### Enter your name

Your\_Name: "Insert text here

Show code

### Building a Star

This code will create a plot for a star (2-D circle) of uniform density (each shell has equal the same density) and plot how the mass and gravity change as a function of radius.



For your Python code, you must first import the relevant packages for your code.

Here we are importing or loading 3 packages that we will use with this code:

1-NumPy is a fundamental package of Python. It allows for a wide range of data types and data manipulation capapbilities. To use it you will

2- Matplolib is a Python 2D plotting library. We are only loading the Pyplot capabilities, which provide a collection of command that make

3- Math is a module that allow for certain math functions.

matplotlib work like MATLAB.

4- %matplotlib notebook allows for interactive plotting.

To execute a code in a cell, you can either select the cell and hit Shift+Enter or click on the run cell button above.

```
1 %matplotlib inline
```

2 import numpy as np 3 import mathlotlib.pyplot as plt 4 import math

Once you have succefully executed a cell, a number should appear in the square brackets next to that cell.

First we will define an array  $rad\_star$  (for radius of star) that will go from the center of the star (r=0) to the star's srurface (r=1). We will use the NumPy function arange which generates evenly spaced values within a given interval. We specify the increment to be 0.01. You need to specify the start and end points for the radius.

Keep in mind that Floats or non-integer numbers (i.e. with decimal places) should be written with the decimal point. For example, if our starting point is 1 and our ending point is 10, then the code should be np.arange[1.0,10.0].

3 rad\_star = np.arange(start,end,0.01)

### Mass

For this exercise you will need to determine the relation between Mass M and radius r, given that the "star" is of constant density, ho. Use the continuity equation

$$\frac{dM_r}{dr} = 4\pi\rho r^2$$

We have worked this out in class, but work this out again on a separate sheet of paper first and enter the expression in the cell below. Your expression will be a normalized mass in terms of the total stellar mass,  ${\cal M}_*.$ 

- To use the constant  $\pi$  in Python, you use math.pi. For example, the circumference of the star can be expressed like
  - In your expression for mass, the radius term r is defined by the variable  $rad\_star$ circumf = 2.0 \* math.pi \* rad\_star
- Remember to hit Shift+Enter to execute the cell.

1 mass =

### Gravity

Now that we have an expression for mass of the star in terms of radius, we will now do the same thing but for gravityg, where

$$l = \frac{G_{AM_T}}{m^2}$$

The gravity term will also be in normalized units, in terms of the surface gravity  $g_{surface}$ 

1 grav =

### Plotting

There are several ways to plot in Python. Here we will use some of the tools that are part of matplotifib. However, you can explore other ways to plot in this useful tutorial. Each line of code is explained below.

- Create a figure object, called fig
- fig = plt.figure()
- Create an "axis object" in the figure
  - ax1 = fig.add\_subplot(111)
- Creates a new set of axes (ax2) that shares the x-axis with ax1, but can have a separate y-axis.
  - ax2 = ax1.twinx()

- Tells python what to plot in the axes object, i.e. your (x,y) values. In this case ax1 will be for the mass. The last part specifies to plot a straight solid line ('-') in blue ('b').
  - ax1.plot(rad\_star,mass, 'b-')
- Plot the gravity in ax2 as a straight, solid line in red.
  - ax2.plot(rad\_star,grav, 'r-')
- When creating figures, you should always properly label your figure and include units when appropriate. ax1.set\_title("Stellar Interior of Star with constant density")
- To label your axes, set\_ylabel defines the y-axis labels and set\_xlabel does the same for the x-axis. ax1.set\_ylabel('Mass (M/M\_{\*})', color='blue')
  - ax2.set\_ylabel('Gravity (g/g\_{surface})', color='red')
- ax1.set\_xlabel('Radius (r/R\_{\*})')
- Each time we plot, we must show the figure plot.
  - plt.show()
- 1 fig = plt.figure()
  2 ax1 = fig.add subplot(111)
  3 ax2 = ax1.plot(rad\_star, mass, 'b-')
  5 ax2.plot(rad\_star, grav, 'r-')
- 7 axi.set\_title("Stellar Interior of Star with constant density") 8 axi.set\_jabel("Mass (\$WM\_{(1}\beta)", color="blue") 9 axi.set\_jabel("Gravity (\$g/k\_{(surfac)\beta)", color="red") a axi.set\_jabel("Radius (\$r/R\_{(1}\beta)"))
- 12 plt.show()
- ⇒ Now see if you can edit the code above and add density to your plot.

## Decreasing density Star

You will now adapt the code from the constant density case to create a plot for a star (2-D circle) of decreasing density and plot how the mass, gravity, density, pressure and potential energy change as a function of radius.

# Decreasing density

$$\frac{\rho}{\rho_o} = 1 - \frac{r}{R_\star}$$

star's srurface (r=1). We will use the Num $\mathcal{P}$ y function arange which generates evenly spaced values within a given interval. We specify the First we will define an array rad\_star\_dec (for radius of star with decreasing density) that will go from the center of the star (r=0) to the

increment to be 0.01. As above, you need to specify the start and end points for the radius

Keep in mind that Floats or non-integer numbers (i.e. with decimal places) should be written with the decimal point. For example, if our starting point is 1 and our ending point is 10, then the code should be np.arange[1.0,10.0].

- 1 start\_dec =
  2 end\_dec =
  3 rad\_star\_dec = np.arange(start,end,0.01)

Mass

For this exercise you will need to determine the relation between Mass M and radius r, given that the "star" is of constant density, ho. Use the continuity equation

$$\frac{dM_r}{dr} = 4\pi\rho r^2$$

We have worked this out in class, but work this out again on a separate sheet of paper first and enter the expression in the cell below. Your expression will be a normalized mass in terms of the total stellar mass,  ${\cal M}_*$ 

- To use the constant  $\pi$  in Python, you use math.pi. For example, the circumference of the star can be expressed like
  - circumf = 2.0 \* math.pi \* rad\_star
- In your expression for mass, the radius term r is defined by the variable  $\,$  rad\_star
  - Remember to hit Shift+Enter to execute the cell.
- 1 mass\_dec =
- Gravity

Now that we have an expression for mass of the star in terms of radius, we will now do the same thing but for gravityg, where

$$g = \frac{GM_r}{2}$$

The gravity term will also be in normalized units, in terms of the surface gravity  $g_{surface}$ 

### Continue to build your model.

Density

Express the normalized density  $ho/
ho_0$ .

Pressure

Now enter the expression for normalized pressure in terms of the central pressure  $P_{
m c}$ 

### Potential Energy

Enter the expression for the normalized potential energy  $\Omega_T/\Omega_{total}$ , where  $\Omega_T$  is the enclosed potential enegy. The potential energy of a small shell is

$$d\Omega_r = -\frac{GM_r}{dM_r}dM_r$$

1 Epot\_dec =

### Plotting

There are several ways to plot in Python. Here we will use some of the tools that are part of matplotlib. However, you can explore other ways to plot in this useful tutorial. Each line of code is explained below.

- Create a figure object, called fig
- fig = plt.figure()
- Create an "axis object" in the figure
- axes = fig.add\_subplot(111)
- Tells python what to plot in the axes object, i.e. your (x,y) values. The last part specifies to plot a line in blue ('b'). axes.plot(rad\_star,mass, 'b-')
- When creating figures, you should always properly label your figure and include units when appropriate. axes.set\_title("Stellar Interior of Star with decreasing density")
- - axes.set\_xlabel('', color='blue') axes.set\_ylabel('')
- . Since we are plotting more than two lines, and all the parameters are normalized, we will use a legend to label each line. You can specify where you want the legend to appear. Best lets Python place it where it thinks it is best. axes.legend(loc='best')
- Each time we plot, we must show the figure plot.

plt.show()

```
4 axes.plot(rad_star_dec, rho_dec, 'r', label='Density')
5 axes.plot(rad_star_dec, mass_dec, 'b', label='Mass')
6 axes.plot(rad_star_dec, grav_dec, 'g', label='Grav_ttv')
7 axes.plot(rad_star_Dec, press_dec, 'm', label='Pressure')
1 fig = plt.figure()
2 axes = fig.add_subplot(111)
```

9 axes.set\_title("")#<--Enter appropriate plot title in the quotations 10 axes.set\_Xlabel("')#<--Enter appropriate x-axis title in the quotations 11 axes.set\_Xlabel("')#<--Enter appropriate y-axis title in the quotations

13 axes.legend(loc='best') 14 plt.show()

# Adapt the plotting routine to include the potential energy.

You can use the following Python color chart to add more colors to your plots.

slategray lightsteelblue cornflowerblue royalblue ghostwhite lavender	midnightblue navy darkblue mediumblue blue	slateblue darkslateblue mediumslateblue mediumpurple rrebecapurple	blueviolet Indigo darkorchid darkviolet mediumorchid thistle plum violet	purple darkmagenta m magenta fuchsia orchid mediumvioletred deeppink hotpink lavenderblush
forestgreen limegreen darkgreen green g	seagreen mediumseagreen springgreen mintcream mediumspringgreen	aquamarine turquoise lightseagreen mediumturquoise	lightcyan lightcyan darksiategray darksiategrey teal c c	aqua darkturquoise cadetblue powderblue lightblue deepskyblue skyblue skyblue skyblue skyblue alightskyblue
linen bisque darkorange burlywood antiquewhite	navajowhite blanchedalmond papayawhip moccasin orange	oldiace floralwhite darkgoldenrod goldenrod	gold lemonchifon khaki palegoldenrod darkkhaki ivory beige iightyellow	lightgoldenrodyellow olive olive yellow olivedrab yellowgreen darkolivegreen greenyellow chartreuse
black k dimgrey dimgray grey gray	darkgray darkgrey silver lightgrey lightgray	e e		mistyrose salmon tomato darksalmon coral orangered silightsalmon sienna seashell
		7 S S S G E :	= = 0 = 5 5 5	0 % % == 0 0 0 0 0 %