MATH2070: LAB #1: Preliminaries

Introduction	Exercise 1
Grading	Exercise 2
Starting up Matlab	Exercise 3
Using a browser to download files	
Getting help	
Quitting	
Lab summaries	
More on Matlab	

1 Introduction

You will find instructions for each lab, including this one, on the web, starting from my home page: `http://www.math.pitt.edu/~sussmanm` . I do not supply copies on paper. Many students find it helpful to print out copies of the lab instructions before the lab session, although it is not necessary. During the lab session, it is convenient to use the online version because you can "copy-and-paste" instructions from the web page directly into Matlab. If you prefer, you will find a version of this lab in Adobe `pdf` format here.

This lab will occupy today's lab session only. This session will introduce the mechanics of using Matlab, particularly on the Linux computers here in the lab. There is some reading to be completed before the second session and you can do that here in the lab or at any other computer with web access. The second lab, two sessions long, will present exercises in Matlab use.

The discussion that follows assumes that you are basically familiar with browsing the Web. The next few sections will give a brief introduction to Matlab and explain how to use it and those aspects of the environment that will be important to doing the labs.

2 Grading

The labs roughly follow the material presented in lecture, but are independent of the homework and other exercises presented in lecture. Lab grades count as 30% of your course grade.

Attendance is not required, but help is most readily available during the lab sessions.

You are encouraged to work together with other students, but you must provide your own diary and summary files (explained further below).

Each lab will be given a grade of A+, A, B, C, D or 0. These grades correspond with percentage grades of 100, 95, 85, 75, 65 and 0. At the end of the semester, your grades will be averaged and then integrated with your grade in lecture. The grading criteria are:

Grade	Value	description
A+	100	All exercises were completed correctly.
A	95	All exercises were attempted and are substantially correct.
B	85	All exercises were attempted but there are some serious errors.
C	75	Substantial portions of some exercises were omitted.
D	65	Little or nothing correct in the submission.
Zero	0	Lab was not submitted.

Some of the labs include extra credit exercises. The percentage values of these extra credit exercises are stated with the exercises themselves. At the end of the semester, the extra credit percentages will be added to the grade percentages and the average computed from the sum, except that averages will not exceed 100.

Unused extra credit on one lab will be applied toward *later* labs. Unsubmitted labs will not be eligible for extra credit.

Each lab is due before 11:59 PM the day the subsequent lab begins. Labs submitted after the day the subsequent lab begins will have 1% deducted from the percentage grade for that lab.

If you are unable to complete a lab within a week of its due date, you may take additional time to complete the lab, *provided you attend each lab session until you have completed and submitted the work.* In that case, 2% will be deducted from that lab's grade for the first week it is late, 3% for the second week, 4% for the third week, *etc. If you do not attend each lab session until you submit your work, you will be given a grade of zero for that lab.* Attending lab sessions will offer the opportunity to resolve any questions you might have.

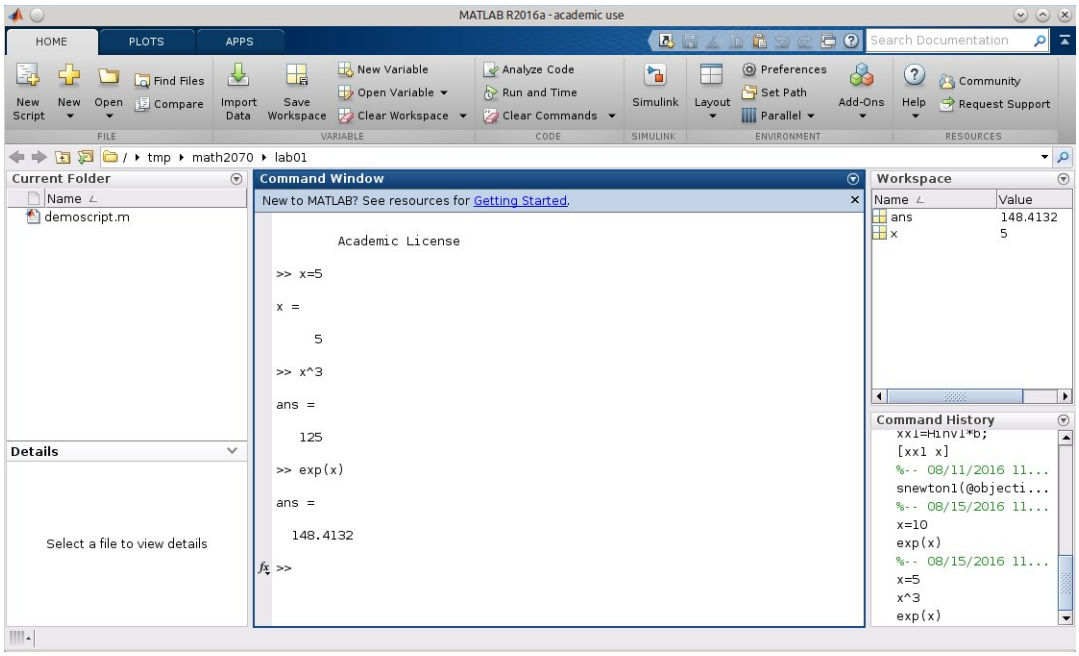
The final due date for labs 1 through 8 is the last day of classes for the semester, and the final due date for labs 9 and 10 will be announced near the end of the semester. Labs that are not submitted on or before their final due date will be given a grade of zero, except under special circumstances.

3 Starting Up Matlab

Matlab is available on the computers in Posvar 1200A, in other computer labs on campus as well as on the Linux computers in the labs in Thackeray hall, such as the seventh floor computer lab opposite the lounge. You can download it for your personal computer from the university software center, and I recommend you do so.

In this section you will see how to start up of Matlab using the windowing interface. These instructions are the same for Unix, Mac and MS-Windows versions of Matlab. I will also give the command-line equivalents of many of the commands. These command-line equivalents are valid for all versions of Matlab as well as for the Matlab clone named Octave. Generally speaking, anything you can do using a menu can also be done with command-line equivalents. You would use the command-line equivalents when writing scripts and the menus when working interactively.

1. To start up Matlab on the computers in Posvar 1200A, go to the start menu, type `matlab` in the search box and then click on it. A "splash" screen will open and then close by itself, followed by the Matlab desktop window. This window will look something like the following:



The Matlab desktop contains menus for many activities along the top, but we will be using very few of these options. Below the menus is a line containing icons representing "Forward", "Back", "Up to parent folder", and "Open folder", plus a line describing the location of the current folder.

By default, the region below the location of the current folder contains several windowpanes. The pane on the left lists files in the current directory and the contents of one of those files. The pane on the right is divided into a "Workspace" pane containing a list of all variables currently known to Matlab and a "Command History" pane containing copies of your recent commands. The center windowpane is the "Command" windowpane and you will be typing Matlab commands in it. On the left side of this command windowpane there will be a prompt of the form `>>`. Your typed commands go next to this prompt.

2. When working on the computers in Posvar 1200A, I strongly suggest you use a USB drive ("Thumb drive", "Jump drive", "Flash drive", "Memory stick") that you carry with you. Plug it into one of the computer's USB ports. You can use local storage, but it will be wiped out as soon as you log out, so you need to remember to copy your work to your Box account or some other permanent location. When removing your USB drive from the computer, it is best to find the small "USB drive" icon at the bottom of the screen and choose "Safely Remove" to be sure all information has been copied to your drive. (This step is particularly important when using Linux computers!)

3. It is a good idea to organize your work into folders (or directories—they are the same thing). Use the line containing the current folder to navigate to your USB drive (often `e:`). Inside that drive, I suggest that you use a directory named `math2070` for all the work in this lab, and subdirectories `lab01`, `lab02`, ..., `lab10` for each of the labs. You will need to create these directories before you can use them.

   If you wish to use the command line to create and switch directories, after locating your USB drive letter, you can create a directory named `math2070` with the command

```
mkdir math2070
```

and you can make it the current directory with the command

```
cd math2070
```

You can get a listing of the files in the current directory with either the command

```
dir
```

or the command

```
ls
```

Before going on, create the directory `math2070`, with subdirectories `lab01`, `lab02`, ..., `lab10`.

4. The "Command History" windowpane is convenient for recalling what commands you have used recently. In addition, Matlab provides the capability to keep a record of *both* the commands *and* their output. The command to do this is

   ```
   diary diary.txt
   ```

   The name `diary.txt` is actually the name of the file that is created and you can use any name you like. It is a text file, and you should use `.txt` to name it. Before going on, type `diary diary.txt` in the "Command" windowpane. Terminate the command by hitting the "Enter" key. In labs that take up several sessions, using the `diary diary.txt` command in the second session will append to the previous work and will not overwrite it.

   You should also type a comment line that will help you identify your work later. It should start with the comment character, a percent sign `%`, and include the lab number and date. This line will serve as an identifier when you look at the file.

   Note: The diary file contains a complete record of all the commands you enter in Matlab and the responses. You are to retain this *complete* record to send to me for grading. Please do not turn the diary file on and off or edit the diary file to hide your errors. There are times when you write an explanation that is not, in my opinion, correct and I will need the complete diary to establish what really did happen.

5. You can double-click a command in the "Command History" windowpane and it will be executed again. You can also use the mouse to drag a command into the "Command" windowpane and then change it using the arrow keys.

6. You can also recover previous commands by using the up-arrow key and then you can change them using the right and left arrow keys.

7. As will be described below, you will be sending your files to me by email. Since there are sometimes a large number of these files, it is inconvenient to attach them one by one, so I suggest you create a "zip" file. Before you create the "zip" file, exit your diary with the command `diary off`. If you do not, your diary will be incomplete in the "zip" file. You can create the "zip" file from the Matlab command line with the command

   ```
   !zip labfiles.zip *.m *.jpg *.txt
   ```

where "`labfiles.zip`" is the name of the file and can be chosen as you wish. (The exclamation point is necessary to tell Matlab that the command is a "system" command and not a Matlab command.) You then only need to attach this one file to your email. You should do this only once, when you have completed all your work and after you have closed your diary file with the command (`diary off`), or after you have closed Matlab down and then opened it again.

8. After you have completed a lab, exit from Matlab with the command `quit` or use the "X" button along the top to close the window.

4 Using a browser to download files

Some of the labs require that you download files from the web in order to use them. The following exercise illustrates how to download files. The file you will download is a very simple script file.

Exercise 1: If you are not using the online version of this lab, please start it up by starting the browser and finding the online version of this page, beginning from my home page,
`http://www.math.pitt.edu/~sussmanm` . clicking on "Math 2070", scrolling down to the list of labs and clicking on Lab 1.

Right-mouse click on the file demoscript.m to bring up a menu. Choose "Save link as" and a file save box will pop up. Navigate to the directory you made: `math2070/lab01`, and save the file with the name `demoscript.m`. You must use the `.m` extension to tell Matlab that the file contains Matlab commands. Return to the Matlab command window. The file should be visible to Matlab, a fact that you can confirm with the `dir` command or by its presence in the "Current Directory" windowpane. If it is not visible to Matlab, you have probably stored it in a different directory than Matlab is looking at. You can find the directory that Matlab is using with the command *(*pwd) ("print working directory") or in the line immediately above the "Command Window" and "Current Folder" windowpanes in the Matlab window.

Edit the file by typing the command

```
edit demoscript.m
```

or by double-clicking on the file in the "Current Directory" windowpane, or using the "Open" menu. An edit window will show up. Read through the file: the comments make it self-explanitory.

You can tell Matlab to execute (that is, cause the statements in the file to be executed) the file by typing its name, without the "`.m`", on the command line. (There is another method for executing a file that is not so appropriate for this course: you can choose "Run" from the menu on the edit window.)

Do not be confused by the final few statements in the file. They refer to the following exercise.

The following two exercises illustrate the use of the debugging capability of Matlab. Most of the time you will be able to see what is wrong from the Matlab error messages, but sometimes the error is not obvious. In Exercise 2 below, you will see what you might do when you just cannot see why something is wrong.

Exercise 2:

(a) Turn on debugging with the commands

```
dbstop if error
dbstop if naninf
```

or through the Breakpoints menu on the Edit window.

(b) Now, look at `demoscript.m` in the edit window. At the bottom there is a statement `%bad=1/(x-1);`. The percent is a comment character, so this statement is not executed. Make this statement active by deleting the percent character and save the changed file.

(c) Execute the script file by typing its name without the `.m` at the command line or by choosing "Run" from the menu.

(d) The division by zero caused an exception and Matlab popped up a window with the offending source line highlighted. You should also note the changed prompt in the command window. **Note:** The window with debugging information in it is the same as the edit window. It may not automatically pop up and you may have to look for it on your desktop.

(e) In the edit window, place the mouse pointer on top of the `x` in the expression `bad=1/(x-1)` and leave it there motionless for a second or so (this is called "hovering"). The value of `x` should be displayed. You can then see why the error occurred.

(f) Look at the prompt in the command window. It has changed to `K>>`. You can do any legal Matlab command at this changed prompt. In particular, you can type the name `x` to get Matlab to print out its value or, more conveniently, you can type the expression `x-1` to find its value. This is an alternative way to see the value of a variable or expression in a debug situation.

(g) You can exit from debugging mode either using the menu in the edit window or with the command

```
dbquit
```

This action will return Matlab to its usual `>>` prompt.

(h) You can turn off debugging features with the command

```
dbclear all
```

or from the debugging menu in the edit window.

Debugging (finding errors in) code you have written is the most time-consuming and least rewarding task in writing code. I am sure you think you will not be making errors, but everyone makes errors and they can be very difficult to find. You can often find your errors based on the line numbers included with Matlab's error messages, but sometimes the error remains a mystery. In that latter case, the Matlab debugger is the most powerful tool you have available.

Another important use of the debugger is for tracing through a piece of code to help you understand how it works. The benefit of the debugger used for tracing is that the effect of each statement is immediatly clear. Careful tracing is the quickest way to learn how code works.

The following exercise illustrates how you can use the debugger to trace execution. It uses the same `demoscript.m` file as before.

Exercise 3:

(a) Type the command `clear` at the command line. This will return Matlab to its state just after starting up. No variables will show in the "Workspace" windowpane.

(b) Click on the first executable line (`x=1.3`) of the `demoscript.m` file in the edit window. Go to the "Breakpoints" menu and "set" a breakpoint there. A breakpoint is a place where the debugger will always stop execution. When a breakpoint is set, a red dot will be placed next to the line number in the edit window. You can also click on the dash to the left of the line number to set a breakpoint at that line.

(c) Begin executing the file either by choosing "Run" from the debug menu in the edit window, by pressing the "Run" button, or by typing the name `demoscript` at the command line. The edit window will show that the script is poised to execute the first line of the file. (That line has not yet been executed.)

(d) Predict, in your mind, what will happen when the line is executed. In this case, the value of x will change to 1.3 and the result "x=1.3" will be printed because there is no semicolon at the end of the line.

(e) Choose "Step" from the Debug menu , press the "Step" button (hovering over a button brings up its description) or issue the command "dbstep" at the command line. This will cause the current line (line 8) to be executed. You will see the variable `x` appear in the Workspace windowpane and `x=1.3000` appear in the command window.

(f) Predict what will happen on the next step. What will the value of the variable `xsquared` become?

(g) Predict what will happen on the next step. What will the value of the variable `p` become? (An estimate is good enough.)

(h) Continue using "Step" and watch the script execute, one line at a time.

(i) If the next line were to call a function, the button next to the "Step" button is the "Step in" button. The "Step" button goes on, using the value of the function while the "Step in" button jumps to the code inside the function and steps there.

(j) Pressing the "Continue" button causes Matlab to continue executing the script until another breakpoint is reached or until the end of the script is reached.

(k) Press the "End Debugging" button to exit from debug mode. Alternatively, use the command `dbquit`.

5 Getting help

It is important to be able to get help when you need it. Matlab provides two help facilities from inside Matlab itself and a third on the web. The easiest way to get help is to use the "Help" menu at the top of the Matlab window. Command-line help is also available from the Matlab prompt by typing "`help` *command*". For example,

```
help diary
```

You will get a short description of how to use the command. You will also get a list of related commands near the bottom of the help description, and you will often find other appropriate commands there. When you write your own Matlab files, you should always include some special comments in the beginning of the file. The comments up to the first executable statement or blank line will be printed out in response to the `help` command. For example, the command

```
help demoscript
```

will give a quick help message from the first three lines of `demoscript.m`. You may notice that the first of these lines is included in the file listing in the "Current Directory" windowpane.

A second way to get help from the command prompt is the following.

```
doc
```

This command brings up a comprehensive help facility, the same one that the Help menu brings up. The content in this help facility is very similar to the one on the web from the URL:
`http://www.mathworks.com/help/matlab/index.html` We will be looking at the help facility on the web later in this lab.

6 Quitting

You exit Matlab by typing `quit` at the command line or by clicking on the "X" at the top of the Matlab window to close the window.

7 Lab summaries

You should complete a report of the results you obtained for each completed lab. This report need not be elaborate. The report consists of at least two files: the complete, original, unmodified, `diary.txt` file(s) plus a summary file. This summary file can be easily created as you do the lab by keeping a text file up in the editor and copying parts of the web page, your commands and output to the file as you work. You can find a sample summary file on my website.

This summary file is very important. It is what I will read first and, if it is well-written and the work is done correctly, I will not need to read anything else. *Never* put incorrect Matlab statements into your summary because it will take me a lot of time to discover you really didn't mean them. I will regard everything in the summary as information you believe is correct and will grade it accordingly. I expect to see

- Brief descriptions of the work you did for each exercise.
- Copies of the main Matlab commands you used for the exercises, along with the numerical results.
- Your description *must* include more detail than merely "yes, I did it and it worked."
- Names of plot files (`.jpg`) corresponding with the different exercises.
- Explanations of anything unusual or interesting, or points of confusion that you were unable to resolve outside lab.
- If you believe I have an error in a lab, please inform me of it. Explain why you think it is an error and, if you like, suggest a correction.

Summarizing your work is important not only for my convenience in grading, but also to help fix in your mind the focus of each exercise.

Equally important, the summary file helps get you into the habit of keeping track of your numerical experiments in some formal manner. When you are doing research, you may be doing hundreds of numerical experiments, and you *must* get into the habit of documenting your work or you will not remember from month to month what each one did. The idea of the summary is that you can easily refresh your memory on exactly what you did to accomplish some particular task.
Here is what I want to see in the summary file:

1. Those parts of the answers to each exercise that I ask for.
2. Explanations of what you did, in full sentences. There should be enough information here to repeat the experiment. Matlab files will help in this documentation.
3. I would like to see a few lines expressing your opinion of the point of each exercise.
4. Easily identified answers to exercises, including numerical values. I do not want to look in your diary file to try to figure out what you did. *Do not* write, "see summary file for details."
5. What was the result of the experiment? Not just the numbers, but what the experiment told you.

Here is what I do NOT want to see in the summary file:

1. Matlab error messages (unless I ask for them).
2. False starts, mistyped commands, etc.
3. Incorrect results that are corrected later.
4. Duplications of anything, unless I explicitly ask for them.

5. Large numbers of printed values, for example, the contents of a vector of length 100. I will not read all these numbers and they end up being like spam.

If you want to know how much detail to include, think of the following scenario. You have completed this course and, a year from now, a friend who is taking the course is having trouble. Your friend comes to you and asks how you did a particular exercise. You have saved your work, so you go look at it. The first place you will look is in your summary to see what you did. If the summary file contains only "Exercise 1.a: complete," you will then have to go re-read the original lab and look for your script files, *etc.* Instead, the summary should describe what you did so you can explain it in general to your friend without referring to other materials. If your friend needs more detail, you can look at the other files you wrote for the lab.

Do not write a summary of the work in today's lab. Instead, please read the following information about Matlab commands from either the PC here in the lab or from another computer on the web.

8 More on Matlab

The Mathworks, maker of Matlab, includes a short tutorial on using Matlab called *Getting Started*. This tutorial is the first part of more comprehensive Matlab documentation. The "Getting Started" tutorial is also available from the Matlab command prompt with the command `helpdesk` and also from the Help menu, and, if you have your own copy of the Matlab manuals, it comprises the "Getting Started" book.

If you feel you need more explanation of programming techniques, you may find the following reference useful.

> Charles F. Van Loan and K.-Y. Daisy Fan,
> "Insight Through Computing, A MATLAB Introduction to Computational Science and Engineering," SIAM, 2010, ISBN: 978-0-898716-91-7

The "Getting Started" tutorial is an excellent presentation of the basic capabilities of Matlab. In order to have an overview of Matlab, read through the the tutorial. There is only the equivalent of about 10 printed pages of material here, mostly very easy to understand. **Begin the "Getting Started" tutorial now, during your first lab session. Read as much of it as you can now, and complete it at home or from any convenient computer connected to the web.**

If you wish a more comprehensive introduction to Matlab, look at the Matlab documentation or look at the (much shorter) Matlab Primer. Focus on the following sections:

1. Language Fundamentals
2. Mathematics, but just the sections on Elementary Math and Linear Algebra (Matrix operations).
3. Graphics, but just the first section on 2-D and 3-D graphics.
4. Programming Scripts and Functions, but just the first four sections.

An excellent alternative to reading through the "Getting Started" material is the "Matlab Onramp."

8.1 Matlab Onramp

The Matlab Onramp Tutorial is a few hours long and covers much the same material as the "Getting Started" booklet recommended above. Subjects covered in this tutorial include:

1. Course Overview
    - Familiarize yourself with the course.
    - Course Overview

2. Commands
    - Enter commands in MATLAB to perform calculations and create variables.
    - Entering Commands
    - Storing Data in Variables
    - Using Built-in Functions and Constants
    - Desktop Overview
3. Vectors and Matrices
    - Create MATLAB variables that contain multiple elements.
    - Manually Entering Arrays
    - Creating Evenly-Spaced Vectors
    - Array Creation Functions
4. Importing Data
    - Bring data from external files into MATLAB.
    - Saving and Loading Variables
    - Import Tool
5. Indexing into and Modifying Arrays
    - Use indexing to extract and modify rows, columns, and elements of MATLAB arrays.
    - Indexing into Arrays
    - Extracting Multiple Elements
    - Changing Values in Arrays
6. Array Calculations
    - Perform calculations on entire arrays at once.
    - Performing Array Operations on Vectors
7. Calling Functions
    - Call functions to obtain multiple outputs.
    - Obtaining Multiple Outputs from Function Calls
8. Obtaining Help
    - Use the MATLAB documentation to discover information about MATLAB features.
    - Obtaining Help
9. Plotting Data
    - Visualize variables using MATLAB's plotting functions.
    - Plotting Vectors
    - Annotating Plots
    - Plots Tab
10. Review Problems

- Bring together concepts that you have learned with a project.
- Project - Electricity Usage
- Project - Audio Frequency

11. MATLAB Scripts
    - Write and save your own MATLAB programs.
    - The MATLAB Editor
    - Running a Script
12. Logical Arrays
    - Use logical expressions to help you to extract elements of interest from MATLAB arrays.
    - Logical Operations and Variables
    - Combining Logical Conditions
    - Logical Indexing
13. Programming
    - Write programs that execute code based upon some condition.
    - Flow Control
    - For Loops
14. Final Project
    - Bring together concepts that you have learned with a project.
    - Project - Stellar Motion
    - Project - Stellar Motion (Script)
15. Additional Resources
    - Discover other resources that will help you use MathWorks software.
    - Survey
    - Additional Resources

If you find the "Getting Started" information too terse or you find you need more detail, there are some video tutorial presentations on the Mathworks web site. I recommend the "Interactive Matlab Tutorial" and the "Computatational Mathematics Tutorial" that are available on the "Academia" portion of the web site. Go to `http://www.mathworks.com`, click on the icon at the top of the page consisting of three short horizontal lines, choose "Academia", and then choose "Tutorials" from the choices in the box at the bottom of the page. You may have to log in to your Matlab account, the same one you used to activate your copy of Matlab, or create a new account.

8.2 Matlab fundamentals

This tutorial is quite extensive and may take as much as ten hours to complete. Topics can be chosen individually

- Introduction
- Working with the Matlab User Interface

  - Matlab desktop
  - Saving and Loading Variables
- Variables and Expressions
  - Matlab commands
  - Command History
- Analysis and Visualization with Vectors
  - Introduction
  - Array Operations
    - Arrays
    - Scalar Expansion
  - Mathematical Operations
    - Operators
  - Plotting
    - Plotting Vectors
- Analysis and Visualization with Matrices
  - Introduction
  - Matrix Multiplication
    - Elementwise multiplication
    - Matrix multiplication
  - Function Behavior
  - Plotting Matrices
- Automating Commands with Scripts
  - Introduction
  - Scripts
    - Whale call model
    - Matlab scripts
  - Comments and Help
- Logic and Flow control
  - Introduction
  - Programming
    - Programming introduction
    - Conditional statements
    - For loops
- Writing functions
  - Introduction
  - Matlab functions
    - Matlab functions
    - Creating Functions

8.3 Interactive Computational Mathematics Tutorial

This tutorial is available at `https://www.mathworks.com/academia/student_center/tutorials/computational-math-tu`
It is also a few hours long and covers topics discussed in both 2070 and 2071. It is an excellent supplementary source for these courses. If you plan to pick particular topics, I recommend the following ones.

- Computational Mathematics Tutorial Introduction
- Linear Algebra (most relevant to 2071)
  - Introduction to Linear Algebra
  - Solving Linear Systems
  - Eigenvalue Decomposition
  - Singular Value Decomposition
- Solving Ordinary Differential Equations (most relevant to 2071)
  - Introduction to Solving Ordinary Differential Equations
  - Numerical Solution to ODEs
  - Solving First-Order Systems
  - Matlab ODE Solve Options
- Data Fitting and Working with Nonlinear Equations (most relevant to 2070)
  - Introduction to Data Fitting and Working with Nonlinear Equations
  - Data-Driven Models
  - Solving Nonlinear Equations
  - Solving for Critical Points (extra, but valuable)

Last change `$Date: 2016/08/15 17:05:37 $`