Homework 9; Your Name Here

Due Thursday, December 10th at 10:00am

Worth 12 Course Points

Note: 0.50 points per question will be given for clear and descriptive comments on your code.

For this assignment, you will be turning in the following files.

1. Your .ipynb file with the filename "YourLastName_HW9.ipynb".
2. Your .pdf file that accompanies your .ipynb file.
3. "YourLastName_Polyval.png" for Problem 1.
4. "YourLastName_TBversusAlt.png" for Problem 2.
5. "YourLastName_Bacteria.png" for Problem 3.
6. "YourLastName_PopulationGrowth.png" for Problem 4.
7. "YourLastName_MORFit.png" for Problem 5.
8. "YourLastName_ChGroup.png" for Problem 6.
9. "YourLastName_NuRaCh.png" for Problem 6.

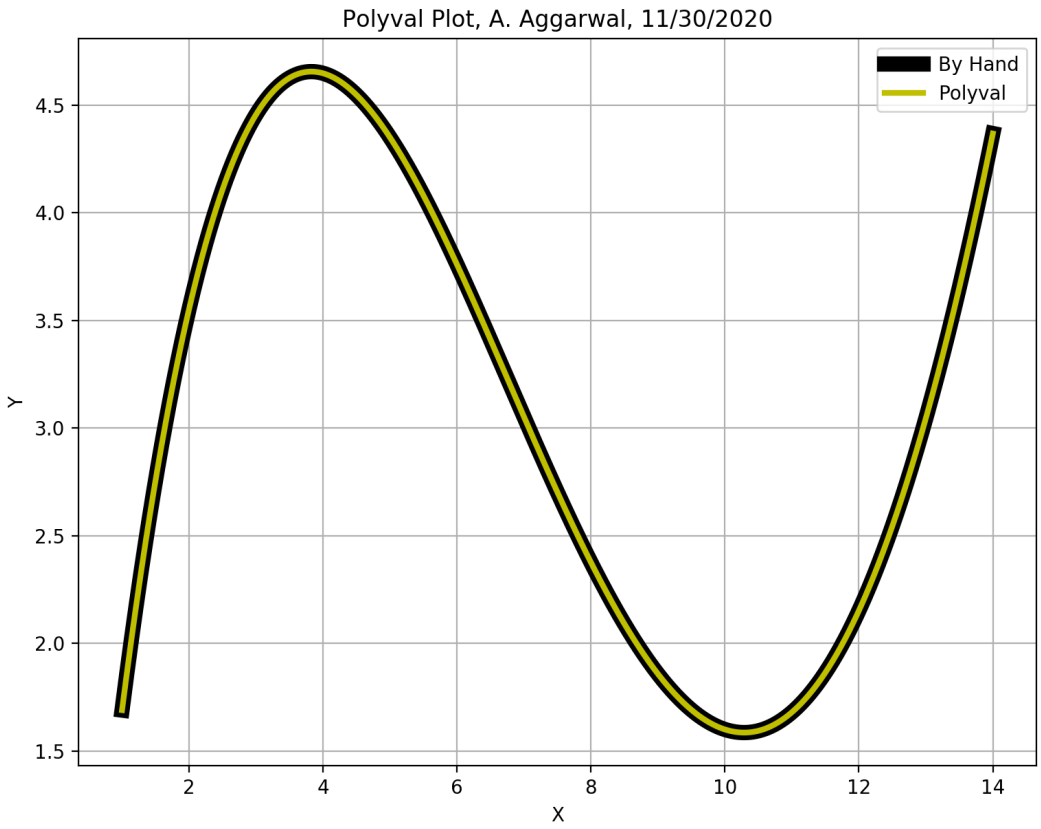
In [2]:
```
from IPython.display import Image      #best way to import images so th
ey render in a PDF
```

Problem 1 (Polynomial Plot; worth 1 point)

1. Define a well-resolved (using greater than 1000 points) array `x` that ranges from 1 to 14 inclusively.
2. We are interested in plotting the polynomial,
   $y = -0.001x^4 + 0.051x^3 - 0.76x^2 + 3.8x - 1.4$ over x.
   First, create an array `y_h` where you code the polynomial y by hand.
3. Second, use the function `np.polyval` (see documentation here (https://numpy.org/doc/stable/reference/generated/numpy.polyval.html)) to define an array `y_p` of the same polynomial.
4. Make a plot of both polynomials.
   - First, plot `x` versus `y_h` using a thick black line (set the linewidth to 8.0 or so). Label this data "By Hand."
   - Next, plot `x` versus `y_p` using a thinner yellow line. Label this data "Polyval."
   - Add a legend.
   - Label your axes.
   - Add a grid.
   - Add a title "Polyval Plot, Your First Initial. Your Last Name, Today's Date."
   - Save the figure as "YourLastName_Polyval.png" with 200 dpi. Upload to CCLE.
5. Finally, sum the difference between `y_h` and `y_p` , and print the result to the screen. Is this the result that you expect? Explain this result in a print line.

In [3]:
```
Image("Aggarwal_Polyval.png", width=700, height=700)
```

Out[3]:


Polyval Plot, A. Aggarwal, 11/30/2020
By Hand
Polyval
Y
X
4.5
4.0
3.5
3.0
2.5
2.0
1.5
2
4
6
8
10
12
14

Problem 2 (Boiling Temperature of Water; worth 1 point)

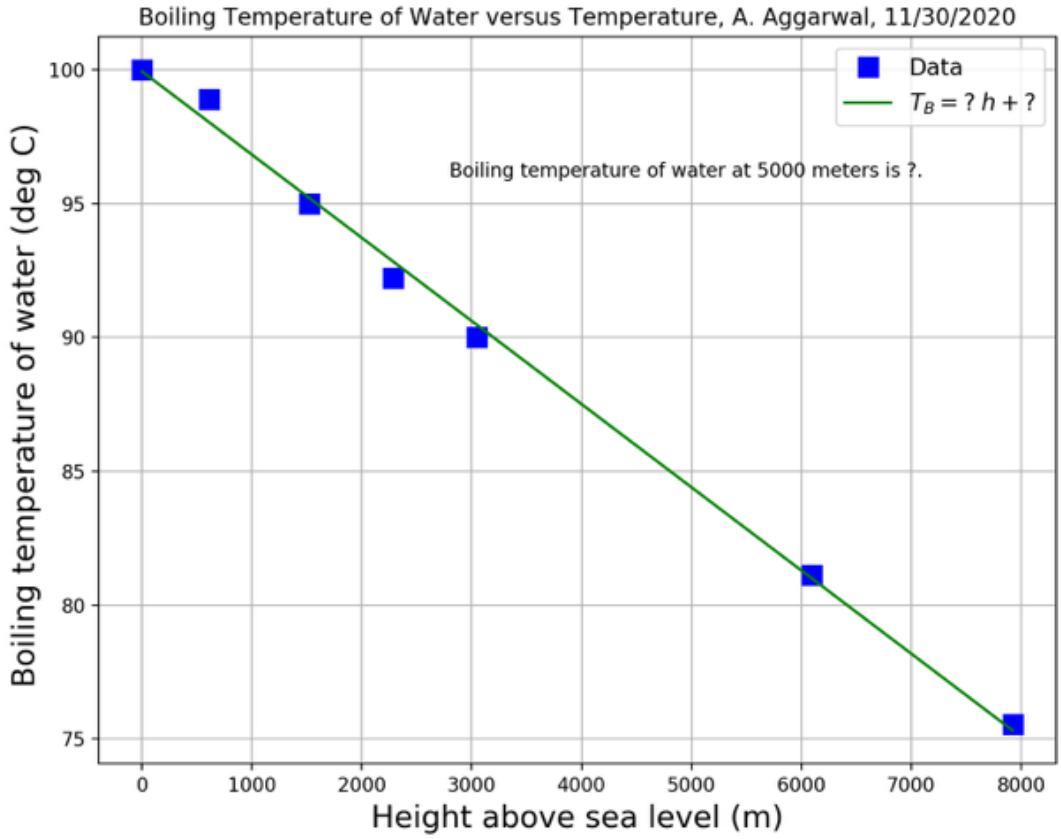
The boiling temperature of water, T_B (deg. Celsius) at various altitudes h (meters) is given in the following table.

h (m)	0	610	1524	2286	3048	6096	7925
T_B (deg. C)	100	98.89	95	92.22	90	81.11	75.56

1. Create two arrays, `h` and `TB` , for the height (meters) and temperature (degrees Celsius) given in the table.
2. Determine the best fit linear equation to the data in the form $T_B = m\ h + b$.
3. Make a nice plot of the data points and the best fit.
   - Plot the data points as blue squares.
   - Plot the best fit as a green line.
   - Include a legend, where the labels include the data and the equation for the best fit line.
     - Use string formatting to include the the coeffcients m and b out to four digits after the decimal in the label for the best fit line. Your values of m and b will replace the question marks in the figure below.
   - Label your axes, turn the grid on, add a title, etc.
4. What is the boiling temperature of water, reported to once place after the decimal, at an altitude of 5000m? Use your best fit equation to determine the result. Include this information on the plot using the `plt.text` command.
5. Save the figure as "YourLastName_TBversusAlt.png" at dpi=200 and upload to CCLE.

In [5]:
```
Image("Aggarwal_TBversusAlt.png", width=700, height=700)
```

Out[5]:



Problem 3 (Bacterial Growth in a Petri Dish; worth 1 point)

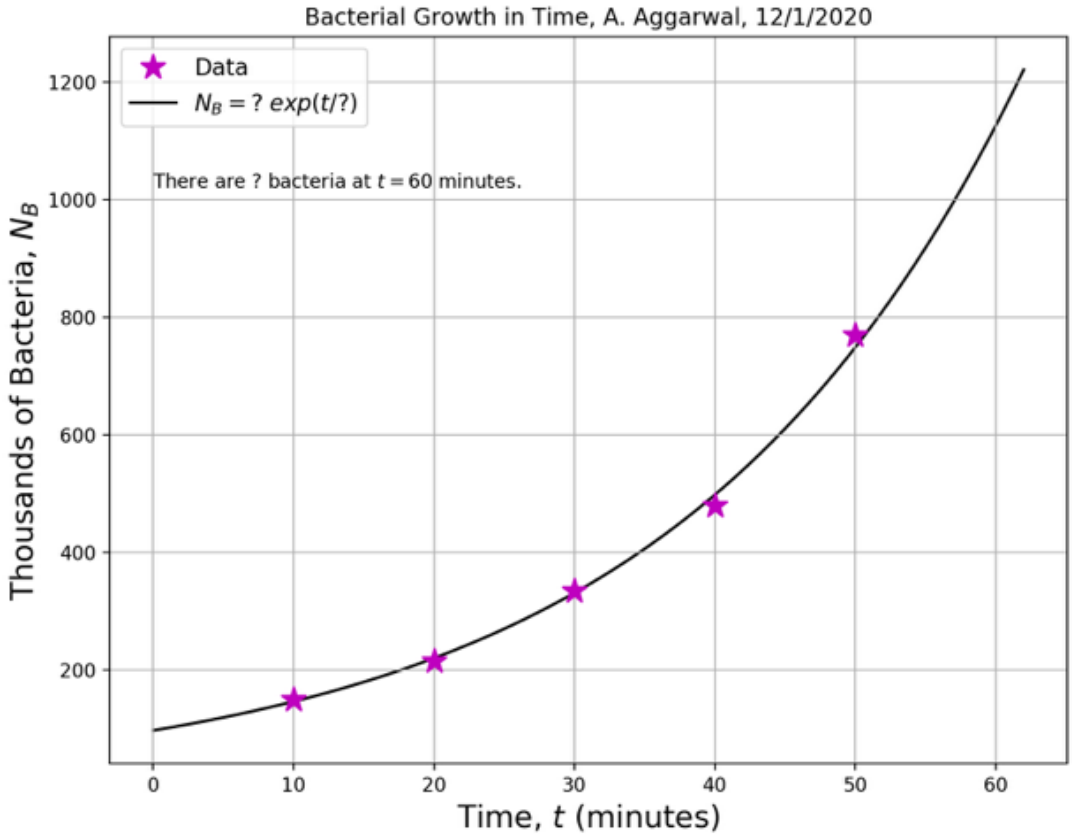
The number of bacteria, N_B, measured in the thousands in a petri dish at different times t (minutes) are given in the following table.

t (min)	10	20	30	40	50
N_B/1000	150	215	335	480	770

1. Create two arrays, `t` and `N_B` for the time and number of bacteria given in the table.
2. Fit the data to an exponential function of the form:
   $N_B = N_0 exp(t/\tau)$
   where N_0 are the number of bacteria at $t = 0$ and the greek letter tau (τ) is the exponential growth time for the bacteria (aka, the e-folding time (https://en.wikipedia.org/wiki/E-folding)).
3. Make a nice plot of the data points and the best fit.
   - Plot the data points as magenta stars.
   - Plot the best fit as a black line behind the data points.
   - Include a legend, where the labels include the data and the equation for the best fit line.
     - Use string formatting to include the the coeffcients N_0 and τ out to one digit after the decimal in the label for the best fit line. Again, your answers will replace the question marks in the figure below.
   - Label your axes, turn the grid on, add a title, etc.
4. What is the number of bacteria at 60 minutes? Use your best fit equation to determine the result. Include this information on the plot using the `plt.text` command.
5. Save the figure as "YourLastName_Bacteria.png" at dpi=200 and upload to CCLE.

In [20]:
```
Image("Aggarwal_Bacteria.png", width=700, height=700)
```

Out[20]:



Problem 4 (Global Population Growth; worth 2 points)

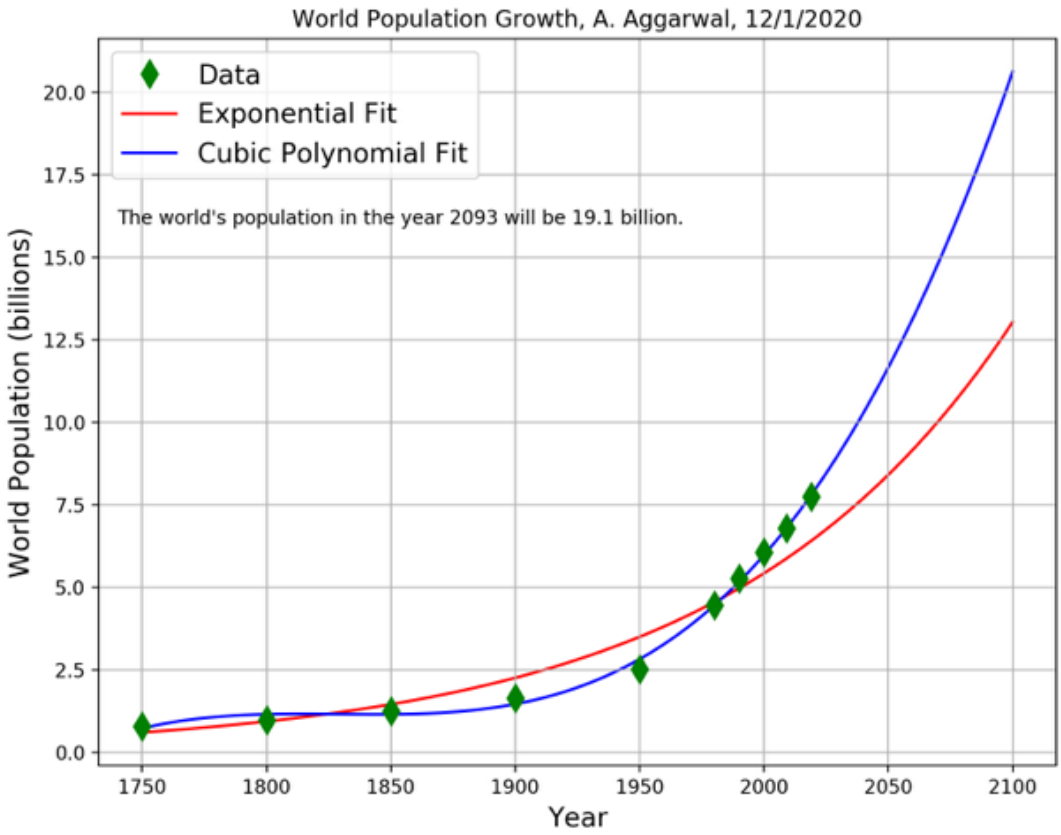
The population of the world for selected years from 1750 to 2019 is given in the following table:

Year	1750	1800	1850	1900	1950	1980	1990	2000	2009	2019
Population (millions)	791	980	1260	1650	2520	4454	5270	6060	6800	7744

1. Create two arrays, `y` and `Pop` for the year and population given in the table.
2. First, use polyfit to determine the best fitting exponential curve to the data.
3. Second, use polyfit to determine the best fitting cubic (aka, degree 3) polynomial to the data.
4. Make a nice plot of the data points and the best fits.
    - Plot the data points as green diamonds.
    - Plot the exponential best fit as a red line.
    - Plot the cubic polynomial best fit as a blue line.
    - Include a legend, where the labels include the data and the descriptions of the best fit lines.
    - Label your axes, turn the grid on, add a title, etc.
5. Estimate the world population using the cubic polynomial in your centenial year (when you are 100 years old). Using `plt.text` to add this information to the figure.
6. Save your figure as "YourLastName_PopulationGrowth.png" at dpi=200 and upload to CCLE.

In [21]:
```
Image("Aggarwal_PopulationGrowth.png", width=700, height=700)
```

Out[21]:



Problem 5 (Mid-Ocean Ridge Data; worth 3 points)

Mid-Ocean Ridges (https://en.wikipedia.org/wiki/Mid-ocean_ridge) form when hot material comes up from Earth's mantle producing new oceanic crust. The hot new crust floats higher on Earth's mantle creating a raised ridge. The material later gets rafted off to the side as newer material gets emplaced along the ridge. The older material cools off and floats lower on the mantle. Plate tectonic theory predicts that the depth to the cooling seafloor will increase with the square root of seafloor age, but it has been found that this behavior only holds for seafloor age less than about 80 million years. To show this, we will plot sea floor bathymetry (depth to the sea floor) vs sea floor age in mya (millions of years) data from Carlson & Johnson, Journal of Geophysical Research 1994, which you can find here (https://doi.org/10.1029/93JB02696).

1. Just as in HW4 and HW5, import the MOR_data.txt file, which shows depth of the ocean as a function of the seafloor for 3 different mid-ocean ridges on the seafloor: Central-Indian Ridge (CIR), Mid-Atlantic Ridge (MAR), East-Pacific Rise (EPR).
2. By default, `age` and `depth` are both arrays that contain strings. We would like to use `age` and `depth` for calculations, so the values in the arrays must be converted from strings to floats. I suggest using `np.asarray` to convert them to floating point arrays.

3. Now, use `np.polyfit` to calculate an age$^{1/2}$ fit for all the data with $t < 80$ **mya.** Hint: Boolean masking might be helpful here!

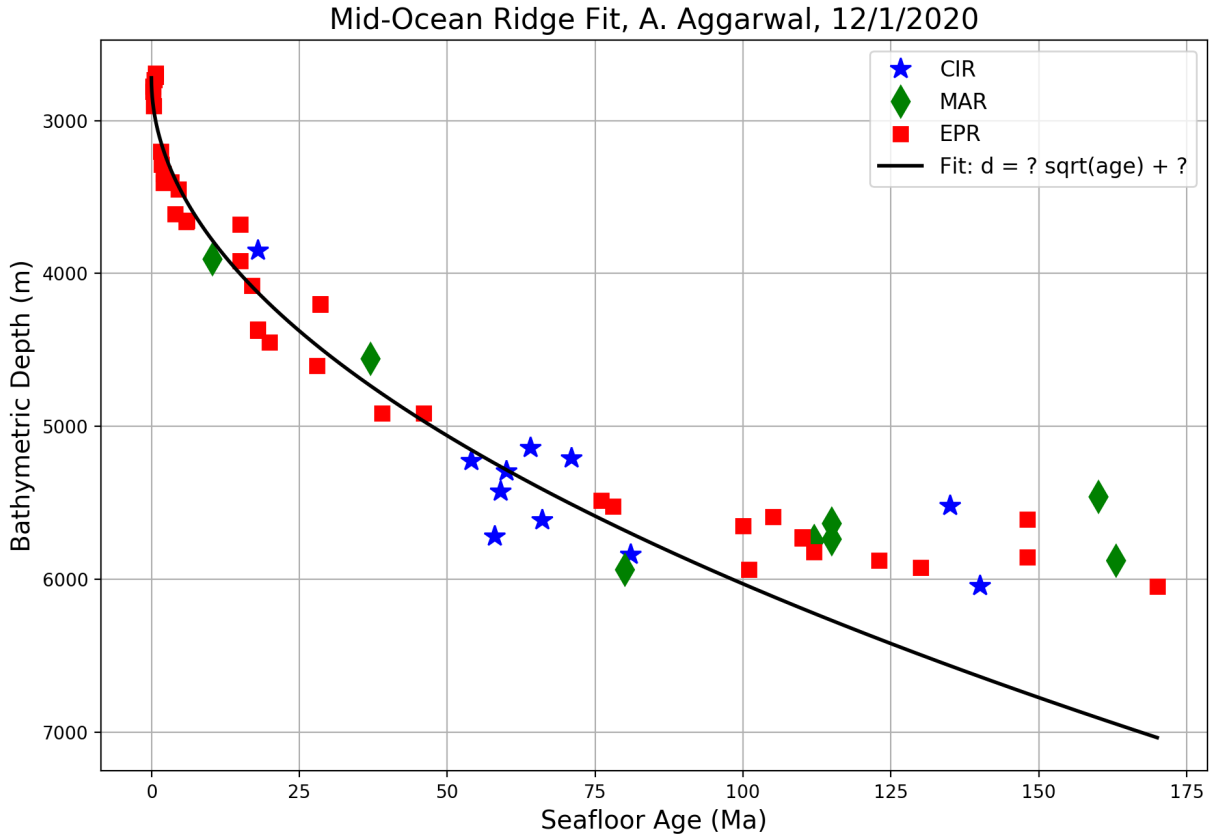
    - Define a new time variable such that you can carry out a linear fit to the equation **depth = m*t$^{1/2}$ + depth_0**
        - The linear `np.polyfit` will output two values. The first value is the slope, **m** (units: m/Ma^(1/2)), and the second value is the y-intercept, **depth_0** (units: m).
4. Make a nice 2 x 1 subplot of the data by following these steps:
    - In the first subplot panel, make a plot that shows the standard sea floor age (units of mya, millions of years) on the x-axis versus the depth to the sea floor (units of m) on the y-axis.
        - Use Boolean masking to plot each ridge's data (based on its name) using a different symbol and color. Use green diamonds for data points from the Mid-Atlantic Ridge, "MAR"; red squares for data points from the East Pacific Rise, "EPR"; and blue stars for data points from the Central Indian Ridge, "CIR."
        - Label your data sets with their corresponding ridge name.
        - Plot the best fit curve as a black line, and use string formatting to include the coeffcients **m** and **depth_0** (out to two digits after the decimal) in the equation for the best fit.
        - Reverse the y-axis, so that the greatest depths are at the bottom.
        - Add a legend to your figure in the upper right corner.
        - Turn the grid on.
    - In the second subplot panel, make a plot that shows depth (y) as a function of **age**$^{1/2}$ (x). It should be similary constructed (grid, legend, fit, reversed y-axis) to the first subplot, but with the square root of age plotted on the x-axis.
    - Why is the best fit linear in this $t^{1/2}$ -depth space? You can answer in a comment line.

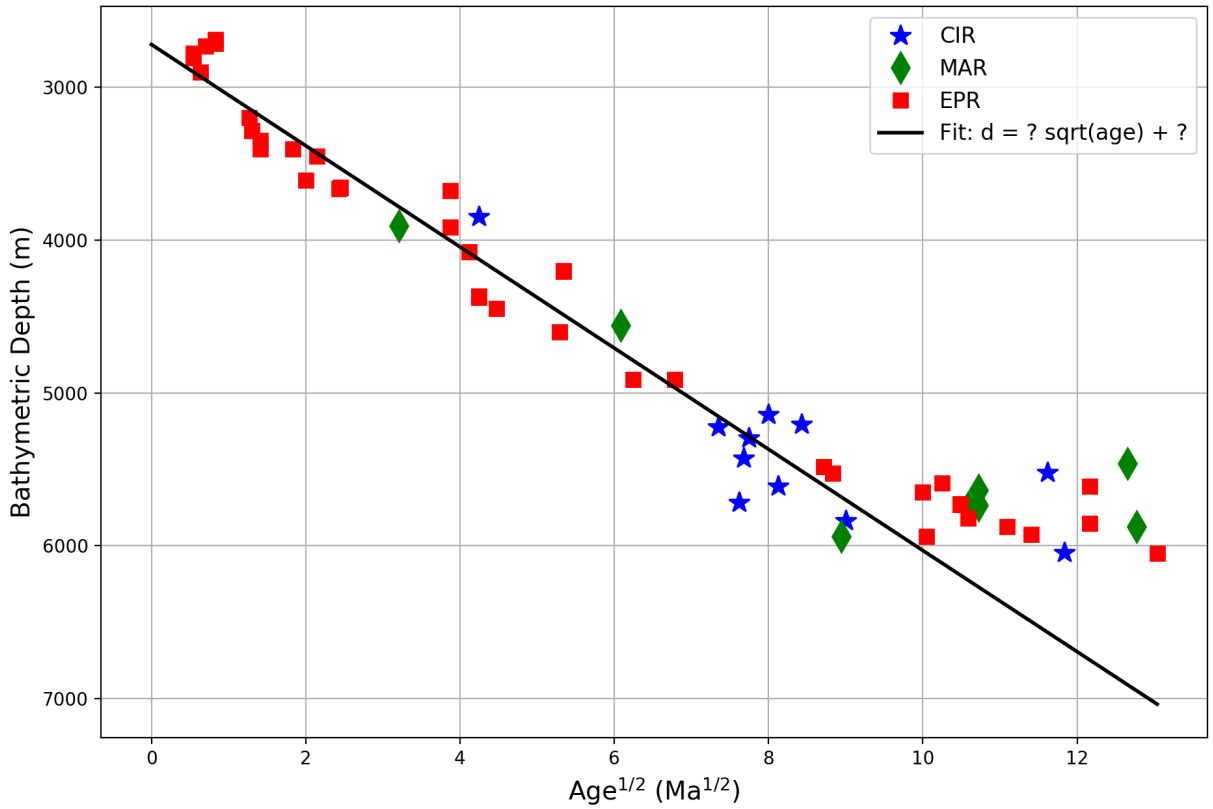
- Why does the data depart from the best fit line for seafloor that exceeds approximately 80 millions years in age? (Graded for completion)

1. Save your figure as "YourLastName_MORFit.png" at dpi=200 and upload to CCLE.

In [24]:
```
Image("Aggarwal_MORFit.png", width = 700, height = 700)
```

Out[24]:





Problem 2 (Magnetic saturation in liquid Gallium worth 4

Problem 6 (Magnetoconvection in Liquid Gallium; worth 4 points)

For this problem, you will work with data from a set of about 50 liquid metal convection experiments made in the pressence of a magnetic field. These experiments, made using liquid gallium, were carried out by Eric King in about 2010. Convection (https://en.wikipedia.org/wiki/Convection) occurs when a hot fluid rises and cold fluid sinks, like in a pot on the stove. Eric and Jon were using the magnetoconvection experiments to simulate convection in Earth's molten iron outer core (https://en.wikipedia.org/wiki/Earth%27s_outer_core).

In the data file, you will find three different data arrays. The first is the Rayleigh number, the second is Chandrasekhar number, and the third is Nusselt numbers.

The Rayleigh (**Ra**) number describes the strength of the buoyancy forces driving the convection. The Chandrasekhar (**Ch**) number describes the strength of magnetic forces which tend to damp the convection. The Nusselt (**Nu**) number estimates the strength of the resulting convective fluid motions through the tank of fluid. (When there is no convection, the Nusselt number value should be unity.)

Your job here is to make the plot below. The data is grouped into 6 groups of Chandrasekhar values (with approximately, but not perfectly, equal values of Ch for the members of each group). Within a clean for-loop, make a log-log plot of the Ch=0 data as black stars and then successively overplot the other Ch-groups as different color-filled circles. In addition, you will generate power-law fits between the Nu and Ra values for each Ch data set, overplotting the fits as thin solid lines. In the legend, the mean value of Ch and the best fit power law exponent (Greek alpha in my plot) are to both be reported.

1. The magnetconvection data is stored in a MATLAB file, "MCdata.mat". To load "MCdata.mat" into Python, you will first import the scipy (https://www.scipy.org/) I/O package, scipy.io. Then, you will use the command `sio.loadmat`, and set the keyword argument "squeeze_me" to True so Python correctly imports the data. With this method, the dataset is saved in Python as a dictionary. You can access the variables "Ch", "Ra", and "Nu" then by addressing the loaded data with those respective strings.

    - The code to load in the data is given below. Feel free to copy and paste it into your notebook. Let us know if you have further questions about this import. After doing this, you are ready to manipulate the data!

```
import scipy.io as sio
MCData = sio.loadmat("MCdata.mat", squeeze_me = True)
Ch = MCData["Ch"]
Ra = MCData["Ra"]
Nu = MCData["Nu"]
```

1. You will organize the data into six groups according to their "Ch" values. To do this, it is helpful to visualize how the Chandrasekhar data is distributed.

- Make a plot of Chandrasekhar on the y-axis and its data index on the x-axis.
  - Sort `Ch` when plotting, and use blue circles for the marker.
  - Use the command `plt.semilogy` (documentation here (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.semilogy.html)). A semilogy plot will change the y-axis to log scale, and leave the x-axis linear. The command is helpful here, because the Chandrasekhar data ranges over several orders of magnitude.
  - Turn the grid on, label your axes, etc.
  - Note, that the points where $Ch = 0$ are not plotted on this figure, because $log_{10}(0)$ is undefined.
  - How many "groups" of Ch values do you see on your plot? When including Ch = 0 data, how many groups are in the dataset in total? You can answer in a comment line.
  - Save you figure as "YourLastName_ChGroup.png" at dpi=200 and upload to CCLE.
- Based on this plot, we can define an array `Ch_ranges` of six numbers whos values are `ChGroup_UpperBounds = np.array([0, 2e4, 6e4, 1.5e5, 4e5, 2e6])` This array will be used in the next step to organize the data into six groups based on their respective Chandrasekhar values.

2. Now, create a single for-loop that iterates 6-times, one iteration for each group of Chandrasekhar values. Within the loop, it should

   - Isolate the appropriate "Ch" values of each "Ch" data set by using the array `Ch_ranges` to set the ranges of each group. So, for example, all data with Ch = 0 should be in the first group. The second group should contain all the data with 0 < Ch < 2e4. The third group should contain the data with 2e4 < Ch < 6e4... And so on. **HINT**. This can be done with Boolean addressing!
     - Calculate the mean "Ch" value of each group.
3. Make a loglog plot using `plt.loglog` (see documentation here (https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.loglog.html)) of the data with **Ra** on the x-axis, and **Nu** on the y-axis for each **Ch** group.

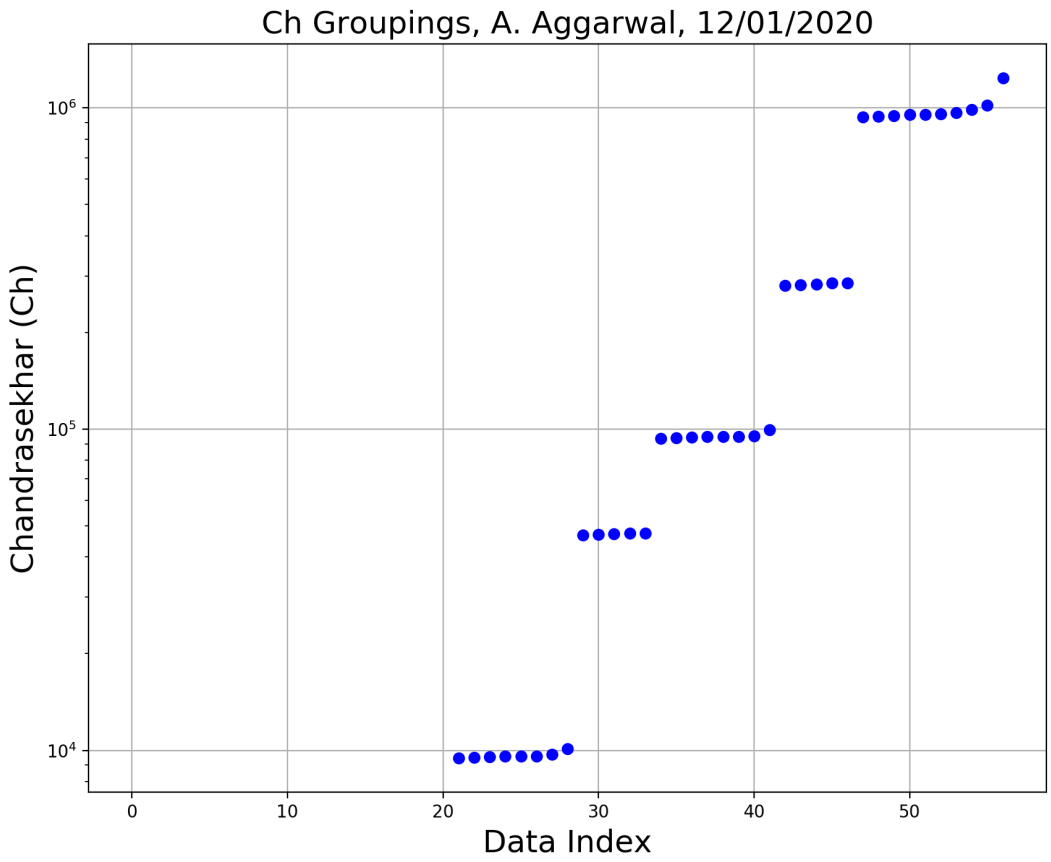
   - For the Ch=0 data, use black stars.
   - For all other data, use filled circles with a color of your choice. I used magenta, blue, green, red, and cyan.
   - Next, find the best power law fit for the **Nu-Ra** data in each **Ch** group.
     - You can do this by using `np.polyfit` . Notice here that **Ra** is your indepdendent (x) variable, and **Nu** is the dependent (y) variable.
   - Theory predicts that
   $$Nu = c\, Ra^{\alpha},$$
   with $\alpha = 1/4$ when $Ch = 0$ and then α should successively increase with increasing Ch. **So our goal here is to find how α varies with Ch to see if the data supports current theory.**
   - Plot each best fit power law as a thin line (set your linewidth to 0.50) atop the data, as shown.
   - Within your loop (aka, for each Ch-group), label each dataset with its mean Chandrasekhar value (in scientific notation to the first digit) and its best fit value of α. Check the α values you find against those on the plot below to see if you are on the right track.
   - Outside of your loop...

- Add a legend to the bottom right corner.
- Label your axes.
- Turn your grid on.
- Make sure your fontsizes are readable. I used size 18 font for everything.
- Title your plot "E. King: Magnetoconvection Data; Your First Initial. Your Last Name, Today's Date."

4. How did theory and data compare? Answer in a print statement. (Graded for completion.)
5. Save your figure as "YourLastName_NuRaCh.png" with dpi=200, and upload to CCLE.

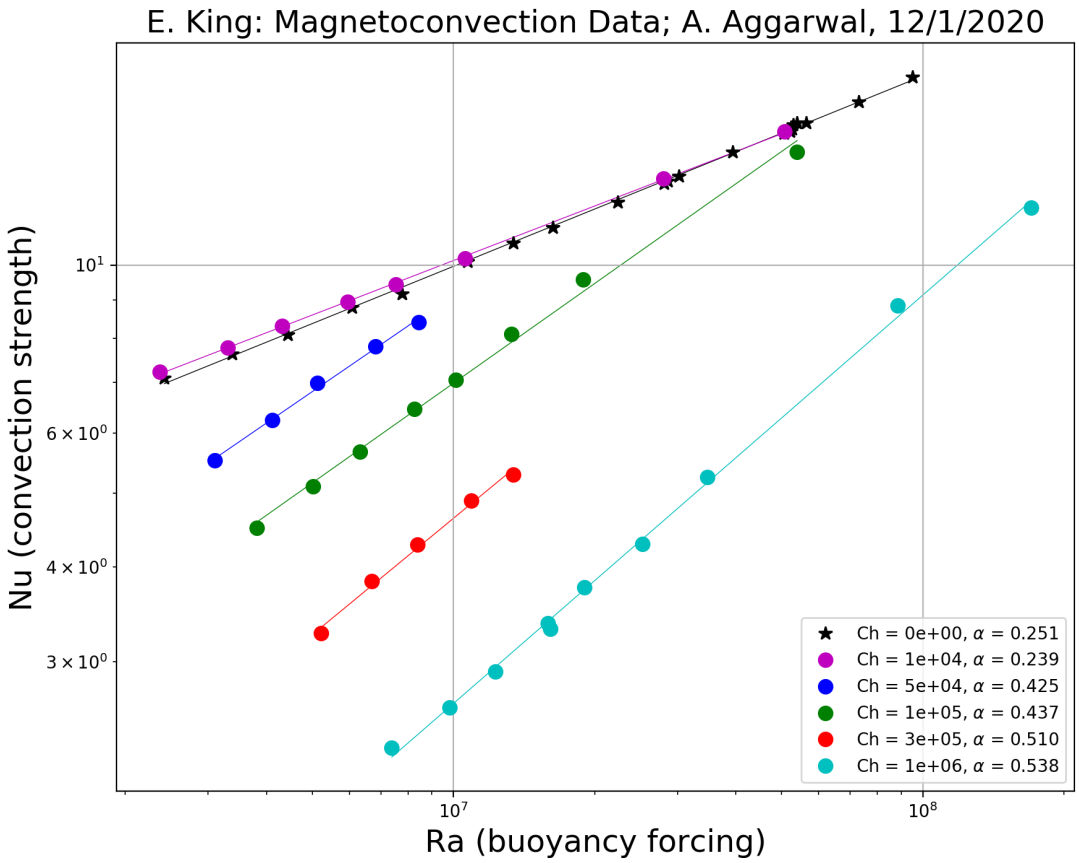
In [25]:
```
Image("Aggarwal_ChGroup.png", width = 800, height = 800)
```

Out[25]:



In [17]:
```
Image("Aggarwal_NuRaCh.png", width = 800, height = 800)
```

Out[17]:



We want your feedback

Congratulations! **You just finished your last assignmnment for M71 Fall 2020 (!).** In a markdown cell, please provide any feedback you may have for improving the questions in this assignment.

In [ ]:
```
```