Matlab Advanced Programming

Matt Wyant
University of Washington

Matlab as a programming Language – Strengths (as compared to C/C++/Fortran)

- Fast to write -no type declarations needed
- Memory allocation/deallocation handled automatically
- Functions and scripts can be accessed automatically from path
- Vector/Matrix operations in many dimensions built in
- Polymorphic with respect to matrix dimensions

Matlab as a programming Language – Strengths

- Fairly easy to handle strings
- Easy input/output
- Easy to build a function library
- Mathematical functions built in
- Plotting and Visualization built in
- Additional toolboxes

Matlab as a programming Language - Weaknesses

- Expensive!
- Debugging complex code can be tricky
- Not object-oriented
- Behavior of built in functions must be learned.
- No type declarations needed (example)
- Namespace collisions (example)

Outline

- Scripts vs. Functions
- Techniques for writing functions
- Structures
- Cell-Arrays
- Examples

Not included

- Matlab graphics
- Object-oriented programming
- Interfaces to other languages
- Not much on handling matrices

Matlab Scripts

plotsquared.m:

```
x = [-5:5];
y = x .^2;
plot(x,y)
```

Run with ‘plotsquared’

Advantages of Scripts

- Easy to write
- Easy to patch together and nest
- Can be cut and paste directly into Matlab command line
- Can use any variables already in the workspace

Disadvantages of Scripts

- Scripts have uncertain variable scope.
- Scripts are hard to modularize.
- Modifications can be cumbersome.
- Script inputs and outputs are not explicit.
- Scripts are hard to loop over safely.
- Scripts can be cumbersome to test.

Ways to protect against scope problems in scripts

- clear all
  - include at top of script or type it
  - sometimes not desirable
- Initialize all variables you use
- Beware assignment to parts of an un-initialized array

Suppose y is a 20x20 array. Then the following script is run:

```
x = [-5:5];
y(1:11) = x .^2;
plot(x,y)
```

Fixed version:

```
y = zeros(1:11)

x = [-5:5];
y(1:11) = x .^2;
plot(x,y)
```

Matlab Functions

- Example (complex_phase_and_magnitude.m)
- Arguments are ‘passed by value’.
- All variables are internal (unless you use GLOBAL variables), so your workspace is not altered.
- Can be called with fewer than specified input arguments.
- Can be called with fewer than specified output arguments.
- Functions can have indefinite arguments at the end of the argument list

Advantages of Functions

- Reuseable – develop your own libraries, or get functions from others
- Robust – rules all in one place (debug shared methods once)
- Sharable
- Self-documenting
- Can be used easily customize built-in matlab functions
- Provide variable scope safety
- Can be called inside of scripts
- ‘one rule, one place’

Disadvantages of Functions

- Designing them takes time and practice.
- Debugging can be more involved than for scripts.

Guidelines for writing functions

- Write at least minimal documentation
  - required inputs
  - optional inputs
  - outputs
  - units
- Use a descriptive function name
- Use descriptive variable names
- Possibly check the input arguments
  - size
  - number of dimensions
  - type

Guidelines for writing functions

- If you can, don’t assume the size of input matrices
- Write functions to apply a process to whole matrices, not just scalars (avoid looping over matrices).
- If you see repeated similar procedures, consider looping over them or making them a function.
- If you can, don’t assume the size of input matrices
- Avoid GLOBAL variables.
  - Make them arguments instead.
  - If you need persistent variables use ‘persistent’.
  - If you need lots of physical constants pass them in as a structure)
- Small simple functions can be very useful (examples)

Evolve a script into a function

- Choose a script or piece of a script that has a narrowly defined purpose.
- Choose a script or piece of a script that you are likely going to want to use multiple times.
- Identify the parameters.
- Gather all parameters at the top of the script.
- Decide which parameters are going to be changed often. These will be the input arguments.
- Identify the useful outputs of the script. These will be the output arguments.
- Choose a good name!

Matlab Structures

- Group variables
- Group data of different sizes/types (meta-data with variables)
- Sort by name
- Nest

Cell-arrays

- Containers for multiple Matlab objects of different types and sizes
- Useful for grouping character strings
- Can be used as comma-separated lists
- () used to get subset of cell array
- {} used to get contents of cells

Examples

- Exploiting Matlab matrix handling
- Passing back complex output using structures
- Extending existing Matlab functions using varargin
- Looping over variables with cell-arrays

Exploiting Matlab matrix handling

- Many built-in and user made functions can handle arbitrarily sized matrices.
- Matrices typically must all be the same size.
- The functions must use element-by-element math methods on these matrices:

```
x .^ y
x .* y
x ./ y
```

- Example: (qsat.m), (sat_vapor_pressure.m)

Passing back complex output using structures

- Example: retrieve_sounding.m
- Saves all useful information about a procedure or computation.
- Adding new info won’t break existing functions or scripts.
- Results can be passed to other functions or grouped in arrays.

Extending existing Matlab functions

- varargin allows you to create functions with an unspecified number of arguments.
- Use varargin as last argument in your function.
- In body of the function, the cell array called ‘varargin’ contains a list of the ‘extra’ arguments.
- Example: fitscatterplot.m, fitscatterplot2.m

Looping over variables with cell-arrays

Cell-arrays provide a convenient way to loop through variable data (noloop.m, loop.m).

References

- Matlab Help
  - help
  - help [topic]
  - help [function name]
  - doc [function name]
- www.mathworks.com
- <u>Graphics and GUI’s with MATLAB</u>, Patrick Marchand and O. Thomas Holland
- This talk will be placed (soon) at ‘www.atmos.washington.edu/~mwyant

Additional Topics

- Matrix manipulations and higher dimensional matrices
- Handle Graphics Techniques
- Matlab Graphics for Publication and Presentation
- Performance Tips
- NetCDF Interface
- Handling Missing Data
- Interface with C/C++ or Fortran