MATH2019 (2019-2020) Coursework 1

Table of Contents

NAME:JAKE DENTON, STUDENT ID:14322189

Question 1

See the file bisectMeth.m

Question 2

Following code uses the bisection method to compute approximations to the root of f and produces a table illustrating its convergence.

```
% Data
f = @(x) x.^3 + 2*x.^2 -6;
a = 1;
b = 2;
Nmax = 20;

% Bisection Method
[p_vec,fp_vec] = bisectMeth(f,Nmax,a,b);

% Create table

format long g
N=1:Nmax; %defines row vector with Nmax entries
n=N'; %takes transpose of N giving us a column vector for the table
table(n,p_vec,fp_vec)%produces the 20x3 table given
```

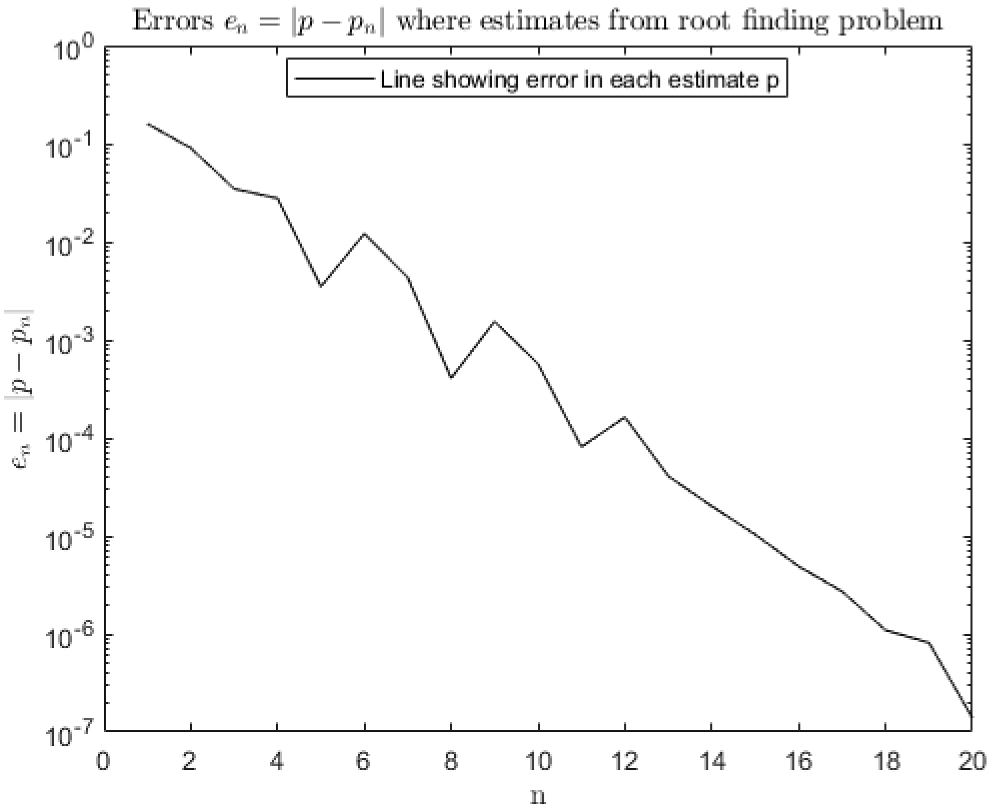
```
ans =

  20×3 table

    n          p_vec                   fp_vec
    __    ________________    _____________________
```

1	1.5	1.875
2	1.25	-0.921875
3	1.375	0.380859375
4	1.3125	-0.293701171875
5	1.34375	0.037689208984375
6	1.328125	-0.129467010498047
7	1.3359375	-0.0462555885314941
8	1.33984375	-0.00437504053115845
9	1.341796875	0.0166340991854668
10	1.3408203125	0.00612378586083651
11	1.34033203125	0.000872937147505581
12	1.340087890625	-0.00175141052750405
13	1.3402099609375	-0.000439326404375606
14	1.34027099609375	0.000216782942288773
15	1.34024047851563	-0.000111277338277205
16	1.34025573730469	5.27514001866791e-05
17	1.34024810791016	-2.9263319499151e-05
18	1.34025192260742	1.17439527302921e-05
19	1.34025001525879	-8.75970528735337e-06
20	1.34025096893311	1.49211824584938e-06

Question 3

Following code creates a corresponding figure displaying the error convergence

```
e=abs(1.3402508301291-p_vec); %defines vector e as absolute value of
 approximate solution minus the estimates p
set(groot,'DefaultTextInterpreter','latex')
semilogy(N,e,'k')%produces logarithmic plot with y axis in base 10 of
 error of estimates for root of f against n
title('Errors $e_n=|p-p_n|$ where estimates from root finding
 problem');%rest of this code formats the graph
xlabel('n');
ylabel('$e_n=|p-p_n|$');
legend('Line showing error in each estimate p', 'Location', 'Best');
```



Question 4

See the file fpiterMeth.m

Question 5

Following code uses fixed-point iteration to compute approximations to the root of f and produces a table illustrating its convergence.

```
g=@(x)x-(1/8)*(x^3+2*x^2-6); %defines the given g(x) as an anonymous
 function which can be used by fpiterMeth
Nmax=20;%the number of estimates for the fixed point we require
 fpiterMeth to compute
p_vec=fpiterMeth(g,Nmax,1);%calls fpiterMeth to produce the column
 vector of estimates p
N=1:Nmax;
n=N';%defines a column vector from 1 to Nmax for use in the table to
 show the number of each estimate (showing convergence)
format long g
table(n,p_vec)%produces a 20x2 table using the column vectors n and
 p_vec (which contains our estimates)
```

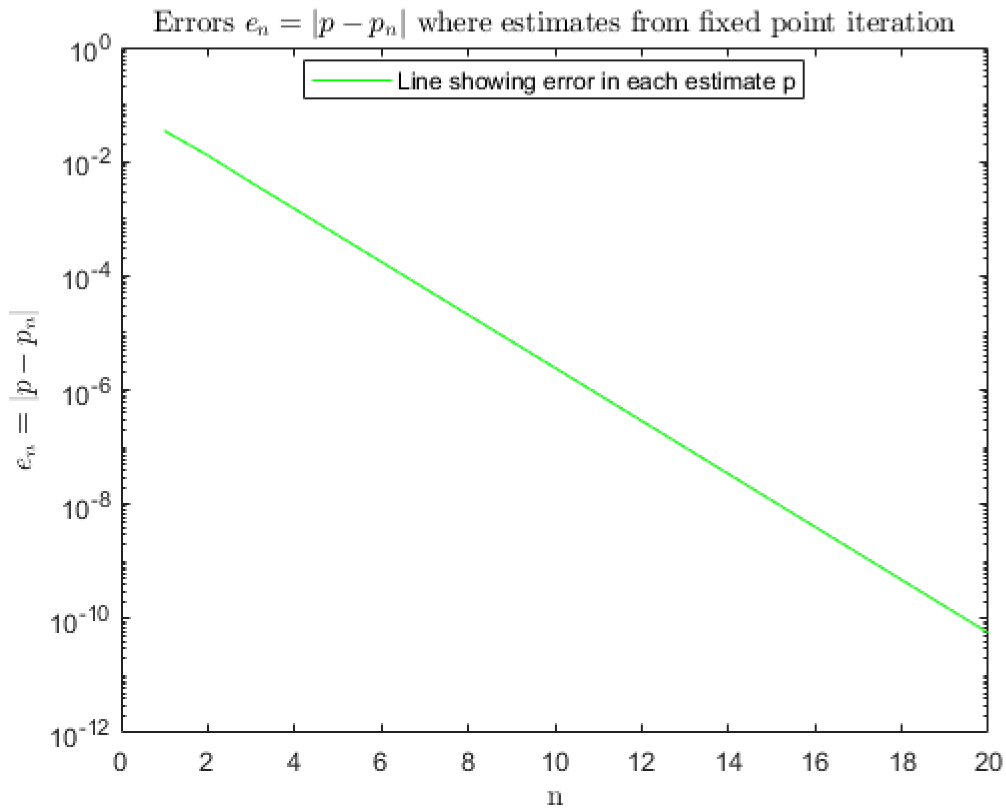
```
ans =
```

```
20×2 table

  n          p_vec
 ___    ________________

  1                1.375
  2       1.327392578125
  3     1.34454640110744
  4     1.33876042743645
  5     1.34076145123132
  6     1.34007511935904
  7     1.34031120352173
  8      1.3402300753892
  9     1.34025796378024
 10     1.34024837805856
 11      1.3402516729687
 12     1.34025054042139
 13     1.34025092970955
 14     1.34025079590055
 15     1.34025084189439
 16     1.34025082608504
 17     1.34025083151915
 18      1.3402508296513
 19     1.34025083029333
 20     1.34025083007265
```

Question 6

Following code creates a corresponding figure displaying the error convergence

```
e=abs(1.3402508301291-p_vec); %again creates a vector e representing
 the error in each estimate
set(groot,'DefaultTextInterpreter','latex')
semilogy(N,e,'g')%produces logarithmic plot with y axis in base 10 of
 errors from estimates of the fixed point of g against n
title('Errors $e_n=|p-p_n|$ where estimates from fixed point
 iteration');%rest of this code formats the graph
xlabel('n');
ylabel('$e_n=|p-p_n|$');
legend('Line showing error in each estimate p', 'Location', 'Best');
```



Question 7

Answer is written as comments: Theoretical answer: We denote pn as the n-th estimate, pn+1 as the (n+1)st estimate, and en,en+1 as their respective errors, with p being the true value of the fixed point of g. Of course we maintain the relationship that (pn+1)=g(pn). Then (pn)=p-(en) and (pn+1)=p-(en+1). We assume that en is small and expand pn via a Taylor series about p: g(pn)=(pn+1)=p-(en+1)= g(p)-(en)*g'(p)+...larger powers of en which we neglect due to the assumption that en is small. Now since p=g(p) from the definition of a fixed point of g, we can subtract p from both sides. This leaves us with the simple equation (neglecting terms with higher powers of en and multiplying by -1): (en+1) (approx.)=g'(p)*(en) where g' is the derivative of g. If we assume the initial error is e0 then by this equation e1=g'(p)*e0 and by repeated use of the equation inductively to n we have en=([g'(p)]^n)*e0. It's clear that the error reduces to zero as n approaches infinity when the absolute value of g'(p) is less than 1, and if our error approaches zero then we have convergence to the true value of the fixed point p. Thus we can find cmax in the following way: g(x)=x-c*(x^3+2*x^2-6). Differentiating: g'(x)=1-c*x*(3*x+4). Now using the approximate solution p given to us in the first part (p=1.3402508301291) and noting that since we are given c>0, g'(p) is so we only need to solve g'(p) -1. 1-c*p*(3*p+4)>-1. Rearranging: c<2/[p*(3*p+4)] which is approximately 0.1860496237. Therefore my approximate value for cmax is 0.186. this value is close to the true value of cmax, differences arise from the error of the approximate value of p given to us in part 1 and the fact that we neglected terms with higher power in the Taylor expansion above. To check my cmax, I used the code below.

```
% Code that illustrates the answer can be found below:
for c=0:0.000001:0.2 %we want to test over a range of c with very
 small differences between each c so our cmax is precise
   gprime=@(x) 1-c*x*(3*x +4);%we define this function which is the
 derivative of the g given in the question
```

```
    if gprime(1.3402508301291)<-1 %this condition is taken from the
 theoretical answer above assuming p is approximately 1.3402508301291
        disp(['Max value of c in the specified interval
 is:',num2str(c,5)]);%if the above condition not satisfied the
 sequence diverges
    break %the for loop stops as soon as a value of c is found so only
 cmax is displayed in the command window

    end
end
```

Max value of c in the specified interval is:0.18605

Question 8

See the file newtonMeth.m

Question 9

Following code uses Newton's method to compute approximations to the root of f and produces a table and figure illustrating its convergence

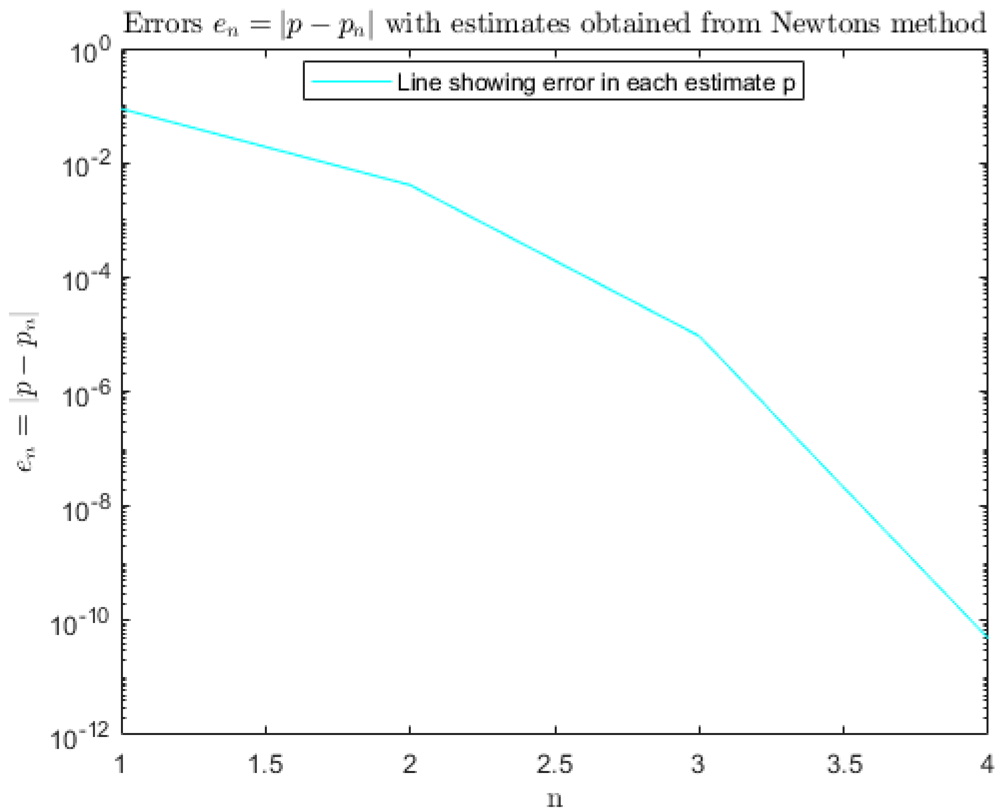
```
f=@(x) x.^3+2*x.^2-6; %defines the function f
df=@(x) 3*x.^2+4*x; %defines the first derivative of f
format long g
[p_vec,fp_vec]=newtonMeth(f,df,1e-9,20,1); %newtonMeth used to obtain
 vectors p_vec/fp_vec
L=length(p_vec); %gives the length of the column vector p_vec
N=1:L; %produces row vector with equivalent length to p_vec
n=N'; %makes column vector n from N so there is dimensional
 consistency for the table
table(n,p_vec,fp_vec) %produces the table required
e=abs(1.3402508301291-p_vec); %defines the absolute error with the
 estimate given
set(groot,'DefaultTextInterpreter','latex')
semilogy(N,e,'c') %plots a logarithmically scaled graph of error
 against iteration number n
title('Errors $e_n=|p-p_n|$ with estimates obtained from Newtons
 method'); %remaining code formats the graph
xlabel('n');
ylabel('$e_n=|p-p_n|$');
legend('Line showing error in each estimate p', 'Location', 'Best');
```

ans =

4×3 table

n	*p_vec*	*fp_vec*
1	*1.42857142857143*	*0.997084548104956*
2	*1.34433497536946*	*0.0440043226723521*
3	*1.34026014241717*	*0.000100105944426332*

```
    4     1.34025083017767     5.22106802236522e-10
```

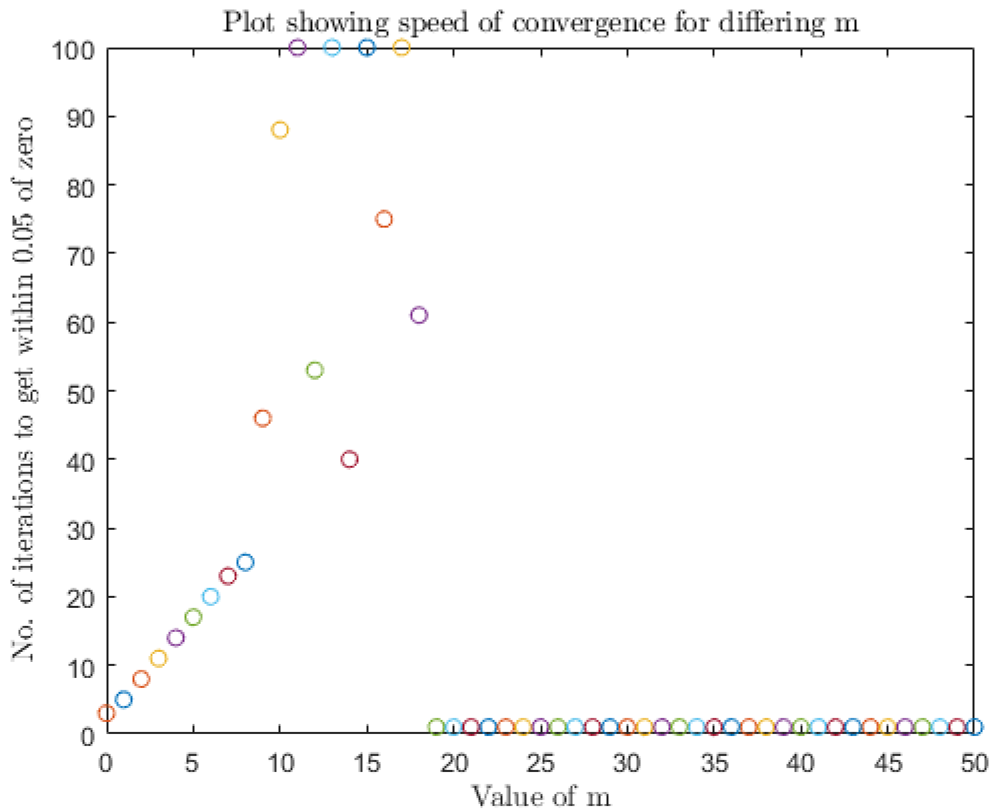


Question 10

Answer is written as comments: f(x)=exp(x)-1-x-...-(1/factorial(m))*x^m Firstly, notice that the terms subtracted from exp(x) define the maclaurin series for exp(x). Due to this, we can conclude that if a root exists with an initial guess of p0=1, then this root should be zero as the maclaurin series is centred at x=0. Therefore when we investigate convergence we could look at a stopping criterion of abs(f(p))<tolerance. As m goes to infinity, f(x) approaches zero for all x, but for smaller m (perhaps 50<m<100) its likely that convergence is immediate in the first estimate using a Newton method due to the tolerance used. To test these ideas the following code was used to produce a graph demonstrating the speed of convergence for differing m. The plot shows that for m in [1,8], convergence somewhat linearly slows down. This linearity is broken at m=9 which took 46 iterations to produce an estimate within 0.05 of zero. There is no visible correlation between the number of iterations (and so speed of convergence) of even m values between 9 and 17 whereas m=11,13,15,17 do not produce estimates within 0.05 of zero in 100 iterations so convergence is extremely slow if it occurs at all. For m>17, the stopping criterion is met immediately due to the tolerance used and the fact that f(x) approaches constant 0 for all x.

```
% Code that illustrates the answer can be found below:
for mmax=1:50 %sets the number of f's we'd like to investigate
    p0=1; %resets the initial guess p0 for each iteration
for n=1:100 %this is the maximum number of estimates we want to
 compute
    fx=exp(p0)-1; %we define fx which represents the function f(x) to
 go into the next for loop
```

```
    dfx=exp(p0); %we then define dfx representing the derivative of
 f(x) for the for loop
for m=1:mmax %this loop produces the information we need to find the
 next estimate p for a specific f(x)
    term=p0^m/factorial(m); %creates the m-th term at p0
    fx=fx-term; %subtracts the m-th term from fx
    dfterm=p0^(m-1)/factorial(m-1);%next two lines give us the value
 of derivative of f at p0
    dfx=dfx-dfterm;
end
    p=p0-fx/dfx;%computes the next estimate p using Newton function
    p0=p;%sets new p0 as the next estimate p for next iteration
    if abs(p)<0.05 %stopping criterion
        break %for loop broken when the estimate p gets below 0.05
 (tolerance)
    end
end
plot(mmax,n,'o');%plots a point for the function with mmax terms
 showing how many iterations it took for abs(p)<0.05 or if never
 achieves this in 100 iterations, n=100
hold all %allows all the points to be plotted in the same figure
end
plot(0,3,'o');%plots the point on the figure corresponding to m=0,
 which satisfies our stopping criterion when n=3
xlabel('Value of m');%rest of the code formats the graph
ylabel('No. of iterations to get within 0.05 of zero');
title('Plot showing speed of convergence for differing m');
axis([0,50,0,100])
```



Published with MATLAB® R2019a