MATLAB Review

Selim Aksoy
Bilkent University
Department of Computer Engineering
saksoy@cs.bilkent.edu.tr

MATLAB

- MATLAB Basics
- Top-down Program Design, Relational and Logical Operators
- Branches and Loops
- Vectors and Plotting
- User-defined Functions
- Additional Data Types: 2-D Arrays, Logical Arrays, Strings
- Input/Output Functions

MATLAB Basics: Variables

- Initialization using shortcut statements
  - colon operator → first:increment:last
    - x = 1:2:10
      x =
      1 3 5 7 9
    - y = 0:0.1:0.5
      y =
      0 0.1 0.2 0.3 0.4 0.5

MATLAB Basics: Variables

- Initialization using keyboard input
  - input()
    - value = input('Enter an input value: ')
      Enter an input value: 1.25
      value =
      1.2500
    - name = input('What is your name: ', 's')
      What is your name: Selim
      name =
      Selim

MATLAB Basics: Subarrays

- Array indices start from 1
- x = [-2 0 9 1 4];
  - x(2)
    ans =
    0
  - x(4)
    ans =
    1
  - x(8)
    ??? Error
  - x(-1)
    ??? Error

MATLAB Basics: Subarrays

- y = [1 2 3; 4 5 6];
  - y(1,:)
    ans =
    1 2 3
  - y(:,2)
    ans =
    2
    5
  - y(2,1:2)
    ans =
    4 5
  - y(1,2:end)
    ans =
    2 3
  - y(:,2:end)
    ans =
    2 3
    5 6

MATLAB Basics: Subarrays

- y = [1 2 3; 4 5 6];
  - y(1,2) = -5
    y =
    1 -5 3
    4 5 6
  - y(2,1) = 0
    y =
    1 -5 3
    0 5 6
  - y(1,2:end) = [-1 9]
    y =
    1 -1 9
    0 5 6
- y = [1 2 3; 4 5 6; 7 8 9];
  - y(2:end,2:end) = 0
    y =
    1 2 3
    4 0 0
    7 0 0
  - y(2:end,2:end) = [-1 5]
    ??? Error
  - y(2,[1 3]) = -2
    y =
    1 2 3
    -2 0 -2
    7 0 0

MATLAB Basics: Displaying Data

- The disp(*array*) function
  - disp('Hello');
    Hello
  - disp(5);
    5
  - disp(['Bilkent ' 'University']);
    Bilkent University
  - name = 'Selim'; disp(['Hello ' name]);
    Hello Selim

MATLAB Basics: Displaying Data

- The fprintf(*format*, *data*) function
  - %d integer
  - %f floating point format
  - %e exponential format
  - \n new line character
  - \t tab character
- Examples:
  - fprintf('Result is %d', 3);
    Result is 3
  - fprintf('Area of a circle with radius %d is %f', 3, pi*3^2);
    Area of a circle with radius 3 is 28.274334
  - x = pi;
  - fprintf('x = %0.2f', x);
    x = 3.14

Programming Rules of Thumb

- *Learn program patterns* of general utility (branching, loops, etc.) and *use relevant patterns* for the problem at hand
- *Seek inspiration* by systematically working test data by hand and ask yourself: "what am I doing?"
- *Declare variables* for each piece of information you maintain when working problem by hand
- *Decompose* problem into manageable tasks
- *Remember* the problem's boundary conditions
- *Validate* your program by tracing it on test data with known output

Relational Operators

- Relational operators are used to represent conditions where the result of the condition is either true or false
- In MATLAB, false is represented by 0 and true is represented by 1 (non-zero)
- Don't confuse equivalance (==) with assignment (=)
- Be careful about roundoff errors during numeric comparisons (you can represent "x == y" as "abs(x-y) < eps")

Logical Operators

input a	input b	and a & b	or a | b	xor xor(a,b)	not ~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Operator Hierarchy

- Processing order of operations:
  - parenthesis (starting from the innermost)
  - ~ operators
  - exponentials (left to right)
  - multiplications and divisions (left to right)
  - additions and subtractions (left to right)
  - relational operators (left to right)
  - & operators (left to right)
  - | operators (left to right)

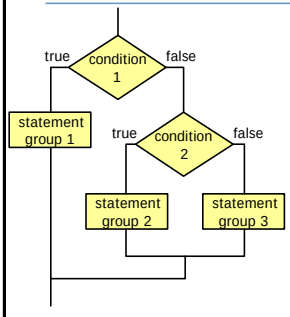
Branches

- Branches are used to select and execute specific sections of the code while skipping other sections
- Selection of different sections depend on a condition statement
- We learned:
  - if statement
  - switch statement

Branches: "if" Statement

- Conditions can be:
  - any real value (0 is false, non-zero is true)
  - combination of relational and logical operators
    - e.g. (x > 0) & (x < 10)
  - logical functions
    - isempty()
    - isnumeric(), ischar()
    - isinf(), isnan()
    - exist()

Branches: "if-elseif-else" Statement




```
if ( condition 1 ),
    statement 1
    statement 2
    ...                    } statement group 1
elseif ( condition 2 ),
    statement 1
    statement 2
    ...                    } statement group 2
else
    statement 1
    statement 2
    ...                    } statement group 3
end
```

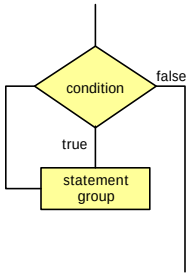

Branches: "switch" Statement

```
switch ( expression ),
case {value set 1},
    statement 1
    statement 2            } statement group 1
    ...
case {value set 2},
    statement 1
    statement 2            } statement group 2
    ...
...
otherwise,
    statement 1
    statement 2            } optional statement group that is executed if none of the cases is satisfied
    ...
end
```


Loops

- Loops are used to execute a sequence of statements more than once
- We learned:
  - while loop
  - for loop
- They differ in how the repetition is controlled

Loops: “while” Loop



- Statements are executed indefinitely as long as the condition is satisfied

```
while ( condition ),
    statement 1
    statement 2
    ...
end
```

statement group

Loops: “for” Loop

- Statements are executed a specified number of times

```
for index = expression,
    statement 1
    statement 2
    ...
end
```

statement group

- Expression is usually a vector in shortcut notation first:increment:last

Loops: “break/continue” Statements

- Break statement terminates the execution of a loop and passes the control to the next statement after the end of the loop
- Continue statement terminates the current pass through the loop and returns control to the top of the loop

Advice

- Use indentation to improve the readability of your code
- Never modify the value of a loop index inside the loop
- Allocate all arrays used in a loop before executing the loop
- If it is possible to implement a calculation either with a loop or using vectors, always use vectors
- Use built-in MATLAB functions as much as possible instead of reimplementing them

Initializing Vectors

- linspace(x1,x2) generates a row vector of 100 linearly equally spaced points between x1 and x2
- linspace(x1,x2,N) generates N points between x1 and x2
  - x = linspace(10,20,5)

```
x =
   10.00   12.50   15.00   17.50   20.00
```

- logspace(x1,x2) can be used for logarithmically equally spaced points

Vector Input to Functions

- You can call built-in functions with array inputs
- The function is applied to all elements of the array
- The result is an array with the same size as the input array

Vector Operations

- Vector-vector operations (element-by-element operations)
  - x = [1 2 3 4 5]; y = [2 -1 4 3 -2];
  - z = x + y

    z =

    3 1 7 7 3
  - z = x .* y

    z =

    2 -2 12 12 -10
  - z = x ./ y

    z =

    0.5000 -2.0000 0.7500 1.3333 -2.5000

Vector Operations

- Vector-vector operations (element-by-element operations)
  - z = x .^ y
  - z =

    1.00 0.50 81.00 64.00 0.04
- Use .*, ./, .^ for element-by-element operations
- Vector dimensions must be the same

Plotting Summary

- plot(x,y)
  linear plot of vector y vs. vector x
- title('text'), xlabel('text'), ylabel('text')
  labels the figure, x-axis and y-axis
- axis([xmin xmax ymin ymax])
  sets axes' limits
- legend('string1', 'string2', 'string3', ...)
  adds a legend using the specified strings
- hold on/off
  allows/disallows adding subsequent graphs to the current graph

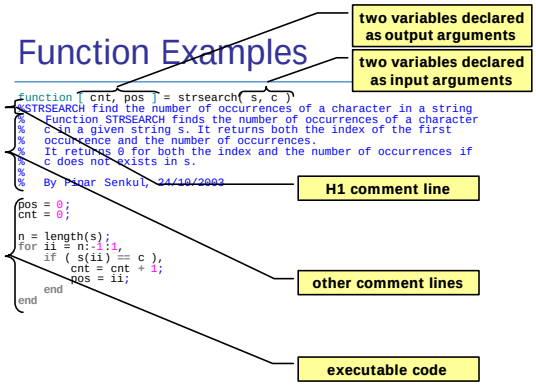
Scripts

- A script is just a collection of MATLAB statements
- Running a script is the same as running the statements in the command window
- Scripts and the command window share the same set of variables, also called global variables

Functions

- A function is a black box that gets some input and produces some output
- We do not care about the inner workings of a function
- Functions provide reusable code
- Functions simplify debugging
- Functions have private workspaces
  - The only variables in the calling program that can be seen by the function are those in the input list
  - The only variables in the function that can be seen by the calling program are those in the output list

Function Examples



2-D Arrays

- Adding the elements of a matrix

```
function s = sum_elements(a)

[r,c] = size(a);
s = 0;
for ii = 1:r,
   for jj = 1:c,
      s = s + a(ii,jj);
   end
end
```


Logical Arrays

- Created by relational and logical operators
- Can be used as masks for arithmetic operations
- A mask is an array that selects the elements of another array so that the operation is applied to the selected elements but not to the remaining elements

Logical Arrays

- Examples
  - b = [1 2 3; 4 5 6; 7 8 9]

```
b =
   1   2   3
   4   5   6
   7   8   9
```

  - c = b > 5

```
c =
   0   0   0
   0   0   1
   1   1   1
```

  - whos

Name	Size	Bytes	Class
a	2x4	64	double array
b	3x3	72	double array
c	3x3	72	double array (logical)

Strings

- A string is an array of characters
  - s = 'abc'

    is equivalent to s = ['a' 'b' 'c']
- All operations that apply to vectors and arrays can be used together with strings as well
  - s(1) → 'a'
  - s([1 2]) = 'XX' → s = 'XXc'
  - s(end) → 'c'

Character Arrays

- 2-D character arrays
  - s = ['my first string'; 'my second string']

    ??? Error
  - s = char('my first string', 'my second string')

```
s =
my first string
my second string
```

    } char function automatically pads strings
  - size(s) → [2 16]
  - size(deblank(s(1,:))) → [1 15]

String Functions

- ischar(), isletter(), isspace()
- strcmp() : returns 1 if two strings are identical
- upper(): Lowercase-to-uppercase
- lower() : Uppercase-to-lowercase
- findstr() : finds one string within another one
- strtok() : finds a token in a string
- strrep() : replaces one string with another
- num2str(), str2num()
- sprintf() is identical to fprintf() but output is a string