**CSCI 2150L**

**Session 2 Topics:**

1. We started the session with operator precedence i.e. a priority list of operators which is looked upon when a computer try to evaluate a mathematical expression. Relevant to this topic we briefly discussed infix, postfix, and prefix expression. The reason for this nomenclature is the relative position of the operator with respect to the operands. For e.g. 2 + 2 \* 4 is an infix expression as the operator is in between the operands. The corresponding postfix and prefix expression would be 2 2 4 \* + and + \* 2 4 2 respectively. As you can observe how the position of the operators with respect to the operands correspond to their names.

We ended the discussion by giving the following list of operators arranged according to their priority within an arbitrary expression.

* Parenthesis ( )
* Exponents ^
* Multiplication/division \* and /
* Addition/subtraction + and –

Notice that this list also includes parenthesis which have highest precedence in an expression as we always evaluate the sub-expression enclosed by the parenthesis first. Also the operators mentioned in the same bullet point have same precedence for e.g. \* and / have same precedence.

1. Next we looked into different ways to generate arrays in MATLAB.

* At first we generated some regular arrays where difference between any pair of consecutive numbers in the array was same. For e.g. command a = 1: 1: 10 would generate an array from 1 to 10 with a step 1 i.e. difference between any two consecutive numbers is 1. Likewise another array b from 2 to 20 with a step 2 was generated with command b = 2: 2: 20. If you remember the array t = -5\*pi : pi/50 : 5\*pi we used for trigonometric functions, it also was a regular array with step pi/50. (**N.B**: default step in MATLAB is 1. Therefore the command a = 1:10 is also a valid command in MATLAB where the step is 1 by default.)
* Array with all zeroes were generated with command zeros (row,col).
* Array with all ones were generated with command ones (row,col).
* Array with random values between 0 and 1 were generated with command rand (row,col)

(Both row and col in the commands above are integers specifying number of rows and columns of each array).

1. We then discussed plot, a nice feature in MATLAB to visualize different functional graphs. The generic command for 2-D plot is plot(A,B) where A and B, both are matrices specifying X and Y axis of the graph.

* We chose trigonometric functions sine and cosine to observe their plots.
* An array t was created as follows; t = -5\*pi : pi/50 : 5\*pi so that the values of sine and cosine calculated for t would generate points which if plotted would give us smooth and continuous curves.
* We calculated two arrays a = sin(t) and b = cos(t) so that a and b holds the values of sine and cosine respectively for the entire range of t.
* To observe sine plot we used command plot(t,a) where t had values for X co-ordinate and a provided the values for Y co-ordinate.
* Likewise to observe cosine plot we used command plot(t,b).
* Subsequently we plotted sine and cosine together on a same figure by using the command plot(t,a,t,b).
* We added legend to the plot by command legend(‘comment1’,’comment2’). Here comment1 and comment2 corresponds to sine and cosine curves respectively. In class comment1 and comment2 were ‘sine’ a ‘cosine’ respectively.
* We added X and Y label to the plot by using the commands xlabel(‘x\_comment’) and ylabel(‘y\_comment’) respectively. In class we labeled X axis as t and Y axis as sin(t) or cos(t). You are at your liberty to add any comment of your choice. Preferably those comments should be relevant to the plot.
* We added title to the plot by command title(‘title\_comment’). In class we used a suitable title for the graph.

4) Data type conversions:

1. Decimal to Base

Convert decimal to base N number in string

**Syntax**

str = dec2base(d, base)  
str = dec2base(d, base, n)

str = dec2base(d, base) converts the nonnegative integer d to the specified base. d must be a nonnegative integer smaller than 2^52, and base must be an integer between 2 and 36. The returned argument str is a string.

str = dec2base(d, base, n) produces a representation with at least n digits.

2. Decimal to Binary

Convert decimal to binary number in string

## Syntax

str = dec2bin(d)  
str = dec2bin(d,n)

## Description

str = dec2bin(d) returns the binary representation of d as a string. d must be a nonnegative integer smaller than 2^52.

str = dec2bin(d,n) produces a binary representation with at least n bits.

The output of dec2bin is independent of the endian settings of the computer you are using.

3. Decimal to hexadecimal

Convert decimal to hexadecimal number in string

## Syntax

str = dec2hex(d)  
str = dec2hex(d, n)

## 

## aDescription

str = dec2hex(d) converts the decimal integer d to its hexadecimal representation stored in a MATLAB® string. dmust be a nonnegative integer smaller than 2^52. MATLAB converts noninteger inputs, such as those of classdouble or char, to their integer equivalents before converting to hexadecimal.

str = dec2hex(d, n) produces a hexadecimal representation with at least n digits.

1. Binary to Decimal

## Syntax

bin2dec(*binarystr*)

## Description

bin2dec(*binarystr*) interprets the binary string *binarystr* and returns the equivalent decimal number.

bin2dec ignores any space (' ') characters in the input string.