Matlab / Octave Notes

by adgsenpai

Basic Operations

Variables

```
a = 3;
b = 'hi';
```

Output

```
a  = pi;
a           % Simply give at the prompt
disp(a);    % print function in
disp(sprintf('Vaue of pi is: %0.2f', a))        % C style printing
```

Arthimetic Operations

```
5 + 6
5 - 4
5 * 8
1 / 2
2 ^ 6       % Exponentation
```

Logical Opeartions

```
1 == 2      % false
1 ~= 2      % not equal to
1 && 0      % AND
1 || 0      % OR
xor(1, 0)
```

Vectors and Matrices

Vectors

```
row = [1, 2, 3]     % Row vector , is a separator in a row

col = [1; 2; 3]     % Column vector ; separates the rows

length(row)         % Gives the length of vector/ length of longest dimension in
matrix
```

In-built Functions

Mathematical

```
v = [0.26, 2.22, 4]
matrix = [1, 23; 30, 4; 5, 6]

log(3)              % Logrithmic
exp([1, 2])         % Exponential
abs([-1, 2, -3])    % Absolute
-v                  % similar to -1 * v

[val, ind] = max(v) % Returns an array of which has max value and it's index
max(matrix)     % Returns column-wise maximum

find([1, 0, 3, 2])      % Returns the indexes of all the non-zero elements
magic(3)        % returns a 3x3 magic square matrix

sum(matrix)     % Sum of a matrix
floor(v)
ceil(v)
```

Matrices

```
matrix = [1, 2; 3, 4; 5, 6]
eye(3)          % Diagonal matrix of order 3
ones(2, 3)      % [1, 1, 1; 1, 1, 1]
zeros(1, 3)     % [0, 0, 0]
mat = rand(3, 2)    % A random 3x2 matrix
randn(1, 100)   % Random matrix following a normal distribution

size(matrix)    % Gives the dimensions of the matrix as a matrix[row, column]
size(<matrix>, [axis])

matrix(3, 2)    % A usual matrix indexing
matrix(:,2)     % : means all the elements in the corrsponding row/column.
matrix([1 3],:) % Everything from the 1st and 3rd row

matrix = [matrix, [7; 8; 9]]   % Appending a row to the matrix(right side)
matrix(:, 3) = [10; 11; 12]    % Assiging a row
```

```
matrix(:)       % Put all the elements into a single column vector

mat_concat = [matrix mat]   % Concatenate matrices side by side [matrix, mat]
mat_concat = [matrix; mat]  % Concatenate matrices one above the other

matrix * mat    % Matrix multiplication
matrix .* mat   % This is element-wise multiplication, similarly .+, ./, .^

matrix'         % Matrix transpose
transpose(matrix)

pinv(matrix)    % Pseudo Inverse matrix
```

Iterators

```
v = 1:0.1:2     % Start:step-size:end deafult step-size is 1
```

Plotting

```
a = magic(5)
plot(x, y)              % Makes a two dimensional graph
hold on;                % To plot one or more plots at a time
xlabel("label")         % Label on the x-axis
ylabel("label")         % Label on the y-axis

print -dpng '<filename>'    % To save into a file
subplot(1, 2, 1)        % Divide the figure into 1 row 2 columns and access the
1st figure

imagesc(a), colorbar, colormap gray;    % Plots using colors

hist(randn(1, 100))     % Plots a histogram hist(<vector>, [bins])
```

Control Statements

If else

```
v = zeros(10, 1);

i = 0;
while true,
    v(i) = i*i;
    if i == 6,
        break;
    else
        disp("Value is not 6")
```

```
    end;
end;
```

For

```
v = zeros(10, 1);

for i=1:10,
    v(i) = i*i;
end;
```

While

```
v = zeros(10, 1);

i = 0;
while i < 5,
    v(i) = 10*i;
    i = i + 1;
end;
```

Functions

```
function y = squareNumber(x),
    y = x ^ 2;
end;
```

Sketching Multiple Graphs

Parameters(@(var),function,range,color)

```
fplot(@(t) exp(0.6667*t)-1.5,[-2 2],'b')
hold on
fplot(@(t) cos(t*t),[-2 2],'g')
hold off
grid on
```

Gauss_Seidel Method A

```
function xnew=gauss_seidel(A,b,xold)
n=size(A)*(1);
At=A;
xnew=xold;
for k=1:n
 At(k,k)=0;
end
for k=1:n
 xnew(k)=(b(k)-At(k,:)*xnew)/A(k,k);
end
end
```

Gauss_Seidel Method B

```
function GuassSeidel(A,B, convergeVal, maxIteration)
    N = length(B);
    X = zeros(N,1);
    err = ones(N,1);
    for i = 1:N
        j = 1:N;
        j(i) = [];
        C = abs(A(i,j));
        Check(i,1) = abs(A(i,i)) - sum(B);
    end
    iter = 0;
    while max(err) > convergeVal
        iter = iter + 1;
        Z = X;
        for i = 1:N
            j = 1:N;
            j(i) = [];
            Xtemp = X;
            Xtemp(i) = [];
            X(i,1) = (B(i,1) - sum(A(i,j) * Xtemp)) / A(i,i);
        end
        Xs = X;
        err = sqrt((X - Z).^2);
        if iter >= maxIteration
            break;
        end
    end
    disp("Solutions using Guass Seidel Model : ");
    disp(Xs)
    disp("Number of iterations :"),disp(iter);
end
```

Jacobi Method

```
% Method to solve a linear system via jacobi iteration
% A: matrix in Ax = b
% b: column vector in Ax = b
% N: number of iterations
% returns: column vector solution after N iterations

function sol = jacobi_method(A, b, N)
    diagonal = diag(diag(A)); % strip out the diagonal
    diag_deleted = A - diagonal; % delete the diagonal

    sol = zeros(size(b, 1), 1); % initial guess of zero

    for i = 1:N
        % computing the matrix inverse
        sol = diagonal \ (b - diag_deleted * sol);
    end

end
```

Finding coefficients of any equation with given data.

Example below of where `x - stock price y-day of month`

writing in the form of `p0-ae^(0.01x)`

```
% x and y values
x = [700 678 984 547 442 418 300]'
y = [6 12 20 5 6 3 2]'

% finding coefficients equation in the form p0-ae^(0.01x)

% first generate expression for your equation
expr = fittype({'1','-exp(0.01*(x))'},'coefficients',{'p0','a'});

% now fit the equation this will spit out your coefficients values
coeff = fit(x,y,expr);

% this sets the range of the graph
xx = linspace(0,2000);

%plotting graph
plot(x,y,'o',xx,coeff(xx),'r-');
legend('Data Points','Exponential curve fit','Location','best');
```