EECE 231: INTRODUCTION TO MATLAB
READING: BIELAJEW, CHAPTER 11

OBJECTIVES

- Introduce the Matlab environment
- Illustrate that Matlab is an interpretation program
- Understand data representation in multidimensional arrays (matrices)
- Cover array construction
- Cover array indexing and slicing
- Cover saving and loading data

OUTLINE

DATA TYPES AND VARIABLES

- Variable names: similar to C++
- Types of variables and declarations
  - C++: several types, MUST declare variable types explicitly
  - Matlab: types implied from context

STATEMENTS

- C++: terminated by a semicolon
- Matlab:
  - terminated by and end-of-line (the `Enter` keyboard key)
  - the semicolon ‘`;`’ suppresses the echo of the statement

```
>> i = 10;
>> disp (i)
    10
>>
```

CONTROL STRUCTURES

- C++: curly braces (‘{’ ‘}’) start and end `if-else` and `while` blocks

```
while (i < 10 ) {
  i = i+1;
}
```

```
if ( i<= b) {
  a = 2*b+1;
} else {
  a = 2*b;
}
```

- Matlab:
  - block starts after conditional expression
  - block ends with the `end` keyword

```
while (i < 10 )
  i = i+1;
end
```

```
if ( i<= b)
  a = 2*b+1;
else
  a = 2*b;
end
```

EXAMPLE: SUM OF THE FIRST 10 INTEGERS

C++

```
int main() {
  int i=0, s=0;
  while (i < 10) {
    i=i+1;
    s=s+i;
  }
  cout << s << endl;
  return 0;
}
```

Matlab

```
i=0;
s=0;
while (i < 10)
  i=i+1;
  s=s+i;
end
disp(s)
```

COMMENTS

- C++:
  - line comments: // this is a C++ line comment
  - block comments: /* this is a C++ block comment */
- Matlab: comments start with the % character
  - % this is Matlab line comment

MATLAB SCRIPTS AND FUNCTIONS

- Matlab script: Matlab code organized in a file
  - Matlab interprets the script line by line
- Matlab function: implement functions with parameters and return values
  - More on that later on

OUTLINE

MATLAB VERSUS C++

MATRICES AND ARRAYS IN MATLAB

- Array construction
- Array Indexing
- Files

SUMMARY

ARRAYS

- Arrays and matrices are interchangeable terms from now on
- An *N*-dimensional array is a sequence of objects organized in *N* dimensions
  - Each dimension has a fixed size
  - The total size of the *completely filled* array is the product of the sizes of its dimensions

ARRAY CONSTRUCTION

- Arrays are fundamental data types in Matlab
- Play with `zeros` and `ones`:
  - `>> x = ones(7,1)`: creates a 7 × 1 array *x* filled with 1.0
  - `>> x = zeros(5,8)`: creates a 5 × 8 array *x* filled with 0.0
  - `>> whos`: lists variables, their dimensions, and their size in memory so far
- The column '`:`' operator creates a list of consecutive numbers
  - `>> x = 1:10`
  - `x = 1 2 3 4 5 6 7 8 9 10`
- The two-column '`:`∆`:`' operator creates a list of ∆ separated numbers
  - `>> y = 1:2:10`
  - `y = 1 3 5 7 9`

FUNCTION LINSPACE FOR ARRAY CONSTRUCTION

- The `linspace(`s`,`e`,`N`)` function constructs an $N \times 1$ array
  - The objects are equally spaced
  - They start at s and end at e
- `>> linspace(1,2,11)`

```
1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2
```

Literal array construction

- Square brackets enclose array elements
- Within the square brackets,
  - the comma ‘`,`’ operator is a column separator

```
>> x = [1.0,5.2,9.7]
x =
   1.0000 5.2000 9.7000
```

  - The semicolon ‘`;`’ operator is a row separator

```
>> y = [1.0,5.2,9.7;3.4,9.3,10.7]
y =
   1.0000 5.2000 9.7000
   3.4000 9.3000 10.7000
```

ARRAY INDEXING

- **No Off-by-one (zero-based) indexing**
- Indices start at 1
- $x(i)$ refers to the i^{th} element of array x
  - Note the use of parentheses in indexing

```
>> x = [3,5;7,9]
x =
   3  5
   7  9
```

x(1,1) evaluates to 3 % row 1, column 1
x(2,1) evaluates to 7 % row 2, column 1
x(1,2) evaluates to 5 % row 1, column 2
x(2,2) evaluates to 9 % row 2, column 2

ARRAY *slice* INDEXING

- Within the array indexing (parenthesis)
  - The slicing column '`:`' means the entire row, or column
    - `>> x = [1,2;3,4;5,6]`
    - `>> y = x(:,2)`: returns the second column of *x*
    - `>> z = x(1,:)`: returns the first row or *x*
- Partial slicing takes ranges (using the '`:`' range operator, or an array of indices)
  - `u = x(1:2,1)`: rows 1 until 2 of column 1
  - `v = x(2:3,:)`: rows 2 until 3 of all columns

Save data

```
sindata = zeros(2,1000);
sindata (1,:) = linspace(0,2*pi,1000);
sindata (2,:) = sin(sindata(1,:));  % array operati
save sindata -ascii -double
```

- Fill the two rows of *sindata* with zeros
- Fill the first row with 1000 values equally spaced between 0 and $2 * pi$
    - Serves as the x-axis
- Fill the second row with the sine value of each corresponding element in the first row.
    - Serves as the y-axis
- Save the data

LOAD THE DATA

```
clear % clears all variables
load sindata;
x = sindata(1,:); y = sindata(2,:);
plot(x,y);
pause
close
```

- Clear existing data and variables
- Load the data from the file
- Plot the data in a figure and wait for any key stroke to close it

OUTLINE

MATLAB VERSUS C++

MATRICES AND ARRAYS IN MATLAB
- Array construction
- Array Indexing
- Files

SUMMARY

SUMMARY

- Matlab syntax by contrast to C++
- Types are implicit, arrays are fundamental types,
- Similar identifiers, % line comments, `end` closes blocks, '`;`' supresses echo
- Functions for array construction: ones, zeros, linspace, ranges, . . .
- Saving and loading data
- **command `help` follows by another command name is very instrumental**

EECE 231: INTRODUCTION TO MATLAB
READING: BIELAJEW, CHAPTER 11