Advanced Matlab – Introduction & Review (v. 1.0)

C. S. Tritt, Ph.D.
November 27, 2011

About Me

- Extensive programming experience (M.S. project involved Pascal programming, Ph.D. project Fortran programming)
- Have taught Fortran, Pascal, C, C++, Java and Matlab courses
- I view programming as a means to an end for BEs and MEs.

2

Structure of Course

- Mix of lectures, demos and independent exercises. I'll be focusing on practical applications and general advice, not the details.
- You should plan to get details from Matlab help (which is great for details, but seriously lacking in advice about when, how and why to use particular Matlab features.
- What we cover is intended to be broadly applicable.

3

Textbook

- No formal textbook, but I'll be using:
  - Matlab Programming for Engineers, 4th ed. by Chapman (any edition you have access to will be useful).
  - A Guide to MATLAB Object-Oriented Programming by Register (This is a great book but I don't recommend you buy it unless you plan to do some serious Matlab object oriented programming.

4

Expected Topics

- Reintroduction (including NetConnect)
- Setup and debugging tips
- I/O review and advanced I/O
- Advanced data types (cell arrays & structures)
- Dealing with errors – try-catch blocks
- Vectorization (including logical data types and logical indexing).
- Handling variable argument lists
- Function handles & function functions
- Graphic handles
- Graphical user interfaces (GUIs)
- Serial communications
- Timers & real time programming
- Compiling & embedding matlab
- Classes & object oriented programming (OOP)

5

About Matlab

- Created by smart people for smart people to solve real science and engineering problems.
- Not created systematically or based on extensive theory.
- Initially lacked some important features for large scale, group projects.
- Many features have added over time, but not necessarily in an orderly way.

6

General Review (1 of 3)

- See my Matlab reference documents for a quick review.
- Matlab environment (the *desktop*).
- Matlab basics (variables, arrays, simple I/O, operators, built-in functions and basic plotting).
- Variable, function & file names must start with letter and may not contain spaces or punctuation marks. Names are generally case sensitive.

7

General Review (2 of 3)

- Selection Flow Control (relational and logical operators, *logical* data type, branching (*if* constructs).
- Looping Flow Control (*while* loops, *for* loops).
- User defined functions, function arguments (a.k.a. parameters) and preserving data (or state) (*persistent* variables).

8

General Review (3 of 3)

- Additional Data Types (complex, string (really an array of char).
- Simple I/O Functions: *disp*, *input*, *load*, *save*, formatted low level I/O using *fscanf*, *fprintf* and *sprintf*).
- What, if anything, do you know about:
  - The *dlmread* and *dlmwrite* functions?
  - The *textscan* function?

9

Online Resources

- Like most engineers, I don't program everyday. As a result, I often have to look up language details when writing programs.
- As a result, I create Quick Reference documents and examples for my own use. I post them at http://people.msoe.edu/~tritt/Matlab/index.html.
- Also, visit for course page at http://people.msoe.edu/~tritt/ge4200/index.html for the course outline slide decks, assignments, examples and other information.

10

My Thoughts on Programming

- The best ways to learn to program is to study examples and to write your own programs.
- Reading books and paying attention in lecture are also important.
- Writing very simple test code is often the best way to learn about language features that are new to you.
- There are no shortcuts to becoming a efficient and effective programmer.

11

Review of Functions & I/O

- User functions are used to break large problems in to smaller parts for easier solution and testing.
- They also encourage code reuse.
- Each function has its own workspace (variables). Values have to be passed in and out using *arguments* (typically in calling program) or *parameters* (typcally in function).

12

Example (f2k.m)

```
function temp_k = f2k(temp_f)
%f2k converts Fahrenheit temperature to Kelvin.
%
% Converts Fahrenheit temperatures to Kelvin and returns result. Returns
% NaN if temperature is out of range.
%
% Arguments
%
% temp_f -- The Fahrenheit temperature to be converted.
%
% Returns
%
% temp_k -- The Kelvin equivalent returned by the function.
%
% Revsion History
%
% Created 11/27/06 by Dr. C. S. Tritt
% Last revised 11/27/11 by Dr. C. S. Tritt
%
% Variables
%
% temp_f -- Entered temperature in Fahrenheit.
% temp_k -- Converted temperature in Kelvin.

%Convert to Kelvin & check for validity.
    temp_k = 5./9.*(temp_f - 32.) + 273.15;
    if (temp_k < 0.) temp_k = NaN;

end % function f2k.
```

13

Example (functconv.m)

```
% File: functconv.m
%
% An interactive console program for converting temperatures.
%
% Purpose: Convert entered temperatures from degrees Fahrenheit to Kelvin
% using a function.
%
% Input(s): A valid temperature in Fahrenheit. Error checking is done in
% function f2k.
%
% Output(s): the Kelvin equivalent of the entered temperature.
%
% Calls: f2k
%
% Revsion History
%
% Created 11/27/06 by Dr. C. S. Tritt
%
% Variables
%
% temp_f -- Entered temperature in Fahrenheit.
% temp_k -- Converted temperature in Kelvin.

%Prompt user for input.
temp_f = input('Enter the temperature in deg. F: ');

%Convert to Kelvin
temp_k = f2k(temp_f);

%Display the result. Don't forget the newline.
fprintf('%6.2f deg. F = %6.2f K\n', temp_f, temp_k);
```

14

The Function Statement

- Specifies name of function and the argument and return lists.
- Each function is placed into an m-file with the same name as the function.
- Incoming (or input) parameters list the names representing values that will be passed from the caller to the function.
  - Also called dummy arguments. They are just placeholders for actual values.
- Returned (or output) parameters are a list of dummy placeholders for the values returned to the caller when the function finishes execution.

15

Invoking Functions

- A function is invoked by naming it in an expression together with a list of actual arguments.
- Can be typed directly into the command window, used in a script file or used in another function.
- The name in the calling program must exactly match the function name **including capitalization**.

16

Function Details

- When the function is invoked, the first actual argument is used in place of the first parameter (dummy argument or variable) and so on. We'll soon expand this!
- Function completes when it reaches the return or end statement.
- Each parameter in the output portion of the function **must** be assigned in the function.

17

Problem 0

- Write a function that converts pressures in common clinical (medical) units of millimeters of mercury (mm Hg) gauge into SI units of kilopascals (kPa) absolute and then use this function in an interactive program.
- Call your function mmHg2kPa and have it accept one argument (the pressure in mm Hg) and return one value (the pressure in kPa).

18

Discussion

- Definitions, quantities & relationships (variables, arguments, equations, etc.).
- Conversion factor(s).
- Function arguments and returns.
- What could go wrong?
- Design.

19

Problem 0 Details

- Don't forget to properly document program and its testing (I consider these critical aspects of the programming process).
- Submit your program (.m files) and documentation (.doc) via e-mail zipped together in a file having a name starting with your last name and ending in *Prob0*.
- Due before class Wednesday (11/30/11).

20