {}^{N_i+Z_i}i \rightarrow \sum_f {}^{N_f+Z_f}f$$

We require also that the number of protons and neutrons (the total number of nucleons) is conserved in the initial stage and final stage, unless we have processes which violate baryon conservation,

$$\sum_i N_i = \sum_f N_f \ \text{ and } \ \sum_i Z_i = \sum_f Z_f.$$

Motivation

Do we understand the physics of dripline systems? Artist's rendition of the emission of one proton from various oxygen isotopes. Protons are in red while neutrons are in blue. These processes could be interpreted as the decay nucleus c into two other nuclei a and b

$$^{N_c+Z_c}c \rightarrow {}^{N_a+Z_a}\ a + {}^{N_b+Z_b}\ b.$$

Q-values and separation energies

The above processes can be characterized by an energy difference called the Q value, defined as

$$Q = \sum_i M(N_i, Z_i)c^2 - \sum_f M(N_f, Z_f)c^2 = \sum_i BE(N_f, Z_f) - \sum_i BE(N_i, Z_i)$$

Spontaneous decay involves a single initial nuclear state and is allowed if $Q > 0$. In the decay, energy is released in the form of the kinetic energy of the final products. Reactions involving two initial nuclei are called endothermic (a net loss of energy) if $Q < 0$. The reactions are exothermic (a net release of energy) if $Q > 0$.

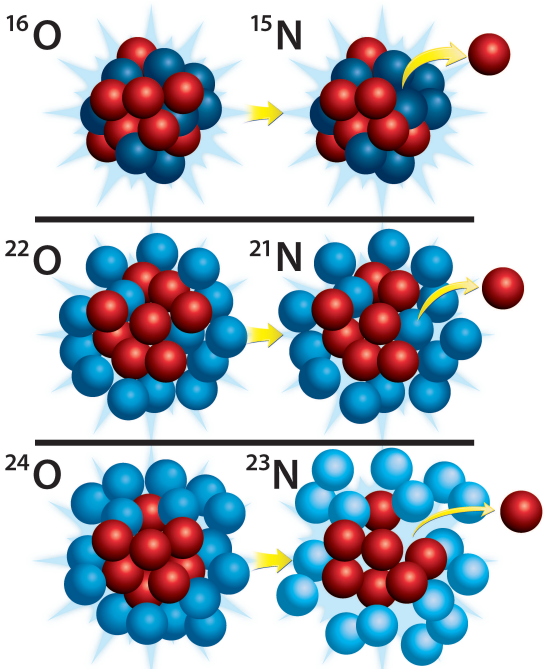

Figure 2: Artist's rendition of the emission of one proton from various oxygen isotopes.

Q-values and separation energies

Let us study the Q values associated with the removal of one or two nucleons from a nucleus. These are conventionally defined in terms of the one-nucleon and two-nucleon separation energies. The neutron separation energy is defined as

$$S_n = -Q_n = BE(N,Z) - BE(N-1,Z),$$

and the proton separation energy reads

$$S_p = -Q_p = BE(N,Z) - BE(N,Z-1).$$

The two-neutron separation energy is defined as

$$S_{2n} = -Q_{2n} = BE(N,Z) - BE(N-2,Z),$$

and the two-proton separation energy is given by

$$S_{2p} = -Q_{2p} = BE(N,Z) - BE(N,Z-2).$$

Separation energies and energy gaps

Using say the neutron separation energies (alternatively the proton separation energies)

$$S_n = -Q_n = BE(N,Z) - BE(N-1,Z),$$

we can define the so-called energy gap for neutrons (or protons) as

$$\Delta S_n = BE(N,Z) - BE(N-1,Z) - \left(BE(N+1,Z) - BE(N,Z)\right),$$

or

$$\Delta S_n = 2BE(N,Z) - BE(N-1,Z) - BE(N+1,Z).$$

This quantity can in turn be used to determine which nuclei are magic or not. For protons we would have

$$\Delta S_p = 2BE(N,Z) - BE(N,Z-1) - BE(N,Z+1).$$

We leave it as an exercise to the reader to define and interpret the two-neutron or two-proton gaps.

Separation energies

To calculate say the neutron separation we need to multiply our masses with the nucleon number A (why?). Thereafter we pick the oxygen isotopes and simply compute the separation energies with two lines of code (note that most of the code here is a repeat of what you have seen before).

```
# Common imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from pylab import plt, mpl
plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'

def MakePlot(x,y, styles, labels, axlabels):
    plt.figure(figsize=(10,6))
    for i in range(len(x)):
        plt.plot(x[i], y[i], styles[i], label = labels[i])
        plt.xlabel(axlabels[0])
        plt.ylabel(axlabels[1])
    plt.legend(loc=0)


# Where to save the figures and data files
PROJECT_ROOT_DIR = "Results"
FIGURE_ID = "Results/FigureFiles"
DATA_ID = "DataFiles/"

if not os.path.exists(PROJECT_ROOT_DIR):
    os.mkdir(PROJECT_ROOT_DIR)
```

```
if not os.path.exists(FIGURE_ID):
    os.makedirs(FIGURE_ID)

if not os.path.exists(DATA_ID):
    os.makedirs(DATA_ID)

def image_path(fig_id):
    return os.path.join(FIGURE_ID, fig_id)

def data_path(dat_id):
    return os.path.join(DATA_ID, dat_id)

def save_fig(fig_id):
    plt.savefig(image_path(fig_id) + ".png", format='png')

infile = open(data_path("MassEval2016.dat"),'r')


# Read the experimental data with Pandas
Masses = pd.read_fwf(infile, usecols=(2,3,4,6,11),
              names=('N', 'Z', 'A', 'Element', 'Ebinding'),
              widths=(1,3,5,5,5,1,3,4,1,13,11,11,9,1,2,11,9,1,3,1,12,11,1),
              header=39,
              index_col=False)

# Extrapolated values are indicated by '#' in place of the decimal place, so
# the Ebinding column won't be numeric. Coerce to float and drop these entries.
Masses['Ebinding'] = pd.to_numeric(Masses['Ebinding'], errors='coerce')
Masses = Masses.dropna()
# Convert from keV to MeV.
Masses['Ebinding'] /= 1000
A = Masses['A']
Z = Masses['Z']
N = Masses['N']
Element = Masses['Element']
Energies = Masses['Ebinding']*A

df = pd.DataFrame({'A':A,'Z':Z, 'N':N,'Element':Element,'Energies':Energies})
# Her we pick the oyxgen isotopes
Nucleus = df.loc[lambda df: df.Z==8, :]
# drop cases with no number
Nucleus = Nucleus.dropna()
# Here we do the magic and obtain the neutron separation energies, one line of code!!
Nucleus['NeutronSeparationEnergies'] = Nucleus['Energies'].diff(+1)
print(Nucleus)
MakePlot([Nucleus.A], [Nucleus.NeutronSeparationEnergies], ['b'], ['Neutron Separation Energy'],
save_fig('Nucleus')
plt.show()
```

Features to be noted

Since we will focus in the beginning on single-particle degrees of freedom and mean-field approaches there are some features to be noted

- In the discussion of the liquid drop model and binding energies, we note that the total binding energy is not that different from the sum of the individual neutron and proton masses.

One may thus infer that intrinsic properties of nucleons in a nucleus are close to those of free nucleons.

- In the discussion of the neutron separation energies for the oxygen isotopes, we note a clear staggering effect between odd and even isotopes with the even ones being more bound (larger separation energies). This is linked with strong pairing correlations in nuclei arising from the nuclear force.

Features to be noted, continues

- The neutron separation energy becomes negative at ^{25}O, making this nucleus unstable with respect to the emission of one neutron. A nucleus like ^{24}O is thus the last stable oxygen isotopes which has been observed. Oxygen-26 has been to be unbound with respect to ^{24}O.
- We note also that there are large shell-gaps for some nuclei, meaning that more energy is needed to remove one nucleon. These gaps are used to define so-called magic numbers. For the oxygen isotopes we see a clear gap for ^{16}O. We will interpret this gap as one of several experimental properties that define so-called magic numbers. In our discussion below we will make a first interpretation using single-particle states from the harmonic oscillator and the Woods-Saxon potential.

In the exercise below you will be asked to perform a similar analysis for other chains of isotopes and interpret the results.

*

Exercise 1: Masses and binding energies

The data on binding energies can be found in the file bedata.dat (go to the datafiles folder)
aragraph!paragraph>paragraph>-0.5em

a) Write a small program which reads in the proton and neutron numbers and the binding energies and make a plot of all neutron separation energies for the chain of oxygen (O), calcium (Ca), nickel (Ni), tin (Sn) and lead (Pb) isotopes, that is you need to plot

$$S_n = BE(N,Z) - BE(N-1,Z).$$

Comment your results.

aragraph!paragraph>paragraph>-0.5em

b) Include in the same figure(s) the liquid drop model results, namely

$$BE(N,Z) = \alpha_1 A - \alpha_2 A^{2/3} - \alpha_3 \frac{Z^2}{A^{1/3}} - \alpha_4 \frac{(N-Z)^2}{A},$$

with $\alpha_1 = 15.49$ MeV, $\alpha_2 = 17.23$ MeV, $\alpha_3 = 0.697$ MeV and $\alpha_4 = 22.6$ MeV. Comment your results

aragraph!paragraph>paragraph>-0.5em

c) Make a plot of the binding energies as function of the number of nucleons A using the data in the file on bindingenergies and the above liquid drop model.

aragraph!paragraph>paragraph>-0.5em

d) Use the liquid drop model to find the neutron drip lines for Z values up to 120. Analyze then the fluorine isotopes and find, where available the corresponding experimental data, and compare the liquid drop model predicition with experiment. Comment your results.

Our first attempt at a model, harmonic oscillator spectrum

The Woods-Saxon potential

The Woods-Saxon potential is a mean field potential for the nucleons (protons and neutrons) inside an atomic nucleus. It represent an average potential that a given nucleon feels from the forces applied on each nucleon. The parametrization is

$$\hat{u}_{\text{ext}}(r) = -\frac{V_0}{1+\exp{(r-R)/a}},$$

with $V_0 \approx 50$ MeV representing the potential well depth, $a \approx 0.5$ fm length representing the "surface thickness" of the nucleus and $R = r_0 A^{1/3}$, with $r_0 = 1.25$ fm and A the number of nucleons. The value for r_0 can be extracted from a fit to data, see for example M. Kirson's article.

The Woods-Saxon potential

The following python code produces a plot of the Woods-Saxon potential with the above parameters.

```
import numpy as np
from  matplotlib import pyplot as plt
from matplotlib import rc, rcParams
import matplotlib.units as units
import matplotlib.ticker as ticker
rc('text',usetex=True)
rc('font',**{'family':'serif','serif':['Woods-Saxon potential']})
font = {'family' : 'serif',
```

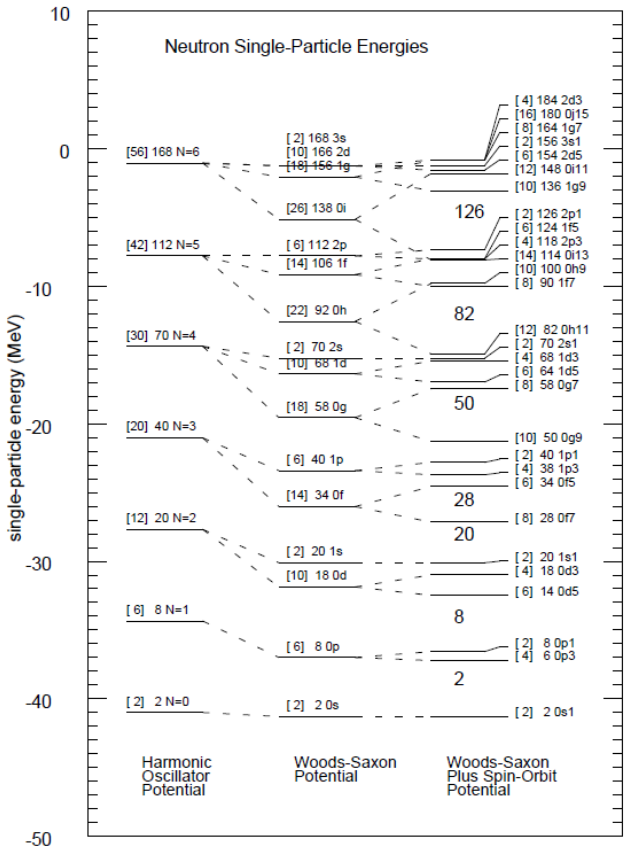

Figure 3: Single-particle spectrum and quantum numbers for a harmonic oscillator potential and a Woods-Saxon potential with and without a spin-orbit force.

```
        'color'  : 'darkred',
        'weight' : 'normal',
        'size'   : 16,
        }
v0 = 50
A = 100
a = 0.5
r0 = 1.25
R = r0*A**(0.3333)
x = np.linspace(0.0, 10.0)
y = -v0/(1+np.exp((x-R)/a))

plt.plot(x, y, 'b-')
plt.title(r'{\bf Woods-Saxon potential}', fontsize=20)
plt.text(3, -40, r'Parameters: $A=20$, $V_0=50$ [MeV]', fontdict=font)
plt.text(3, -44, r'$a=0.5$ [fm], $r_0=1.25$ [fm]', fontdict=font)
plt.xlabel(r'$r$ [fm]',fontsize=20)
```

```
plt.ylabel(r'$V(r)$ [MeV]',fontsize=20)

# Tweak spacing to prevent clipping of ylabel
plt.subplots_adjust(left=0.15)
```

From the plot we notice that the potential

- rapidly approaches zero as r goes to infinity, reflecting the short-distance nature of the strong nuclear force.
- For large A, it is approximately flat in the center.
- Nucleons near the surface of the nucleus experience a large force towards the center.

Single-particle Hamiltonians and spin-orbit force

We have introduced a single-particle Hamiltonian

$$H_0 = \sum_{i=1}^{A} \hat{h}_0(x_i) = \sum_{i=1}^{A} \left(\hat{t}(x_i) + \hat{u}_{\text{ext}}(x_i) \right),$$

with an external and central symmetric potential $u_{\text{ext}}(x_i)$, which is often approximated by a harmonic oscillator potential or a Woods-Saxon potential. Being central symmetric leads to a degeneracy in energy which is not observed experimentally. We see this from for example our discussion of separation energies and magic numbers. There are, in addition to the assumed magic numbers from a harmonic oscillator basis of $2, 8, 20, 40, 70 \ldots$ magic numbers like 28, 50, 82 and 126.

To produce these additional numbers, we need to add a phenomenological spin-orbit force which lifts the degeneracy, that is

$$\hat{h}(x_i) = \hat{t}(x_i) + \hat{u}_{\text{ext}}(x_i) + \xi(\boldsymbol{r})\boldsymbol{ls} = \hat{h}_0(x_i) + \xi(\boldsymbol{r})\boldsymbol{ls}.$$

Single-particle Hamiltonians and spin-orbit force

We have introduced a modified single-particle Hamiltonian

$$\hat{h}(x_i) = \hat{t}(x_i) + \hat{u}_{\text{ext}}(x_i) + \xi(\boldsymbol{r})\boldsymbol{ls} = \hat{h}_0(x_i) + \xi(\boldsymbol{r})\boldsymbol{ls}.$$

We can calculate the expectation value of the latter using the fact that

$$\xi(\boldsymbol{r})\boldsymbol{ls} = \frac{1}{2}\xi(\boldsymbol{r}) \left(\boldsymbol{j}^2 - \boldsymbol{l}^2 - \boldsymbol{s}^2 \right).$$

For a single-particle state with quantum numbers nlj (we suppress s and m_j), with $s = 1/2$, we obtain the single-particle energies

$$\varepsilon_{nlj} = \varepsilon^{(0)}_{nlj} + \Delta\varepsilon_{nlj},$$

with $\varepsilon^{(0)}_{nlj}$ being the single-particle energy obtained with $\hat{h}_0(x)$ and

$$\Delta\varepsilon_{nlj} = \frac{C}{2} \left(j(j+1) - l(l+1) - \frac{3}{4} \right).$$

Single-particle Hamiltonians and spin-orbit force

The spin-orbit force gives thus an additional contribution to the energy

$$\Delta\varepsilon_{nlj} = \frac{C}{2}\left(j(j+1) - l(l+1) - \frac{3}{4}\right),$$

which lifts the degeneracy we have seen before in the harmonic oscillator or Woods-Saxon potentials. The value C is the radial integral involving $\xi(\boldsymbol{r})$. Depending on the value of $j = l \pm 1/2$, we obtain

$$\Delta\varepsilon_{nlj=l-1/2} = \frac{C}{2}l,$$

or

$$\Delta\varepsilon_{nlj=l+1/2} = -\frac{C}{2}(l+1),$$

clearly lifting the degeneracy. Note well that till now we have simply postulated the spin-orbit force in *ad hoc* way. This term arises from the nuclear force in a natural way.

Single-particle Hamiltonians and spin-orbit force

With the spin-orbit force, we can modify our Woods-Saxon potential to

$$\hat{u}_{\mathrm{ext}}(r) = -\frac{V_0}{1+\exp{(r-R)/a}} + V_{so}(r)\boldsymbol{ls},$$

with

$$V_{so}(r) = V_{so}\frac{1}{r}\frac{df_{so}(r)}{dr},$$

where we have

$$f_{so}(r) = \frac{1}{1+\exp{(r-R_{so})/a_{so}}}.$$

Single-particle Hamiltonians and spin-orbit force

We can also add, in case of proton, a Coulomb potential. The Woods-Saxon potential has been widely used in parametrizations of effective single-particle potentials.

However, as was the case with the harmonic oscillator, none of these potentials are linked directly to the nuclear forces.

This is solved by building a mean field based on the nucleon-nucleon interaction and forms the basis for the discussions of Brown and Hergert.

Single-particle Hamiltonians and spin-orbit force

The Woods-Saxon potential does allow for closed-form or analytical solutions of the eigenvalue problem

$$\hat{h}_0(x_i)\psi_\alpha(x_i) = \varepsilon_\alpha\psi_\alpha(x_i).$$

For the harmonic oscillator in three dimensions we have closed-form expressions for the energies and analytical solutions for the eigenstates, with the latter given by either Hermite polynomials (cartesian coordinates) or Laguerre polynomials (spherical coordinates).

To solve the above equation is however rather straightforward numerically.

Numerical solution of the single-particle Schroedinger equation

We will illustrate the numerical solution of Schroedinger's equation by solving it for the harmonic oscillator in three dimensions. It is straightforward to change the harmonic oscillator potential with a Woods-Saxon potential, or any other type of potentials.

We are interested in the solution of the radial part of Schroedinger's equation for one nucleon. The angular momentum part is given by the so-called Spherical harmonics.

The radial equation reads

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = ER(r).$$

Numerical solution of the single-particle Schroedinger equation

In our case $V(r)$ is the harmonic oscillator potential $(1/2)kr^2$ with $k = m\omega^2$ and E is the energy of the harmonic oscillator in three dimensions. The oscillator frequency is ω and the energies are

$$E_{nl} = \hbar\omega\left(2n + l + \frac{3}{2}\right),$$

with $n = 0, 1, 2, \ldots$ and $l = 0, 1, 2, \ldots$.

Numerical solution of the single-particle Schroedinger equation

Since we have made a transformation to spherical coordinates it means that $r \in [0, \infty)$. The quantum number l is the orbital momentum of the nucleon. Then we substitute $R(r) = (1/r)u(r)$ and obtain

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(V(r) + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r).$$

The boundary conditions are $u(0) = 0$ and $u(\infty) = 0$.

Numerical solution of the single-particle Schroedinger equation

We introduce a dimensionless variable $\rho = (1/\alpha)r$ where α is a constant with dimension length and get

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \left(V(\rho) + \frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2}\right)u(\rho) = Eu(\rho).$$

Let us specialize to $l = 0$. Inserting $V(\rho) = (1/2)k\alpha^2\rho^2$ we end up with

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \frac{k}{2}\alpha^2\rho^2u(\rho) = Eu(\rho).$$

We multiply thereafter with $2m\alpha^2/\hbar^2$ on both sides and obtain

$$-\frac{d^2}{d\rho^2}u(\rho) + \frac{mk}{\hbar^2}\alpha^4\rho^2u(\rho) = \frac{2m\alpha^2}{\hbar^2}Eu(\rho).$$

Numerical solution of the single-particle Schroedinger equation

We have thus

$$-\frac{d^2}{d\rho^2}u(\rho) + \frac{mk}{\hbar^2}\alpha^4\rho^2u(\rho) = \frac{2m\alpha^2}{\hbar^2}Eu(\rho).$$

The constant α can now be fixed so that

$$\frac{mk}{\hbar^2}\alpha^4 = 1,$$

or

$$\alpha = \left(\frac{\hbar^2}{mk}\right)^{1/4}.$$

Numerical solution of the single-particle Schroedinger equation

Defining

$$\lambda = \frac{2m\alpha^2}{\hbar^2}E,$$

we can rewrite Schroedinger's equation as

$$-\frac{d^2}{d\rho^2}u(\rho) + \rho^2u(\rho) = \lambda u(\rho).$$

This is the first equation to solve numerically. In three dimensions the eigenvalues for $l = 0$ are $\lambda_0 = 3, \lambda_1 = 7, \lambda_2 = 11, \ldots$.

Numerical solution of the single-particle Schroedinger equation

We use the standard expression for the second derivative of a function u

$$u'' = \frac{u(\rho + h) - 2u(\rho) + u(\rho - h)}{h^2} + O(h^2), \tag{1}$$

where h is our step. Next we define minimum and maximum values for the variable ρ, $\rho_{\rm min} = 0$ and $\rho_{\rm max}$, respectively. You need to check your results for the energies against different values $\rho_{\rm max}$, since we cannot set $\rho_{\rm max} = \infty$.

Numerical solution of the single-particle Schroedinger equation

With a given number of steps, $n_{\rm step}$, we then define the step h as

$$h = \frac{\rho_{\rm max} - \rho_{\rm min}}{n_{\rm step}}.$$

Define an arbitrary value of ρ as

$$\rho_i = \rho_{\rm min} + ih \qquad i = 0, 1, 2, \ldots, n_{\rm step}$$

we can rewrite the Schroedinger equation for ρ_i as

$$-\frac{u(\rho_i + h) - 2u(\rho_i) + u(\rho_i - h)}{h^2} + \rho_i^2 u(\rho_i) = \lambda u(\rho_i),$$

or in a more compact way

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \rho_i^2 u_i = -\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + V_i u_i = \lambda u_i.$$

Numerical solution of the single-particle Schroedinger equation

Define first the diagonal matrix element

$$d_i = \frac{2}{h^2} + V_i,$$

and the non-diagonal matrix element

$$e_i = -\frac{1}{h^2}.$$

In this case the non-diagonal matrix elements are given by a mere constant. *All non-diagonal matrix elements are equal.*

Numerical solution of the single-particle Schroedinger equation

With these definitions the Schroedinger equation takes the following form

$$d_i u_i + e_{i-1} u_{i-1} + e_{i+1} u_{i+1} = \lambda u_i,$$

where u_i is unknown. We can write the latter equation as a matrix eigenvalue problem

$$\begin{pmatrix} d_1 & e_1 & 0 & 0 & \dots & 0 & 0 \\ e_1 & d_2 & e_2 & 0 & \dots & 0 & 0 \\ 0 & e_2 & d_3 & e_3 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & d_{n_{\rm step}-2} & e_{n_{\rm step}-1} \\ 0 & \dots & \dots & \dots & \dots & e_{n_{\rm step}-1} & d_{n_{\rm step}-1} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ \dots \\ \dots \\ u_{n_{\rm step}-1} \end{pmatrix} = \lambda \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ \dots \\ \dots \\ u_{n_{\rm step}-1} \end{pmatrix} \tag{2}$$

Numerical solution of the single-particle Schroedinger equation

To be more detailed we have

$$\begin{pmatrix} \frac{2}{h^2}+V_1 & -\frac{1}{h^2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2}+V_2 & -\frac{1}{h^2} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2}+V_3 & -\frac{1}{h^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & \frac{2}{h^2}+V_{n_{\rm step}-2} & -\frac{1}{h^2} \\ 0 & \dots & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2}+V_{n_{\rm step}-1} \end{pmatrix} \tag{3}$$

Recall that the solutions are known via the boundary conditions at $i = n_{\rm step}$ and at the other end point, that is for ρ_0. The solution is zero in both cases.

Program to solve Schroedinger's equation

The following python program is an example of how one can obtain the eigenvalues for a single-nucleon moving in a harmonic oscillator potential. It is rather easy to change the onebody-potential with ones like a Woods-Saxon potential.

Program to solve Schroedinger's equation

The code sets up the Hamiltonian matrix by defining the the minimun and maximum values of r with a maximum value of integration points. These are set in the initialization function. It plots the eigenfunctions of the three lowest eigenstates.

```
#Program which solves the one-particle Schrodinger equation
#for a potential specified in function
#potential().

from  matplotlib import pyplot as plt
import numpy as np
#Function for initialization of parameters
def initialize():
    RMin = 0.0
    RMax = 10.0
    lOrbital = 0
    Dim = 400
    return RMin, RMax, lOrbital, Dim
#Harmonic oscillator here, easy to change
def potential(r):
    return 0.5*r*r
    #  return 0.0
    #  return -1.0/r
    #if r >= 0.0 and r <= 10.0:
    #    V = -0.05
    #else:
    #    V =0.0
    #return V


#Get the boundary, orbital momentum and number of integration points
RMin, RMax, lOrbital, Dim = initialize()

#Initialize constants
Step    = RMax/(Dim+1)
DiagConst = 1.0/ (Step*Step)
NondiagConst =  -0.5 / (Step*Step)
OrbitalFactor = 0.5*lOrbital * (lOrbital + 1.0)

#Calculate array of potential values
v = np.zeros(Dim)
r = np.linspace(RMin,RMax,Dim)
for i in range(Dim):
    r[i] = RMin + (i+1) * Step;
    v[i] = potential(r[i]) + OrbitalFactor/(r[i]*r[i]);

#Setting up a tridiagonal matrix and finding eigenvectors and eigenvalues
Matrix = np.zeros((Dim,Dim))
Matrix[0,0] = DiagConst + v[0];
Matrix[0,1] = NondiagConst;
for i in range(1,Dim-1):
    Matrix[i,i-1]  = NondiagConst;
    Matrix[i,i]    = DiagConst + v[i];
    Matrix[i,i+1]  = NondiagConst;
Matrix[Dim-1,Dim-2] = NondiagConst;
Matrix[Dim-1,Dim-1] = DiagConst + v[Dim-1];
# diagonalize and obtain eigenvalues, not necessarily sorted
EigValues, EigVectors = np.linalg.eig(Matrix)
# sort eigenvectors and eigenvalues
permute = EigValues.argsort()
EigValues = EigValues[permute]
EigVectors = EigVectors[:,permute]
# now plot the results for the three lowest lying eigenstates
for i in range(3):
    print(EigValues[i])
FirstEigvector = EigVectors[:,0]
```

```
    SecondEigvector = EigVectors[:,1]
    ThirdEigvector = EigVectors[:,2]
    plt.plot(r, FirstEigvector**2 ,'b-',r, SecondEigvector**2 ,'g-',r, ThirdEigvector**2 ,'r-')
    plt.axis([0,4.6,0.0, 0.025])
    plt.xlabel(r'$r$')
    plt.ylabel(r'Radial probability $r^2|R(r)|^2$')
    plt.title(r'Radial probability distributions for three lowest-lying states')
    plt.show()
```

*

Exercise 2: Eigenstates and eigenvalues of single-particle problems

aragraph!paragraph>paragraph>-0.5em

a) Compute the eigenvalues of the three lowest states with a given orbital momentum and oscillator frequency ω. Study these results as functions of the the maximum value of r and the number of integration points n, starting with $r_{\max} = 10$. Compare the computed ones with the exact values and comment your results.

aragraph!paragraph>paragraph>-0.5em

b) Plot thereafter the eigenfunctions as functions of r for the lowest-lying state with a given orbital momentum l.

aragraph!paragraph>paragraph>-0.5em

c) Replace thereafter the harmonic oscillator potential with a Woods-Saxon potential using the parameters discussed above. Compute the lowest three eigenvalues and plot the eigenfunction of the lowest-lying state. How does this compare with the harmonic oscillator? Comment your results and possible implications for nuclear physics studies.

Where do we go from here?

With a single-particle basis (an orthonormal basis) we can construct a basis of anti-symmetrized and orthogonal many-body states (called Slater determinants). This basis can then be used to study various many-body methods and properties of nuclei. Recall that nuclei are complicated many-body systems. It is an essential input to Brown's and Hergert's talks later this week.

A single-particle basis like the harmonic oscillator basis is also a useful basis for computing integrals involved in the computation of various matrix elements.

Hopefully, this gives you an input to the discussions to come of Alex Brown, Dean Lee and Filomena Nunes. Best wishes to you all and enjoy your time at MSU.