Creating and Using MATLAB Functions

Increasing Readability and Usability of Codes
What is a function, and when should I create one?

- Examples of functions:
  - sin, exp, plot, find, +, ./, /

- Functions Simplify a Program
  - Move code from one level to another
  - Hide ugly details from the programmer

- Use Functions when:
  - A series of steps are repeated multiple times
  - A complicated procedure must be performed
    - eg. Solving a linear system
Properties of Functions

Function Workspace vs. MATLAB Workspace

Scope
- Variables used within the function are *different* than those used elsewhere
- Function variables are only defined while the function is executing!!!

Types of variables
- Local Variables (default)
  - Variables used within a function. These are completely separate from any variables defined elsewhere (other functions, main program, etc)
- Global Variables
  - These can be seen by all routines (functions) within a program
  - Use these cautiously!
- Persistent Variables
  - Retain their value after the function is exited
Argument Lists

Getting Variables in and out of Functions

- Order of arguments in function declaration and function call
  - Order is important, names are not

- Number of arguments
  - Using “nargin” and “nargout” in programs
Creating User-Defined Functions

The Syntax

- function [result]=fun_name(arg₁, arg₂, ... argₙ)
  - Function declaration
  - result output from the function
    - Could be several variables, i.e. [a,b,c] = fun_name(...)
  - fun_name name of the function
    - Should be the same as the file name
      - MATLAB doesn’t actually use the function name
  - arg₁, arg₂, ... argₙ input parameters for the function

EXAMPLE: Create a function to compute the factorial of a number.
Recursive Functions

What is a recursive function?

• A function that calls itself
  – If some condition holds, then the function calls itself.
  – f(f(f(f(x))))

EXAMPLE: revisit the factorial function.

• Could we write this as a recursive function?
  – f(x) = x * f(x-1), for x>2 ⇒ y = f(f(f(...f(x))))...

• Algorithm:
  – Input: number to compute factorial of (x)
  – If x is larger than 2, then
    – Save the result as x * factorial(x-1)
  – Otherwise, the result is equal to x!
  – Output: result
function \( y = \text{recurFact}(x) \)

if \( x > 2 \)
    \( y = x \times \text{recurFact}(x-1); \)
else
    \( y = x; \)
end

\( X = 4 \)
\( y = 4 \times \text{recurFact}(3) \)
\( Y = 24 \)

\( X = 3 \)
\( y = 3 \times \text{recurFact}(2) \)
\( Y = 6 \)

\( X = 2 \)
\( y = 2 \)
\( Y = 2 \)
Inline Functions

A Fast Way to Create Simple Functions

- fun = inline(‘function’, ‘arg₁’, ‘arg₂’, ...);
  - Defines a function
  - arg₁...argₙ are variables passed into this function

- Example:
  f=inline(‘a*x^2+b*x+c’, ‘a’, ‘b’, ‘c’, ‘x’);
  myNum = f(1,0,0,2);

- More later...