Lab 1: Introduction to MATLAB as a Crazy Expensive Calculator

Created: 18 July 2018

By: Samuel Bechara, PhD

Figure 1.1: MATLAB is WAY cooler than this bad boy! At least it will be...eventually.

Table of Contents

Lab 1: Introduction to MATLAB as a Crazy Expensive Calculator 1
- Disclaimer: First Lab Manual Only 2
- Introduction 2
- Key Concepts and Takeaways 3

MATLAB Basics 3
- The Banner 4
- The "Current Folder" Browser Window 5
- The "Command Window" 6
- The "Workspace" Window 7
- The "Editor" Window 7

Rice is great if you're really hungry and want to eat two thousand of something 8

The Great Excel-Matlab Showdown 9

I thought Dyson made vacuums?!?.....9

Disclaimer: First Lab Manual Only

I spent a lot of time designing these labs manuals and writing them. Hopefully you find that they help guide you as you explore Mechanical Engineering and MATLAB. I expect you to spend a lot of time working on the labs and reading through the lab manuals. Engineering isn't easy! It takes lots of work to become a professional!

Because of this expectation, the LAs and I will get upset if you ask a question on something that is directly addressed in the lab manual. **It is your responsibility to read through the lab manuals in their entirety before asking questions on the labs or the material covered in the labs. This doesn't mean that you can't ask questions if you get stuck.** Now that we have that out of the way...

Introduction

In class, you were given a brief introduction to MATLAB in lecture. That is ok but it isn't the best way to actually learn how to use MATLAB. Now it is your opportunity to dive head first into MATLAB and actually learn. But the learning part is up to you! Make sure you take your time with this lab and answer the questions that it asks you. It is ***CRITICAL*** that you actually play around with MATLAB during the lab, try different things, and generally get a feel for the software so that you can be successful in future labs in addition to the delieverable. Don't forget you will have a lab quiz on this lab due next week! All of the questions asked in this lab would be fair game on a quiz.

I think it helps to think of working on these labs like a workout and the labs are like a gym. Watching me show you MATLAB in class is equivalent to watching a workout video on YouTube. It isn't a TOTAL waste of time, you need to learn where the weights are in the gym and what moves to do to properly excersize. But you have to go to the gym to get a workout. When you do go to the gym, it is ok to get a spot (*ask LA's for help*) but it isn't ok to just watch someone else workout (*copy what someone else is doing or something you found on the internet*). It is ok to work together, but YOU need to bang away on the keyboard and type everything out yourself just like if you want your biceps to be strong YOU need to do the curls.

Moral of the story?

- **DO NOT COPY AND PASTE ANYTHING!**
- **WORK HARD TO DISCOVER THINGS YOURSELF!**
- **DON'T BE AFRAID TO ASK QUESTIONS BUT TRY TO FIGURE IT OUT ON YOUR OWN FIRST!**

Also, enjoy this sweet 80s workout picture. Man, the 80s were a weird time...

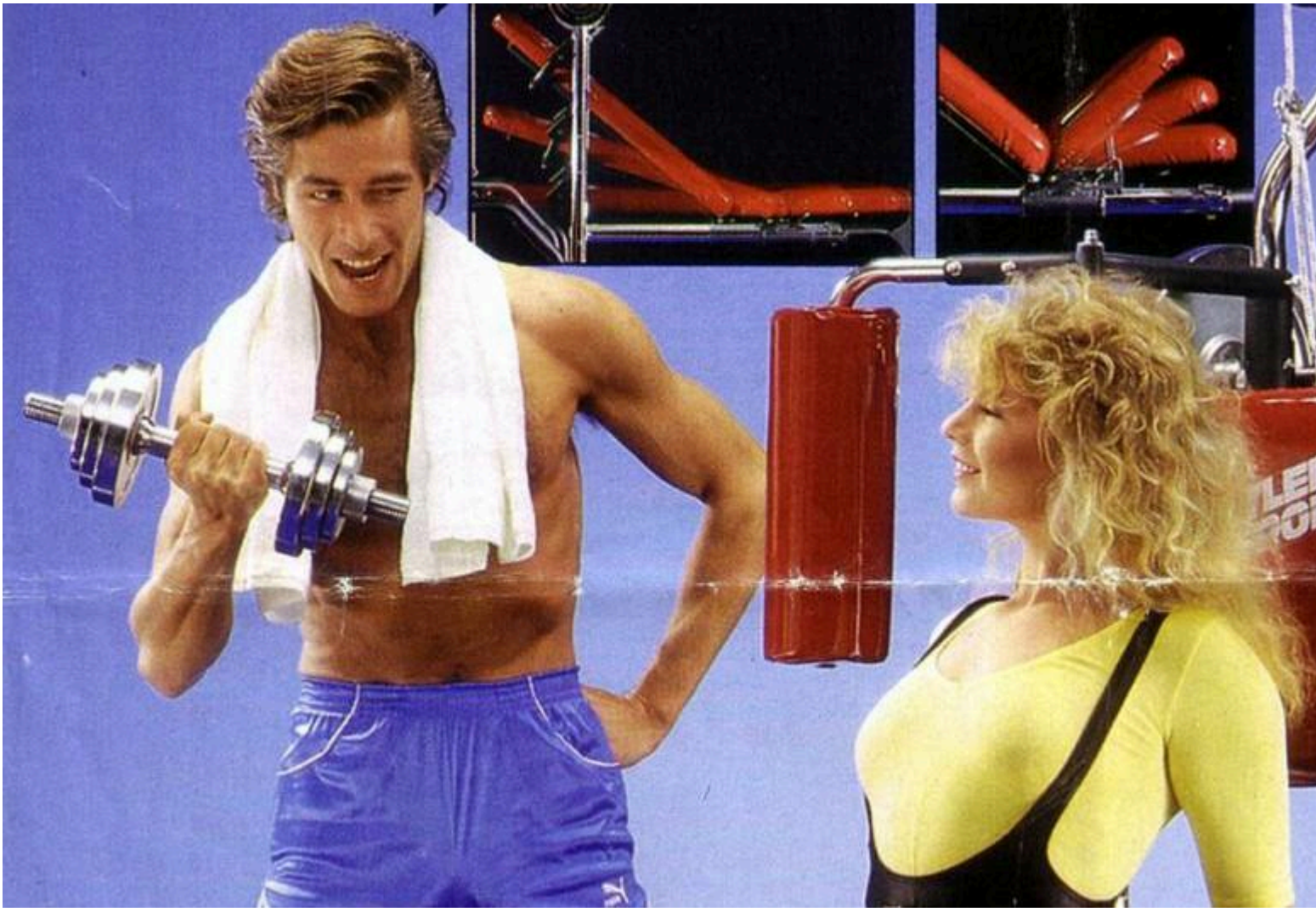


Figure 1.2: Sweet 80's Workout Picture for your Enjoyment

Key Concepts and Takeaways

Every lab will have a list of key concepts and takeaways listed as bullet points. These bullet points effectively act as your "study guide". They let you know what we except you to have learned from the lab each week. They will also inform the basis of your lab quizzes.

- Understand the importance and use of the MATLAB: editor, command window, workspace variables, and current folder.
- MATLAB commands: `clear(), clc()`
- MATLAB Variables
- MATLAB Math Operations (addition, subtraction, etc)
- The `ans` automatic variable
- Scripting basics

MATLAB Basics

For this section it is best to follow along with this guide. You should open up MATLAB to click on stuff and explore on your own, but this should give you a good foundation for navigating MATLAB. Go ahead and open MATLAB, chances are it is going to look something like this:

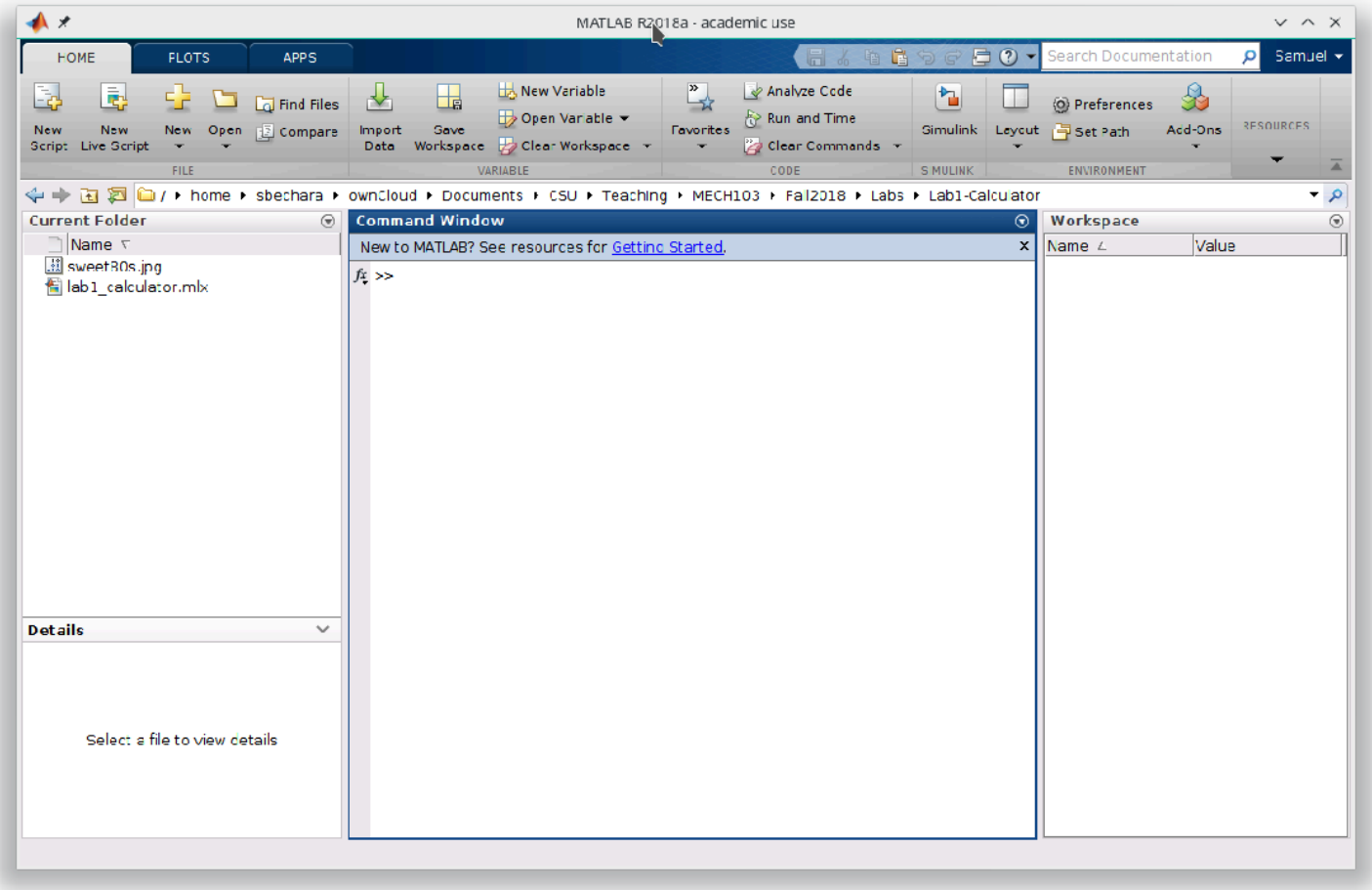


Figure 1.3: Overview of MATLAB

Woah, there is a lot of stuff going on! By the end of the semester, we will have clicked almost every button and figured out what all this stuff does. Try clicking the "New Script" button in the top left corner (Note: Do not click the "New Live Script" button, we will save that feature to investigate on a different day). For now, there are a couple of important things to notice.

- The banner with icons and text that sits at the top
- The "Current Folder" browser window and address bar
- The "Command Window"
- The "Workspace" window
- The "Editor" window

Lets take a deeper look at each of these in more detail.

The Banner

The banner is a tabbed menu sitting at the top of the screen consisting of icons and text that help us do a TON of stuff. The banner will update with new tabs and options depending on what you are doing. If you

haven't already, try clicking the "New Script" button and watch what happens to the banner. It shows you different options now to try and help you with your new script.

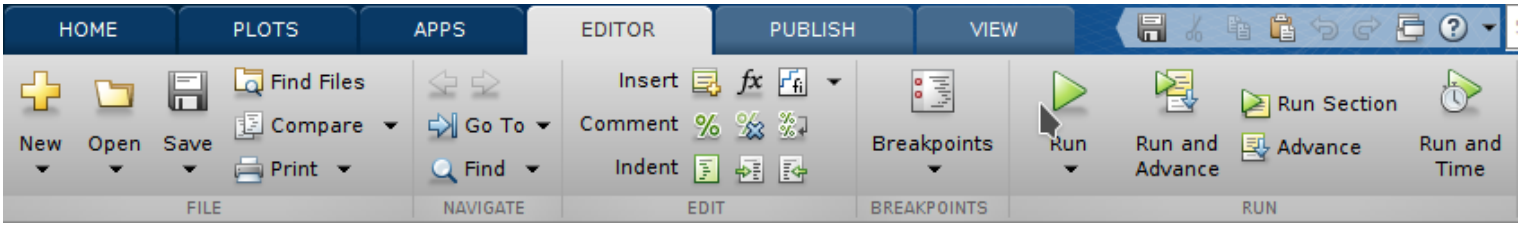


Figure 1.4: MATLAB Banner showing Editor Options

Keep your new script open for now. *If you don't have a script open, and you try to follow the rest of this guide, your gonna have a bad time*. Before you continue to the next section do the following and take notes so you don't forget:

- Click through the banner tabs and look at the different options for the script (specifically the tabs: "Editor" and "Publish"). We are going to be using those tabs a lot throughout the semester so it is a good idea to get a feel for the options and buttons. Pay special attention to the "Editor" tab and the options presented to you there. That is where we will be spending most of our time this semester.
- Notice that if you "hover" your mouse over a banner icon or text it gives you a little description about what the button does. This can be useful to help remind you what things do. Using this technique, what does the "Breakpoints" button under the "Editor" tab do? This feature will be **critical** as we continue the semester.

The "Current Folder" Browser Window

The folder that you are currently located inside is *super important* when it comes to using MATLAB. So this is a great time to figure out how you will organize and save your files. If you dump everything into one folder, technically that will work, but it will make navigating your files a *nightmare*. Trust me. I have seen how some students organization structure (or lack therof) can lead to problems.

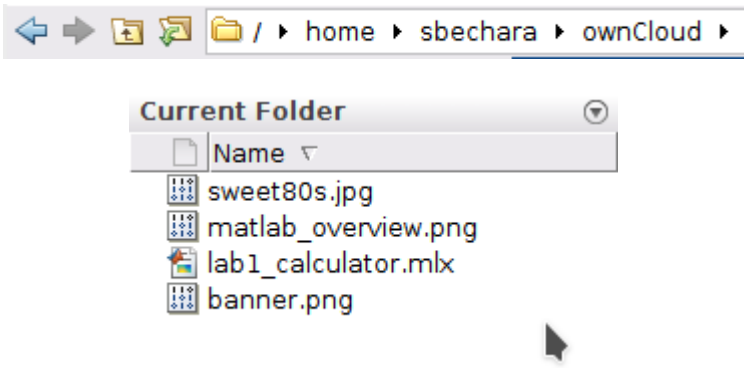


Figure 1.5 and 1.6: MATLAB address bar (above) MATLAB Current Folder Browser Window (below)

In any case, MATLAB always defaults saving, looking for stuff, and more to the **current working directory. The current working directory** is simply the file-system folder that MATLAB is currently "inside". The contents of that folder are displayed in the "Current Folder" browser window and the path to that folder is located in the address bar directly above the broswer.

- Try navigating your file system using the browser window and the address bar.
- Type some text in your open editor window and save the file somewhere. Close it, make sure you can find it using the "Current Folder" browser and the address bar.

- Use the address bar navigation options (arrows, up folder, browse for folder, and path). Remember that hovering over a tool will give you a tip as to what it does!
- Notice that you can right click and edit files from the "Current Folder" browser. You can change file names, etc.

The "Command Window"

The "Command Window" is a place where you can issue single commands to MATLAB. Go ahead and try and use MATLAB as a calculator. Give it mathematical operations to perform.

If you can't be bothered to come up with your own calculations, try:

```
5+3
```

```
ans = 8
```

```
6*8
```

```
ans = 48
```

```
5^2
```

```
ans = 25
```

Note: it is your responsibility to figure out what the correct MATLAB Operators are (+,-,etc. Don't worry about the relational operators just yet). Most are intuitive, others are not. A comprehensive list can be found here: MATLAB Operators. **We will discuss the difference between Element-wise and Matrix operations at a future date.**

After playing around for a little bit there are few important things to notice.

- A new *variable* called `ans` was added to the "Workspace" window on the right. Notice how it is overwritten with a new value everytime you type in a new math operation to perform.
- Try adding a semicolon `;` to the end of a mathematical operation. What happens? Does the `ans` variable get updated (check workspace window)? What does the semicolon do?
- After you have tried several different math operations, try typing `clc` into the command window. What does it do? Did the "Workspace" variables change?
- Don't worry you didn't lose your history. *Try hitting the up arrow key while in the command window*. What does it do? To select multiple lines, hold the shift key then type arrow keys in a direction. You can hit enter to re-run the command selected (or multiple commands if you held shift).

A word on variables. Instead of typing in numbers it is possible (and will become necessary for us in the near future) to perform calculations using variables. Variables have strict naming conventions in MATLAB (see here for the rules: MATLAB Variable Rules) and can store numbers, text, arrays, matrices, and more! For now, lets try some calculations with variables.

Lets say we want to calcuate the force of a spring using Hooke's Law. Try issuing the following commands to the MATLAB command window:

```
x = 4.213;
k = 2.112;
F = k*x
```

There are a few things to notice.

- The variables make the code WAY easier to read. It is clear that in the third line, we are calculating force using Hooke's Law.
- The variables will save us time if we want to perform other calculations with the same k (spring constant) in the future. Instead of typing 2.112 everytime for a different displacement, we can simply reuse the variable.
- Notice what changes in the workspace. Try overwriting the variable x by assigning it a new value in the command window. You can then re-calculate F with your new x.

The "Workspace" Window

We have already talked about the workspace a little. It is the place where MATLAB indicates to the user what variables are currently loaded and available for the user to have access to. We will learn there are some exceptions to this in the future (i.e. variable scope). For now, unless a variable is shown in the workspace, we will assume MATLAB doesn't know about it.

For example, we have already entered in some commands at this point and you should have several variables loaded in the workspace. We have already seen how we can issue commands to the command window to use these variables. If you ran my Hooke's Law example you should have variables `x`, `k`, and `F` already loaded (at this point you should be comfortable referencing variables in the command window). Now try the following command in the command window:

```
clear
```

- What happens to your variables?
- Try and use the variable F in a math operation? What does MATLAB tell you?

The "Editor" Window

Up to this point, we have been issuing MATLAB commands one at a time via the command window. For example, you should be comfortable with calculating different mathematical operations using MATLAB and the command window. So MATLAB is essentially acting as an obscenely expensive calculator, don't worry, things get WAY cooler. Baby steps.

Well it is fine to us MATLAB as a calculator, but it isn't really the best way of making user friendly programs. To do that, we need to use the editor to create *script files*. Script files are just like they sound, they are a script for MATLAB to follow. Script files contain MATLAB commands on each line. When MATLAB runs a script file, it simply starts at the top, and executes the commands on each line in a sequential order. **THAT IS ALL THERE IS TO A SCRIPT FILE**! All MATLAB script files end in a .m extension but there is nothing special about them. Technically you can make .m files in *any* text editor or word processor and MATLAB will read them.

You should have an editor open already.

- Try and recreate the Hooke's Law example, make it a script file, and save it as hookes_law.m. Notice what changes in the current folder window.
- Include the `clear` command at the top of your script. This is a good idea for all your scripts so it is good to get in the habit now. The reason it is helpful is because it forces you to ensure that all of the variables you need to run your script are included in the script itself.

- Find the "Run" button in the banner and run your script! Play around with omitting semicolons and see what happens with the command window output.

Rice is great if you're really hungry and want to eat two thousand of something

The formula for exponential growth of a variable x at the growth rate r, as time t goes on at integer times (i.e. t = 0, 1, 2, 3 ...) is given by:

$$x_t = x_0(1 + r)^t$$

Where x_0 is the value of x at $t = 0$.

An old legend states that the inventor of the game of chess was summoned by the king. The king loved chess and and asked the inventor what he would like as a reward. The inventor surprised the king by asking for one grain of rice on the first square, two grains on the second, four grains on the third, eight grains of rice on the fourth, etc. The king was shocked by such a humble request and readily agreed! A few grains of rice was a small price for such a great invention... or was it?

Figure 1.7: Love me some Basmati

For our purposes, lets pretend that the chessboard only has 32 squares (they really have 64, and we will get to that in a minute). Write a Matlab script that computes the number of grains of rice on the 32nd square.

Make sure your script meets the following requirements:

- Write your name and the date in comments at the top of the file.
- The variable `x_t` is used to store the value of rice grains on the number t square.
- the variable `x_0` is used to store the initial value for grains of rice
- The variable `r` contains the correct number the cooresponds to the growth rate described by the inventor. **THINK!**
- The variable `t` is used to store the chess square number
- Remember that Matlab, like most math programming languages (Fortran, Mathematica, Wolfram, etc.) starts counting from 1. This is in contrast to other common languages like C, Java, Python, and so on.
- Test your script with values that you should know. How many grains of rice should be on the first square, when t=1? How about on the second? The third?
- Calculate how many grains of rice need to be placed on the 32nd square (and store in `x_t` variable) using the formula outlined above.
- See the power of the script file! Change your `t` to 64 and run again! That is why script files are convienent.
- Turn in your script file with `t=64`

- Is that a lot of rice? Did the king make a mistake?

Turn in your script on Matlab Grader with t=64.

The Great Excel-Matlab Showdown

Get your tickets now to the great showdown featuring the venerable Excel and the RAM hungry Matlab! Recall from Lab 3 we used Excel to calculate the specific gravity of three different balls. Guess what? We're going to repeat that in Matlab and now you get to decide which software you prefer! Hopefully you don't hate one or the other (or both), because you're going to use them a lot! I'll give you the circumference this time.

Try not to write the equations 3 times. In fact, be lazy and don't even calculate all three at the same time. Make your script so that you only need to change the circumference and mass variables to see what the specific gravity is.

Density of water: 1g/cm^3

Softball: circumference=12in, mass=190g

Rubber ball: circumference=8in, mass=160g

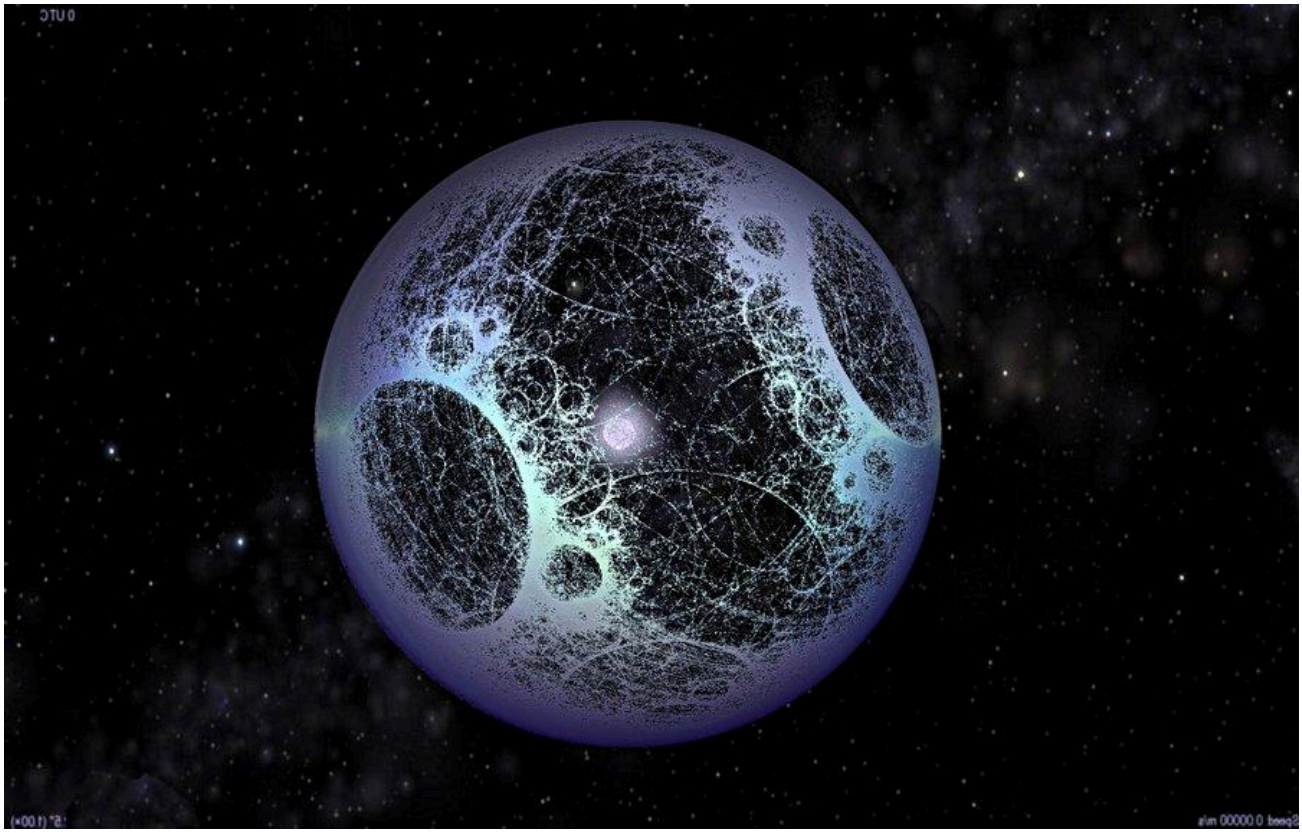
Tennis ball: circumference=8in, mass=60g

Write a MATLAB **script** and submit it online to Matlab Grader that meets the following requirements:

- Fill out the your name and date in the comments at the top of the file.
- Compute the specific gravity of the rubber ball.
- The variable **circumference** is used to store the circumference of the rubber ball in inches
- The variable **radius** stores the radius of the rubber ball in cm
- The variable **volume** stores the volume of the rubber ball in cm^3
- The variable **mass** stores the mass of the rubber ball in grams
- The variable **density** stores the density of the rubber ball in g/cm^3
- The variable **specificGravity** stores the specific gravity of the rubber ball

We will learn later on how to better reuse code to compute all of the specific gravities at once. Submit your script on Matlab Grader.

I thought Dyson made vacuums?!?



The future is now! Humanity has completed the most impressive engineering feat ever dreamed of, constructing a full Dyson sphere around the sun (and consequently turned the Earth into a giant snowball... whoops). Now that we've built the Dyson sphere, we need to figure out how much energy we are getting from the sun. We can approximate the radiative heat transfer with the equation:

$$\dot{Q}_e = \epsilon \sigma A_{sun}(T_{sun}^4 - T_{dyson}^4)$$

Where:

$\dot{Q}_e$ is the net rate of radiant exchange,

$A_{sun} = 6.09E{+}12 \ \mathrm{km}^2$ is the surface area of the sun,

$T_{sun} = 5,778 \ \mathrm{K}$is the surface temperature of the sun

$T_{dyson} = 263.15 \ \mathrm{K}$ is the surface temperature of the Dyson sphere

$\epsilon = 0.95$ is a measure of the efficiency of the Sun's radiation, we are treating it as an almost-black-body here

$\sigma = 5.67E{-}8 \ \ W/(m^2 - K^4)$is the Stefan-Boltzmann constant

Over the course of one day, how much energy has our Dyson sphere collected? Fort Collins charges an average of 11 cents/kWh. At that rate, how many seconds does it take for our Dyson sphere to make one yottaDollars for us? We're going to have a lotta yottaDolla!

Write a MATLAB **script** and submit it online to Matlab Grader that meets the following requirements:

- Fill out the your name and date in the comments at the top of the file.
- Compute the amount of energy collected in one day by the Dyson sphere.

- Compute how many seconds it takes to earn one yottaDolla. Look up what the SI prefix yotta means!
- The variable **netEnergyTransfer** stores the amount of power gathered by the Dyson sphere, measured in Watts.
- The variable **energyPerDay** stores the amount of energy collected by the Dyson sphere in one day, measured in Joules
- The variable **secondsToYottaDolla** stores the number of seconds it takes to earn one yottaDollar.