# AMSC 460 - Computational Methods

## Table of Contents



# HOMEWORK 1 - Problem 1

Let f(x) = e^x + x^2 _ 5x.

(a) The bracket [1.5, 2] contains a root. Explain why using the Intermediate Value Theorem. For this bracket, estimate the number of iterations N that would be needed to compute the root to an accuracy of 10^-4.

```
clear all
format compact
close all
syms f(x) x y

f(x) = exp(x) + x^2 - 5*x;
double(f(1.5))

ans =
   -0.7683

double(f(2))

ans =
    1.3891

          Because f(x) is continuous on [1.5, 2], f(1.5) = -0.7683 < 0 and f(2) =
          Then by the Intermediate Theorem f(x) has at least one root on [1.5, 2].
```

(b) The bracket given in (a) contains a root, but there is another root. Find a bracket for it. Then use the bisection method to find the two roots to an accuracy of 10^-4.

```
double(f(0))
fprintf(['Because f(0)= %g > 0 f(1.5)= %g < 0. So find a root in', ...
        ' [0, 1.5]'], double(f(0)),double(f(1.5)))

ans =
     1
Because f(0)= 1 > 0 f(1.5)= -0.768311 < 0. So find a root in [0, 1.5]

bisect(@(x) exp(x) + x^2 - 5*x, 0, 1, 0.0001)

Undefined function 'bisect' for input arguments of type
 'function_handle'.
```

```
Error in Hw1ds (line 36)
bisect(@(x) exp(x) + x^2 - 5*x, 0, 1, 0.0001)

bisect(@(x) exp(x) + x^2 - 5*x, 1.5, 2, 0.0001)

          So roots are at x = 0.2805 and at x = 1.7339.
```

# HOMEWORK 1 - Problem 2

Consider the cubic f(x) = x^3 _ x _ 1.

(a) Use the MATLAB command fzero to find a root in the interval [1, 2].

```
fzero(@(x) x^3 - x - 1,1)
```

(b) Show that f(x) = 0 can be rewritten as a fixed point problem for both the functions (i) g1(x) = x^3 _ 1, and (ii) g2(x)=(1+x)^(1/3).

```
Let f(x) = x^3 # x # 1 = 0, then we can have x^3 # x # 1 = 0
                                    and then     x^3 # 1 = x
So we can write g1(x) = x^3 # 1.

Also, we can have x^3 # x # 1 = 0
             then         x^3 = 1 + x
             then           x = (1 + x)^(1/3)
And write g2(x) = (1 + x)^(1/3).
```

(c) Which of the functions g1 and g2 is a contraction mapping near the root r from part (a)? Which of g1 or g2 will be successful in making the iteration xi+1 = g(xi) converge locally to the root r?

```
diff((1 + x)^(1/3))

          Take the derivative of g1 and g2, then we have g1'= 3 * x^2 and g2'= 1/(
          Since g1'(1) = 3 and g1'(2) = 12 and g1' is strictly increasing on [1,2]
          g2' is continuous and strictly decreasing on [1,2],
          g2'(1) = 0.21 and g2'(2) = 0.1602
          So # L,  0 # L < 1 s.t |g2'(x)| # L < 1   # x # [1,2]     g2'converges.
          Thus by the Contraction Mapping Theorem only g2 is a constraction on [1,
          g2 will be successful in making the iteration xi+1 = g(xi) converge loca
```

(d) Write a script or function in MATLAB to carry out 10 steps of the fixed point iteration for both g1 and g2, each starting with the guess x0 = 0. What approximate root does your algorithm give for g1? For g2?. Are your results consistent with the analysis from part (c)?

```
function x = fpi_root(g,x0,steps)
    x = x0;
    iter = 0;

    while ( iter < steps)
        xNew = g(x);
        x = xNew;
        iter = iter + 1;
        fprintf('\tAfter %g steps, root = %g\n', iter, xNew)
    end
end
```

```
g1 = @(x) x.^3 - 1;
g2 = @(x) (1 + x).^(1/3);

disp('Fixed point iteration for g1 starts with x0=0:')
fpi_root(g1,0,10)
fprintf('The approximate root I got for g1 is %.15g',ans)

disp('Fixed point iteration for g2 starts with x0=0:')
fpi_root(g2,0,10)
fprintf('The approximate root I got for g2 is %.15g',ans)
```

```
Only g2 is successful in making the iteration xi+1 = g(xi) converge loca
The results consistent with the analysis from part (c).
```

# HOMEWORK 1 - Problem 3

(a) Write a MATLAB program to implement Newton's method for root finding.

Code for Newton's method:

```
clear all
syms x
f = input('Type your equation please: f = ');
x = input('The starting guess x0 = ');
xNew = x + 100;
fd = inline(diff(sym(f)));

iter = 0;
err = 100;

while err > 10^-8
    xNew = x - (f(x)./fd(x));
    err = abs(x-xNew);
    x = xNew;
    iter = iter + 1;
    fprintf('\tAfter %g steps, root = %.15g\n', iter, xNew)
end
```

(b) To compare root finding algorithms, we will approximate $\sqrt{2}$ using two methods: Newton and Bisection. Using the equation f(x) = x^2 _ 2 = 0, use your program from part (a) to ensure $\sqrt{2}$ is obtained. For Newton, use x0=2, and for Bisection use the starting bracket [1, 2]. In each case use 10^$_{8}$ for the error tolerance.

Use Newton's method:

```
clear all
syms x
f = @(x) x^2-2;   % Given f(x) = x^2 # 2
x = 2;  % The starting guess x0 = 2
xNew = x + 100;
fd = inline(diff(sym(f)));

iter = 0;
err = 100;

while err > 10^-8
```

```
    xNew = x - (f(x)./fd(x));
    err = abs(x-xNew);
    x = xNew;
    iter = iter + 1;
    fprintf('\tAfter %g steps, root = %.15g\n', iter, xNew)
end
```

Use Bisection method

```
f = @(x) x^2-2; a=1; b=2;
fa = f(a); fb = f(b);
k=0;

while (b-a)/2 > 10^-8
  c = (a+b)/2;
  fc = f(c);
  k = k+1;
  fprintf('\tAfter %g steps, root = %.15g\n', k, c)

  if fc == 0
    break
  end
  if sign(fc)*sign(fa) < 0
    b = c; fb = fc;
  else
    a = c; fa = fc;
  end
end
xc = (a+b)/2;
```

(c) Modify the algorithms to keep track of the absolute error en =|r _ xn| at each iteration. Store these errors in a vector (for plotting purposes). Then plot the absolute errors on the same graph, and with a semilogarithmic y-axis (use semilogy in MATLAB). Which algorithm used the least steps to achieve the required error tolerance?

Modified Newton's method:

```
clear all
syms x
f = @(x) x^2-2;    % Given f(x) = x^2 # 2
x = 2;  % The starting guess x0 = 2
r = sqrt(2);       % Given root =  #2
xNew = x + 100;
fd = inline(diff(sym(f)));

iter = 0;
err = 100;
en = 0;

while err > 10^-8
    xNew = x - (f(x)./fd(x));
    err = abs(x-xNew);
    x = xNew;
    en = abs(r-x);
    iter = iter + 1;
```

```
    N(iter) = en;
    fprintf(['\tAfter %g steps, root = %.15g,',...
        ' absolute error = %.15g\n'], iter, xNew, en)
end
```

Modified Bisection method:

```
f = @(x) x^2-2; a=1; b=2;
fa = f(a); fb = f(b);
k=0; Ben = 0; r=sqrt(2);

while (b-a)/2 > 10^-8
  c = (a+b)/2;
  Ben = abs(r-c);
  fc = f(c);
  k = k+1;
  B(k) = Ben;
  fprintf(['\tAfter %g steps, root = %.15g,',...
        ' absolute error = %.15g\n'], k, c, Ben)

  if fc == 0
    break
  end
  if sign(fc)*sign(fa) < 0
    b = c; fb = fc;
  else
    a = c; fa = fc;
  end
end
xc = (a+b)/2;
```

Plot of absolute errors

```
hold on;semilogy(B);semilogy(N);
title 'The graph of absolute error for Newton’s method and Bisection
 method ';
legend({'Bisection method','Newton’s method'});
hold off;
```

```
          We can see the Newton's method used the least steps to achieve the requi
```

# HOMEWORK 1 - Problem 4

4. Suppose you wish to find values of x where f (x) = 0, but f is given first in terms of y, and then a second equation determines y for any given x. Let f(y) = y^3 + 3*y + 1, and y + x = e^(-6*y) . To evaluate f given x, the second equation must first be solved for y and then this value substituted into f(y).

(a) Describe how the bisection method could be used to find an x such that f(x) = 0.

```
          According to the Intermediate Theorem we can to find an interval where z
          Then try different number a and b to get f(a)>0 and f(b)<0, then we have
          Use bisection method to find the zero of f(y) in the interval.
          As the interval become smaller and smaller, we got the midpoint close en
          Then we can plug the zero of f(y) which is the y value it into y + x = e
```

```
the x value we need.
```

(b) Writing f as f(y(x)), explain how Newton's method could be used to find the root x such that f(x) = 0.

```
First we take a gusee where is the zero of f(y(x)) could be and denote i
then we apporximate the root by let y_(n+1) = y_n - [f(y_n)/f'(y_n)] unt
precise value is reach. then plug that y_(n+1) into y + x = e^(-6*y) to
```

(c) Using MATLAB (and any method you prefer), find the root x where f (x) = 0.

Use Bisection method

```
clear all
f = @(y) y^3 + 3*y +1;
f(0); f(-1);
fprintf(['Because f(0)= %g > 0 f(-1)= %g < 0. So find a root in', ...
        ' [-1, 0]'], f(0),f(-1))

a=-1; b=0;
fa = f(a); fb = f(b);
k=0;

while (b-a)/2 > 10^-8
  c = (a+b)/2;
  fc = f(c);
  k = k+1;
  fprintf('\tAfter %g steps, root = %.15g\n', k, c)

  if fc == 0
    break
  end
  if sign(fc)*sign(fa) < 0
    b = c; fb = fc;
  else
    a = c; fa = fc;
  end
end
xc = (a+b)/2;
fprintf('Then we have y = %0.15g is an approxiamte zero of f(y)',c)
```

```
Plug this y into y + x = e^(-6*y)
```

```
syms y x
y = c;
x = exp(-6*y) - y
fprintf('Then we have x = %0.15g, and f(x) = 0',x)


fNew = @(h) ( 4*x*(h.^2) - (h.^3) - 6*(x^2)*h + 4*(x^3) );
fNew = @(h) ( 4*x*h.^3 - h.^4 - 6*x^2*h.^2 + 4*h*x^3 )./(h);
```

*Published with MATLAB® R2020b*