MATLAB Notes

Week 1

`clc` command clears the command window.

`clear` remove all variables, `clear x y z` clear only `x`,`y`,`z` variables from the memory.

`who` command displays the variables, `whos`displays also information about their size.

`format type` command format value as wanted type.

`sqrt(x)` square root.

`nthroot(x,n)` Real `n`th root of real number `x`.

`exp(x)` exponential.

`abs(x)` absolute value.

`log(x)` natural logarithm (ln).

`log10(x)` base 10 logarithm.

`factorial(x)` the factorial function (x must be positive integer).

`round(x)` round to nearest integers.

`fix(x)` round toward zero.

`ceil(x)` round toward infinity.

`floor(x)` rount toward minus infinity.

`rem(x,y)` returns the remainder `x` after devided by `y`.

`sign(x)` Signum function. Return 1 if `x>0`, -1 if `x<0`, and 0 if `x==0`.

Pre-defined variables are: `pi` number pi, `inf` infinity, `NaN` not a number,

Trigonometric function names same as `sin(x)`, `cos(x)`… is in radians, `sind(x)`, `cosd(x)`…is in degrees. Inverse trigonometric functions start with ‘a’, `asin(x)`,`acos(x)`… and hyperbolic functions end with ‘h’, `sinh(x)`, `cosh(x)`…

Week 2 - Arrays

Creating a Vector

`variable_name = [1 3 5]` define a vector `[1 3 5]` and assign it to a variable.

`m:q:n` create vector with constant spacing. `m` is first term, `q` is spacing, `n` is last term.

`linspace(xi,xf,n)` create vector with linear(equal) spacing. `xi` is first elemet, `xf` is last element, n is number of elements.

Creating a Array

`variable_name = [1st_row_elements; 2nd_row_elements;...;last_row_elements]` create array and assign it to variable, semicolons `;` specify new line, also new line can be created with ‘Enter’ key.

zeros, ones and eye Commands

`zeros(m,n)` create mxn matrix with all elements 0.

`ones(m,n)` create mxn matrix with all elements 1.

`eye(n)` create nxn square matrix which diagonal elements are 1 and other elements are 0.

Indexing with Arrays

Let `V` be a variable of a vector. `V(5)` gets 5th element of `V`.

Let `A` be a variable of a matrix. `A(3,5)` gets matrix’s (3rd row,5th column) element.

`A(m:n,p:q)` get elements between from m-th row to n-th row, and p-th column to q-th column. All elements are optional.

Assigning element to a vector or matrix is same, `=` operator assign element to specified indexes. Multiple assignment for specified indexes or only assignment to specified index is possible. For deleting elements from arrays or vectors, `[]` element is used.

Array Operators and Commands

Let `A` is a variable of a matrix, `A'` is **transpose** of matrix `A`.

`length(A)` return numbers of elements in vector A.

`size(A)` return a row vector with `[row_count,column_count]` values of A.

`reshape(A,m,n)` creates a mxn matrix from A. A should be bigger or equal to mxn sizes.

`diag(v)` if `v` is a vector, create square matrix with elements of v in the diagonal. if `v` is a matrix, creates a vector from diagonal elements of `v`.

Week 3 - Operations with Arrays

`inv(A)` or `A^-1` finds inverse of A.

For AX = B equation, `A\B` or `inv(A) * B` return `X`.

For element-by-element operations, dot `.` is used in front of operators. `.*` for multiplication, `.^`for exponentiation, `./` for right division, `.\` for left division

`mean(v)` v is a vector, returns the mean value.

`max(v)` v is vector, return largest elements.

`max(A)` A is matrix, return `[d,n]`, d is largest value and n is n is the position of the element (first largest element).

`min(A)` is same as `max(A)`, but for smallest element.

`sum(v)` return the sum of elements.

`sort(v)` arranges the elements of the vector in ascending order.

`median(v)` returns the median value of elements.

`std(v)` return standard deviation of the elements.

`det(A)` return determinan of **square matrix A**.

`dot(v1,v2)` returns dot product of two vector.

`cross(v1,v2)` returns cross product of two vectors (vector sizes must be equal).

rand Command

`rand` command is using for generating **random floating point**.

```
rand %generates random number 0<x<1
rand(1,n)  %generates vector of n random numbers
rand(n) %generates nxn square matrix with random numbers
rand(m,n) %generates mxn matrix with all elements are random
number
randperm(n) %generates vector with n elements that random
permutation of integers 1 through n
```

`(b-a)*rand + a` generate random floating number between `a` and `b`

randi Command

`randi` command is using for generate **random integer**.

```
randi(imax) %generate random integer between 1 and imax
randi(imax,n) %generate nxn matrix with random integers that
elements are between 1 and imax
randi(imax,m,n) %generates mxn matrix with random integers
that elements are between 1 and imax
```

Instead of imax, `[imin imax]` can be used, this will generate between imin and imax numbers.

randn Command

`randn` command generates with mean 0 and standard deviation of 1. Usage of `randn` same as `rand` command.

Week 4

`input('prompt_message','s')` get input from user, `s` define input as string format.

`display(variable)` displays the variable value.

fprintf command

`fprintf('text')` display the text. Escape characters can be used such as: `\n` Newline, `\b` Backspace, `\t` Horizontal tab.

we can use `fprintf` command to display mix of text and numerical data as: `fprintf('text %-5.2f text', variable_name)`, `%` marks the spot where variable will be displayed, `-`Left-justifies the number,`+` prints sign character (+ or -), `0` adds zeros if number shorter than field, `5` field width and `2` is precision, `f` is conversion character. We can use as mush variable as we want.

Common conversion characters are

`e` - Lowecase exponential

`E` - Uppercase exponential

`f` - Fixed-point

`g` - The shorter of `e` or `f`

`G` - The shorter of `E` or `f`

Using fprintf to save output to a file

1. Open a file using `fopen` command.
2. Write the output to the opened file using `fprintf` command
3. Close the file using `fclose` command

Usage:

```
fid1 = fopen('file_name','permission'); %open file with
permission, default is r(read), for write w(write) used.
fprintf(fid1,'text % text', variable)_name); %print output to
the file
fclose(fid1); %close file
```

save and load Command

`save file_name` and `save('file_name')` saves variables of workspace to file.

or

`save file_name variable_name variable_name2` or `save('file_name','variable_name', 'variable_name2')` save specified variables to file.

`load` command usage is the same as save command, but using for loading variables from file.

Week 5 - 2D Plots

plot Command

`plot(x,y, 'line_specifiers', 'property_name', property_value)` create 2D plots, x and y are vectors, specifiers and properties optional.

Examples:

```
plot(x,y) %blue solid line
plot(x,y,'r') %red solid line
plot(x,y,'--y') %yellow dashed line
plot(x,y,'*') %points marked with *, no line between points
plot(x,y,'g:d') %green dotted line marked with diamond markers
plot(x,y,'-
mo','LineWidth',2,'markersize',12,'MarkerEdgeColor',
'g','markerfacecolor','y')
```

fplot Command

`fplot('function',limits,'line_specifiers')`

Examples:

```
fplot('math_expression',[-3 3]) % limits as [xmin,xmax] or
[xmin,xmax ymin,ymax]
```

hold Commands

`hold on` command is used to hold plotting from display, and `hold off` command display the plots to the user.

axis Command

`axis([xmin,xma,ymin.ymax])` overwrite the default limits of plot command.

```
axis equal %set same scale
axis square %set axes to be square
axis tight %set axis limits to range of data
```

line Command

`line(x,y,'propert_name',property_value)`, property arguments are optional.

grid Command

`grid on` Adds grid lines to the plot `grid off` Removes grid lines from the plot

Making Plots with Logarithmic Axes

```
semilogy(x,y) %log (base 10) scale for y axis and  linear
scale for x axis
semilogx(x,y) %log (base 10) scale for x axis and  linear
scale for y axis
loglog(x,y) %log (base 10) scale for both axes
```

errorbar Command

`errorbar(x,y,e)` , e vector holds error value at each point, x,y,e vectors should have same size

Other Plottings

```
bar(x,y) %vertical bar
barh(x,y) %horizontal bar
stairs(x,y) %stairs
stem(x,y) %stem
pie(x) %pie
hist(y) %histogram
hist(y,x)  %histogram with x bar
polar(thetas,radiuses,'line_specifiers') %polar plotting
```

Multiple Plots on the Same Page

`subplot(m,n,p)`, page divided into mxn rectangular subplots, p is the index value of plot.

also plotting into different figures possible with `figure` command. After plotting a graph, `figure` command open new figure to plot.

Week 8 - Programming in MATLAB

Operators

`~=` not equal operator, different than other programming languages.

`~` Not operator.

`xor(a,b)` return true (1) if one operand is true and the other is false.

`all(A)` return true if all elements in A is true (1).

`any(A)` return true if any of elements of A true.

`find(A>d)` return elements indexes of A that is larger than d (other operator can be used).

Conditionals

```
if
.
.
elseif
.
.
else
.
.
end
```

```
switch value
    case value1
    ..
    ..
    case value2
    ..
    ..
    otherwise
```

```
    ..
    ..
end
```

```
for k = f:s:t %f is first value to assign x, s is increment
value, t last value
.
.
end
```

```
while
.
.
end
```

`break` command:

If inside a loop, terminate it.

If outside of a loop, terminate the execution of file.

`continue` command:

Terminate current iteration, continue to next iteration in loop.

Formatting Plot

`xlabel('text')`, `ylabel('text')` and `zlabel('text')`

`title('text')` place title automatically, `gtitle('text')` want user input for position when command run.

`legend('string1','string2',...,pos)` command shows a sample of the line type for each plotted graph.

```
pos = -1 %outside the axes on the right side
pos = 0 %inside the axis
pos = 1 %upper-right corner(default)
pos = 2 %upper-left corner
pos = 3 %lower-left corner
pos = 4 %lower-right corner
```

Week 7 - Functions

First line in a function file must be the function definition line. Definition line:

```
function [output arguments] = function_name(input arguments)
```

and end with `end` keyword at the last row.

Example:

```
function [A]=arearect(a,b)
  A=a*b;
end
```

In a function, all variables are local, to define a global variable, **global** keyword should be used, all capital letters recommended for global variable name

```
global VARIABLE_NAME
```

Anonymous Functions

anonymous function is for one mathematical expression, it is created by typing in command:

```
anonymous_function_name = @ (arguments) expression
```

usage:

```
anonymous_function_name(argument1,argument2)
```

Function As Parameter

We can define argument as function with name as same as arguments, but in usage, functions passed through `''` symbols into functions as argument in usage.

Week 8 - Three Dimensional Plots

Line Plots

A three-dimensional line plot created by **plot3** command, x,y,z are vectors, **line_specifiers** are optional specifiers that define type and color of line and markers, properties are optional with values that can used to specify line width, marker’s size, edge and fill colors.

```
plot3(x,y,z,'line_specifiers', 'PropertyName',property_value)
```

Creating a grid in the xy-plane (Cartesian coordinates)

X and Y are matrix of the coordinates of the grid points that function gives as output, x and y is vector inputs.

```
[X,Y] = meshgrid(x,y)
```

Making Mesh and Surface Plots

```
mesh(X,Y,Z)
```

```
surf(X,Y,Z)
```

Example:

```
x=-3:3; %creates a vector
y=-3:3; %creates a vector
[X,Y]=meshgrid(x,y); %creates meshgrid (X,Y matrices)
Z = 'mathematical_expression based on x and y'; %find z value
based on x and y variables
mesh(X,Y,Z) %plot mesh
xlabel('x'); ylabel('y'); zlabel('z'); %labels
```

Mesh Curtain Plot

```
meshz(X,Y,Z)
```

Mesh, Surface Contour Plot

```
meshc(X,Y,Z)
```

```
surfc(X,Y,Z)
```

Waterfall Plot

```
waterfall(X,Y,Z)
```

3-D Contour Plot

n is (optional) number of contour levels.

```
contour3(X,Y,Z,n)
```

2-D Contour Plot

Draws a projections of contour levels on the xy-plane.

n is (optional) number of contour levels.

```
contour(X,Y,Z,n)
```

3-D Pie Plot

X is a vector with same length as X of 0's and 1's. 1 offsets the slice from the center.

```
pie3(X,explode)
```

Polar Coordinates Grid in the xy-plane

3-D plot of a function based on r and theta (for example`z = r \theta`)

```
[th,r] = meshgrid((0:5:360)*pi/180, 0:.1,2); %create meshgrid
of values of theta and r
Z=r.*th; %calculate value of z at all points by element-by-
element operation
[X,Y] = pol2cart(th,r); %Convert polar coordinates to
cartesian coordinates with built-in  function 'pol2cart'
mesh(X,Y,Z); %plot 3-D
```

View Command

`view(az,el)` or `view([az,el])`

The view command can be used to plot projections of 3-D plots on various planes.

`az` is azimuth, angle in degrees in the xy-plane relative to -y-axis, defined in counterclockwise direction

`el` is elevation, angle in degrees from xy-planerelated to z-axis.

`view(2)` set defaults values to `az=0` and `el=90` (projection onto xy-plane);

`view(3)` set defualts values to `az=-37.5` and `el=30`.