Recitation 2

Recitation Instructor: Shivam Verma
Email: shivamverma@nyu.edu
Ph: 718-362-7836
Office hours: WWH 605 (2.50 - 4.50 pm, Tuesdays)

Installing MATLAB (Updated)

1. Visit the ITS help desk at 10 Astor Place (4th floor) during working hours and ask them to help you use MATLAB using the VCL - http://www.nyu.edu/life/resources-and-services/information-technology/help-and-service-status/nyu-it-service-desk.html

2. Install a trial version of MATLAB at https://www.mathworks.com/programs/trials/trial_request.html
This will allow you to use MATLAB for 30 days. (Obtain a new trial license after that.)

More about MATLAB

- **Recap**
  - x = 1:0.1:100 %defines row vector from 1 to 100 with 0.1 gap
  - y = exp(x); z = x.^2 %dot operator applies operation to every element
  - A = [1,2,3; 4,5,6; 7,8,9] %defines a matrix
  - A*[1;2;1] %matrix-vector product
  - A*A %matrix-matrix multiplication
  - Z = zeros(10) %defines 10x10 matrix of zeros. Also see ones/eye
  - A(1,2) %returns element at 1st row, 2nd column
  - if-else conditional
    Eg.

```
%example of conditional
myage = 23;
can_drink=false;
can_drive=false;

if(myage<18)
    can_drink=false;
    can_drive=false;
    disp('Too young to drink or drive');
elseif(myage>=18 && myage<25)
    can_drink=false;
```

```
        can_drive=true;
        disp('You can drive, but not drink');
    else
        can_drink=true;
        can_drive=true;
        disp('You can drink and drive. Dont do both together!');
    end;
```

- for/while loop

  Eg.

```
vec=zeros(5,1);
for i=1:size(vec,1)
    vec(i)=rand(1); %adds a random number to vec's element
end;
vec
```

- anonymous functions

- **Writing a procedure**

```
function return_variable = function_name(param1,param2,...)
    %add commands here
end
```

Eg 1.

```
function sq_x = square(x)
    sq_x = x.^2;
end
```

Eg 2.

```
function z = inner_product(x,y)
    z = zeros(size(x));
    for i=1:size(x,1)
        z(i) = x(i)*y(i);
    end;
end
```

Remember, functions need to be stored in a separate MATLAB ".m" file, with the same name as the function_name. For example, `square` should be stored as `square.m`, and `inner_product` as `inner_product.m`.

Let's solve a few problems using Newton's method

1. $f(x) = 1/x + ln(x) - 2, x_0 = 0.5$
2. $f(x) = x^5 - 0.25, x_0 = 0.5$
3. $f(x) = \sqrt{3x+1}, x_0 = 0.5$
4. $f(x) = x^3, x_0 = 0.5$

Sample procedures for Newton method

```
%using number of iterations
function xn = newton1(f,df,x0,iter)
    for i=1:iter
        x0 = x0 - f(x0)/df(x0);
    end
    xn=x0;
end
```

Save above script in a file called **newton1.m** and run the following code on MATLAB command window:

```
>> format long e
>> f = @(x) 1/x + log(x) - 2;
>> df = @(x) -1/(x.^2) + 1/x;
>> x0 = 0.5;
>> iter = 100;
>> xn = newton1(f,df,x0,iter)

Output:
>> xn = 0.317844432899373
```

–

```
%using tolerance
function xn = newton2(f,df,x0,tol)
    x1 = x0 - f(x0)/df(x0); %first iteration
    count = 1;
    while abs(x1-x0)>tol
        x0 = x1;
        x1 = x0 - f(x0)/df(x0);
        count = count + 1;
    end
    xn=x0;
    count
end
```

Save above script in a file called **newton2.m** and run the following code on MATLAB command window:

```
>> format long e
>> f = @(x) 1/x + log(x) - 2;
>> df = @(x) -1/(x.^2) + 1/x;
>> x0 = 0.5;
```

```
>> tol = 1e-6; %sets tolerance as 10^(-6)
>> xn = newton2(f,df,x0,tol)

Output:
>> count = 7
>> xn = 0.317844426413115
```

You should be able to reproduce these results. Try the other three functions above and see what you get. Also try using this sample code to write functions using the Bisection, Secant and Illinois methods.

To know how to plot these functions, check out the sample code in last week's recitation notes. Also use `help plot` on the MATLAB command window to know more.

Note: You can use the **fzero** function in MATLAB for finding roots Type **`help fzero`** on the MATLAB command window for more information.

Rate of convergence: Check out https://www.math.ust.hk/~mamu/courses/231/Slides/ch02_2b.pdf for an extensive and very helpful discussion on the rate of convergence of fixed point method.

Tips about using MATLAB for Numerical Analysis

- Use **format long e** (scientific notation) during evaluation
- Use a **tolerance of 10^{-6}**, though you can also try and find the maximum accuracy possible (in MATLAB/Octave, this should be 10^{-16})
- Use procedures in MATLAB for Bisection/Newton/etc. methods (easy to reproduce)
- **Plot the function in MATLAB** to get an intuition of the roots, and choose starting points close to the root. Try experimenting with the location of the starting point and note what you observe (in terms of number of iterations, convergence etc.).