MATLAB Introduction
November 5
Rough schedule

- First week, “MATLAB as calculator”: basics, MATLAB-specific syntax, MATLAB command line, CONVERTF.m (Fortran example redone in MATLAB)
- Second week, “programming in MATLAB”: more detail about arrays, functions, “vectorized” calculations, basic profiler
- Third week: more advanced plotting, debugger, profiler, requests?
What is MATLAB?

• MATrix LABoratory: www.mathworks.com

• In contrast to Fortran, MATLAB is both a language and a numerical computing environment.
  
  – This means you can write programs (like Fortran), but you can also use it interactively to analyze and display data, simulations, etc. (In a sense, it is like a much more powerful Excel)
  
  – The interactive nature makes it extremely useful for initial analysis of data or development of algorithms
  
  – Creating a program requires a plan of what to do ahead of time; interactive use lets you develop on the fly.
  
  – MATLAB is good at moving between these two regimes,
MATLAB strengths

- There are really no *truly* standard programming languages in atmospheric science, but MATLAB comes close
- Interpreted (e.g. not compiled) language
- On the fly type conversion & variable creation
- Lots of built-in functions & toolboxes to leverage
- Lots of built in graphics
- Many ways to solve numerical problems
MATLAB weaknesses
(yes, almost the same list)

- Interpreted (e.g. not compiled) language
  - *interpreted code can be much slower*
- On the fly type conversion & variable creation
  - *Can lead to sloppy code, confusing run time errors*
- Lots of built-in functions & toolboxes to leverage
- Lots of built-in graphics
- Many ways to solve numerical problems
  - *Many ways to do the same thing can be confusing*
- Cost – each tool box is an added charge
  - but, student license is reasonable, $100 from DoIT
  - open source alternative (Octave)
    http://www.gnu.org/software/octave/
Different ways to run MATLAB

- From UNIX shell:

- The GUI
  - `user> matlab &`

- “Lightweight” options – this will run it inside the terminal window (good if you are running remotely)
  - `user> matlab -nodesktop`
  - `user> matlab -nojvm`
The GUI (lots of stuff)
Command Line

- Similar to a (graphing) calculators
- `simple_example_script.m` ...
Basic Scripts

- Can easily collect basic commands into a script file (or sometimes called a “batch script” e.g., run a batch of commands)
- Go back & forth with mouse cursor context menu, “cells” in the script file
- Examples ...
MATLAB syntax

- Similar to Fortran ("functional" language)
- Some Specifics:
  - comment character %
  - Relational operators: == ~= < > <= >=
  - Operators: − + * / ^ (for scalars – arrays are different, more on that later)
  - Arrays are indexed as A(n, m). First index is 1
  - **LOTS** of ways to create arrays (more on that later)
  - End each line with ; (otherwise the result is “echoed” to the command line)
  - Variable names are case sensitive (Search “Naming Variables” for more info)
MATLAB variable classes

- Standard things:
  - Numeric types: double precision floating point (default), single precision floating point, signed/unsigned integers
  - Logical (True or False)
  - Characters
  - Structures
  - Arrays (note that scalars are really size 1 arrays)

- Not so standard things:
  - Empty array (zero elements)
  - Cell arrays (more on these later)
  - Function handles

- Use whos or class(x) to find a variable's type.
The empty array

• Lots of ways to make one (but rarely do you want to). Example below.

• Some built-in functions return these, so mind them...

EDU>> not_empty_array = 1; empty_array = [];  
EDU>> size(not_empty_array)  
an =  
   1   1  
EDU>> size(empty_array)  
an =  
   0   0  
EDU>> isempty(not_empty_array)  
an =  
   0  
EDU>> isempty(empty_array)  
an =  
   1
Function Handle

• Can create a variable that is a reference to a function

• Useful for certain built-in numeric computations (like quad: numeric integration on any function specified by the input function handle)

• Think of this as a representation of a mathematical function (*rather than a “function” as a computer programming term.*)

• Syntax:

```matlab
function_handle = @function_name;
function_handle = @(x) f(x);
```
Function Handle

- Simple example – cos(x), integrate from 0 – pi/4.
- Analytic result is

$$\int_{0}^{\pi/4} \cos(x) \, dx = \left[ \sin(x) \right]_{0}^{\pi/4} = \sqrt{\frac{1}{2}}$$

EDU>> foo = @(t) cos(t);
EDU>> quad(foo,0,pi/4)
ans =
  0.7071
EDU>> sqrt(1/2)
ans =
  0.7071
Function Handle

• Simple example – $x^3$, and then integrate from 0 – 1.5.
• Analytic result is

$$\int_{0}^{3/2} x^3 \, dx = \left[ \frac{1}{4} x^4 \right]_{0}^{3/2} = \frac{1}{4} \left( \frac{3}{2} \right)^4 = \frac{81}{64}$$

EDU>> xcubed = @(x) x.^3;
EDU>> quad(xcubed, 0, 1.5)
an =
    1.2656
EDU>> 81/64
ans =
    1.2656
EDU>> quad(xcubed, 0, 1.5) - 81/64
ans =
    -2.2204e-16
Let's make something “useful”

- Blackbody curve (emittance) as a function of wavenumber:

\[ E(\nu, T) = 1.911 \times 10^{-8} \frac{\text{W}}{\text{m}^2 \text{cm}^{-1}} \frac{\pi \nu^3}{\exp\left(1.439 \left[\text{K cm} \frac{\nu}{T}\right]\right) - 1} \]

Create a MATLAB function handle – use quad to integrate & compare to Stefan-Boltzmann

\[ E(T) = 5.67 \times 10^{-8} \left[ \frac{\text{W}}{\text{m}^2} \right] \]
Let's make something “useful”

\[ E(\nu, T) = 1.911 \times 10^{-8}\,[\text{W/m}^2\text{cm}^{-1}] \frac{\pi \nu^3}{\exp\left(1.439 [\text{K cm}] \frac{\nu}{T}\right) - 1} \]

\[
\text{bbemit} = @(T,\nu) 1.191e-8.*\text{pi.}.*\nu.^3 ./ ... \text{ (exp}(1.439*\text{nu.}/T)-1);
\]

\[ E(T) = 5.67 \times 10^{-8}\,[\text{W m}^{-2}]\]

\[
\text{stefbolt} = @(T) 5.67e-8 * T.^4;
\]
Let's make something “useful”

EDU>> T = 330;
EDU>> stefbolt(T)

ans =

672.4172

EDU>> quad(bbemit,0,9000)
??? Input argument "nu" is undefined.

Remember that bbemit requires 2 inputs – T and nu.
Need to create another function handle that “sets” the value of T:

EDU>> quad(@(nu) bbemit(T,nu), 0, 8000)
ans =
672.5745
Revisit CONVERTF from Fortran introduction

• Reminder of what CONVERTF does:
  – Convert temperatures in Fahrenheit, to temperatures in Celsius and/or Kelvin.
  – Ask user for the number of temperatures.
  – Ask user whether or not to convert to Celsius or Kelvin.
  – Print back results to the command window.

• What this looks like in MATLAB (note, I've just translated the Fortran code, but this isn't necessarily the best use of MATLAB.)