EECE 231: Introduction to Matlab: Part II
Reading: Bielajew, Chapter 11

OBJECTIVES

- Introduce Matlab terminology
- Introduce loops in Matlab
- Introduce control structures in Matlab
- Introduce Matlab operators
- Introduce Matlab script M-files
- Matlab functions

OUTLINE

TERMINOLOGY

- *scalar*: is a 1×1 array
- *row vector*: is a row of length M or a $1 \times M$ array.
- *column vector*: is a column of length N or an $N \times 1$ array.
- *matrix*: is an $N \times M$ array.
- examples:
  - `>> x= 9; y = ones(1,10); z=rand(9,1); w=rand(4,5);whos`

OUTLINE

THE FOR LOOP

- The `for` loop takes an iterator, an iteration range, and a loop body
  - The range is typically an array
  - The iterator assumes the values in the range one column at a time
  - The iterator can not be re-assigned inside the loop body

```
%for i array
 % Some loop body using i where i can not be
 % reassigned. i is array(:,1) the first pass,
 % array(:,2) the second and so on.
%end
```

EXAMPLES FOR LOOP

```
% sum all numbers between 1 and 10
sum = 0;
for i = 1:10
  sum = sum + i;
end
disp(sum);
```

```
% sum all odd numbers
%  between 1  and 10
i = 1:2:10;
sum = 0;
for j = i
  sum = sum + j;
end
disp(sum);
```

ITERATING WITH MULTI-ROW COLUMNS

```
% iterating with multi-row columns
i = [ 1, 2, 3, 4, 5; 6, 7,  8, 9, 10];
sum = 0;
for j = i
  % j is tied to the columns of i
  % accumulate the two
  % columns of j
  sum = sum + j(1) + j(2)
end
```

- The loop iterates 5 times (`length(i)`)
- At each iteration *j* is set to a column of *i*.
- Illustrates the general concept of the looping index in Matlab

USEFUL FUNCTIONS

- *size* returns the number of rows and columns.
- *length* returns the maximum between the number of rows and the number of columns.
- *rand* returns an array of uniformly distributed pseudo-random numbers between 0 and 1
- *plot* plots a two dimensional figure.
- *surf* plots a three dimensional figure.

```
%2D plot
N=100;f=zeros(1,N);
x=linspace(-5*pi,5*pi,N);
for i = 1:N
  if (x(i) == 0)
    f(i) = 1;
  else
    f(i) = sin(x(i))/x(i);
  end
end
plot(x,f);
```

```
%3D plot
g=zeros(N,N);
for i = 1:N
 for j = 1:N
   g(i,j) = f(i) * f(j);
 end
end
surf(x,x,g);
```

THE WHILE LOOP

```
while expression
 % Some loop body
 % executes repeatedly as long as
 % expression evaluates to true
end
```

- Similar to C++
- Different than C++ only in syntax

OUTLINE

THE IF/ELSEIF/ELSE STRUCTURE

- The `if/elseif/else` is similar to C++
  - `elseif` is a keyword as well
  - `end` terminates the selection structure
- Indentation is almost necessary for code comprehension

```
if expression1
  % some code that executes if expression1 is true
elseif expression2
  % some code that executes if expression 1 is false
  % and expressin2 is true
elseif expression3
  % some code that executes if expressions 1,2 are false
  % and expressin3 is true
else
  % some code that executes if expressions 1,2,3
  % are false
end
```

OUTLINE

OPERATORS

Operation	Symbol	Example	Precedence
Parenthesized expression	$(,)$	$(1+2)/3$	Highest
Exponentiation	^	2^3	Medium high
multiplication	$*$	$x*y$	Moderate
division	$/$	x/y	Moderate
division	$\backslash$	$x\backslash y$	Moderate
addition	$+$	$x+y$	Medium low
subtraction	$-$	$x-y$	Medium low
assignment	$=$	$y=x$	Lowest

- Exponentiation: x^y read x to the power y
- Not typical: back slash division (aka right division)
  - $y\backslash x$ is equivalent to x/y
- For the current Matlab precedence table: `>> help precedence`.

MATLAB OPERATOR EXAMPLES

- All are 2-D matrix operations

```
>> a = 2* ones(2);
>> b = a/4;
>> a*b
>> a+b
```

MATLAB POINT-WISE OPERATORS

- Some Matlab operators have pointwise versions:
  - Point-wise exponentiation: $x.\hat{}\,y$
  - Point-wise multiplication: $x.*y$
  - Point-wise division: $x./y$ or $y.\backslash x$
- One to one array entry mapping

```
>> x = 2* ones(2);
>> y = [0,1;2,3];
>> x.*y
>> x.^y
>> x./y
```

LOGICAL OPERATORS

- Similar to Matlab:
  - `<, <=, >, >=, ==`
- Negation
  - $\sim$ instead of ! (Not)
  - $\sim=$ instead of $!=$ (not equivalent to)
- Logical and
  - `&` instead of `&&`
- Logical or
  - `|` instead of `||`

LOGICAL OPERATORS ON ARRAYS

- Multi-dimensional operators: work on arrays and can return arraysS

```
>> [1  2 3]<=[5 10 0]
ans =
     1     1     0
>> ~[1 0 1]
ans =
     0     1     0
```

TRANSPOSE OPERATOR ‘’

- Gives transpose of matrix.

```
>> a = [1 2 ; 1 2 ; 3 4 ]
a =
     1     2
     1     2
     3     4
>> a'
ans =
     1     1     3
     2     2     4
```

TRANSPOSE OPERATOR WITH COMPLEX NUMBERS

- If matrix entries are complex numbers, operator gives complex conjugate of transpose.
- The complex conjugate of a complex number $a + ib$ is the number with the complex part negated $a - ib$

```
>> a  = [i 1 ; 1+2*i 1]
a =
          0 + 1.0000i   1.0000
   1.0000 + 2.0000i   1.0000
>> a'
ans =
          0 - 1.0000i   1.0000 - 2.0000i
             1.0000                1.0000
```

OUTLINE

SCRIPT FILES

- Commands entered in a Matlab window can be organized and stored in files.
- The file name should contain the ".m" extension.
- Enter the file name to execute the commands.
  - If there is a builtin Matlab command with the same name, it takes precedence.
  - If there is a variable with the same name it takes precedence.

USEFUL MATLAB FUNCTIONS IN SCRIPTS

- *disp* displays the string or the value of the variable.
- *echo* toggles echoing the statements that are not supressed.
  - *echoon* turns echoing on, and *echooff* turns it off.
- *input* takes input from the user and issues a prompt
  - `x = input('please enter a value for x');` is similar to `cout << "please enter a value for x"; cin >> x;`
- *keyboard* gives control to the Matlab interpreter.
  - Can experiment with intermediary results of the script.
  - Type *return* on the command prompt to give control back to the script.
- *pause* continue after user presses a keyboard key.
  - *pause*(x) wait for x seconds and then continue.

FORMATTED SPRINTF FUNCTION

- `sprintf(format, A1, ..., An)` takes
  - a `format` string with special formatting specifications, and
  - an argument list that matches the format specifications.
- a format specification starts with the % character
  - the % character is followed by a format conversion specifier:
    - %d: matches a scalar in the argument list.
    - %g: matches a double floating number.
    - See specifications of C/C++ sprintf for full documentation.
- \n: line feed, \t: tab, \b: backspace
- \\: back slash character
- %%: the percent character
- Function `sprintf` returns the formatted string

EXAMPLE SPRINTF FUNCTION

- The three dots allow continuation on the next line.
- %s matches a string.
- %f matches a double float number as well.
- The `.2` prefix before `f` specifies the precision.
- Example:

```
>> price = 3.7834;
>> day = 22;
>> s = sprintf('Price of bread on %d-12-2014 was $%.2f.',day, price);
>> disp(s)
Price of bread on 22-12-2014 was $3.78.
```

- C++ equivalent:

```
cout<<"Prince of bread on "<<day<<"-12-2014 was $"<<price<<endl;
```

- Unlike C++, a string is inside ' ' (single quotes) in Matalb instead of " " (double quotes)

MATLAB FUNCTIONS

- To define a new function, write function body:

```
function  [out1,...,outN] = functionName(in1,...,inM)
 ...
 out1 = ...; % assign values of output variables
 ...
 outN = ...;
 return;
```

- We can put function body in a separate file called functionName.m, or in the file of another function, or in a script file.
- The general syntax of a function call is:

```
[out1,...,outN] = functionName(in1,...,inM)
```

MATLAB FUNCTIONS: LARGEST EXAMPLE

- Function body (possibly in a file called largest.m):

```
function  [x] = largest(a,b)
 if a>b,
   x =a ;
 else
   x = b;
 end;
 return;
```

- Function call example:

```
>> w = 3.2;
>> c = largest(2,w);
>> disp(c)
       3.2000
```

- Compare with C++:

```
double  largest(double a, double b)  {
  double x;
  if(a>b) x =a ;
  else x = b;
  return x;
 }
```

MATLAB FUNCTIONS: QUADRATIC ROOTS EXAMPLE

- Function body (possibly in a file called quadRoots.m):

```
function [root1,root2] = quadRoots(A,B,C)
 root1 = (-B + sqrt(B^2 -4*A*C))/(2*A);
 root2 = (-B - sqrt(B^2 -4*A*C))/(2*A);
return % This is optional
```

- Function call example:

```
>> [r1 r2] = quadRoots(1,0,-1);
>> disp([r1 r2])
     1    -1

>> [r1 r2] = quadRoots(1,1,1);
>> disp([r1 r2])
  -0.5000 + 0.8660i  -0.5000 - 0.8660i
```

- In C++, we cannot directly return 2 values

MATLAB FUNCTIONS: ARRAY MIN AND MAX EXAMPLE

- Function body (possibly in a file called arrayMinMax.m):

```
function  [minimum, maximum] = arrayMinMax(A)
  n = length(A);
  maximum=A(1);
  minimum=A(1);
  for i=2:n,
    if A(i)> maximum,
            maximum = A(i);
    end;
   if A(i)< minimum,
        minimum=A(i);
    end;
  end
return; % optional
```

- Function call example:

```
>> [mn mx] = arrayMinMax([2 -3 4 5.9  1.1]);
>> disp([mn mx])
   -3.0000    5.9000
```

MATLAB FUNCTIONS: PASSING AND RETURNING ARRAYS

- Matlab functions can take arrays (vectors, matrices) as input argument
- Can also return vectors and matrices
- Unlike C++, array input arguments are in general passed by **copy** and not by reference
- However Matlab does some smart tricks. For instance, if the function does not modify the input array, Matlab passes the array by reference (because this is faster).

OUTLINE

SUMMARY

- `for` loops take arrays as ranges for iterators.
- Scalar, column vector, row vector, and matrix are terms that will live with us.
- `elseif` is a keyword.
- Operators in Matlab operate on matrices and arrays.
- Script files can house Matlab “programs”.
- User input function: *input*
- formatting strings: *sprintf*
- Matlab functions

SUMMARY -

EECE 231: INTRODUCTION TO MATLAB: PART II

READING: BIELAJEW, CHAPTER 11