MMA307

Laboratory work 2

Magnus Sörensen and Wicktor Löw

20161023

1 Quadratic and linear convergence of newton Raphson method.

1.1 Description - Problem 1

Consider the function 1 interval $[0, 3]$

$$f(x) = 12 - 26x + 20x^2 - 7x^3 - 12e^{x-2} + 12xe^{x-2} \tag{1}$$

1. Plot the function to get some ideas of the zeroes and choose suitable initial guesses.
2. Apply the Newton–Raphson method to find both zeroes of the function $f(x)$.
3. For each root print out:
   - The sequence of iterates.
   - The absolute error $|E_i|$.
   - The absolute relative error ratio $\frac{|E_{i+1}|}{|E_i|}$ and $\frac{|E_{i+1}|}{(|E_i|)^2}$ that converges to a nonzero limit.

Each of those print out can be found in the Results part in 1.4 on page 5.

1.2 Description - Problem 2

1. Determine for which root the convergence is quadratic, and also compute the asymptotic error constant A.
2. If for one of the roots the method does not show quadratic convergence, explain why it doesn't, i.e. why the theorem on quadratic convergence isn't applicable.
3. Does it obey linear convergence? If yes, why?

1.3 Solution

```
%show graph
plt_func

f = @(x) 12 - 26.*x + 20.*x.^2 - 7.*x.^3 -12.*exp(x-2) + 14 .*x.*exp(x-2)
% Settings
approx1 = 0.2 % Approximation for f(x) = 0 poor guess to get many rep.
approx2 = 1.9
tol1 = 1e-9
tol2 = 1e-9
rep = 20
% Derivitive.
df = @(x) -21.*x.^2+40.*x+14.*x.*exp(x-2)+2.*exp(x-2)-26;
ddf = @(x) 14.*x.*exp(x-2)-42.*x + 16.*exp(x-2)+40;
% Find error with nownon.
xn1 = newton(f, df, approx1, tol1, tol2, rep);
xn2 = newton(f, df , approx2, tol1, tol2, rep);
% Write data to disk.
t_writer(xn1, xn2);
% Roots
root = [0.857142857142857 2.0]
root(1) = fzero(f, 0.8);
% Calculate error
[Ei1 Er1 Er12 ] = calcerror(xn1, root(1));
[Ei2 Er2 Er22 ] = calcerror(xn2, root(2));
% Save the error to disk
t_writer2('table_t1.dat', Ei1, Er1,Er12)
t_writer2('table_t2.dat', Ei2, Er2,Er22)

A1 = Er12(end)
A2 = Er2(end)
M = abs(ddf(root(1)))/abs((2*df(root(1))))
```

Listing 1: Solution

```
function [Ei Er Er2] = calcerror(approx_list, value)
    len = length(approx_list);
    %Ei = zeros(len);
    %Er = zeros(len);
    %Er2 = zeros(len);
    for i = 1:len
        Ei(i) = abs(value - approx_list(i));
    end

    for i = 1:len-2
            Er(i) = Ei(i+1)/(Ei(i));
            Er2(i) = Ei(i+1)/((Ei(i)).^2);
    end


end
```

Listing 2: Calculate error

```
 function  [x] = newton(f,df,x0,tol1,tol2,iterNr)

 % The function newton.m solves a nonlinear
 % equation f(x) = 0 using the Newton-Raphson method

 % Inputs:
 %       f - function handle, e.g. f = @(x) exp(-x)
 %       df - derivative of f, also a function handle
 %       x0 - initial guess
 %       tol1 - tolerance for x(k)
 %       tol2 - tolerance for the function values y
 %       iterNr - maximum number of iterations
 % Output:
 %       x - sequence of root estimates
   %df = @(x) cdd(f, x);

 x(1)=x0;
    for k=2:iterNr
      x(k)=x(k-1)-f(x(k-1))/df(x(k-1));
         err=abs(x(k)-x(k-1));
         y=f(x(k));
         if (err<tol1)||(abs(y)<tol2)
                break;
         end
     end
end
```

Listing 3: Newton

```
function t_writer(t1, t2)
% Function to write 2 tables to to disk
fileID = fopen('iterations.dat', 'w');
lenMax = max(length(t1) , length(t2));
str = sprintf(',x1, x2\n');
fprintf(fileID,str);
for i=1:lenMax
     if (i <= length(t1)) & (i <= length(t2))
         str = sprintf('%d,%d,%d\n',i,t1(i),t2(i));
     elseif (i > length(t1)) & (i <= length(t2))
          str = sprintf('%d,,%d\n',i,t2(i));
     elseif (i <= length(t1)) & (i > length(t2))
          str = sprintf('%d,%d,\n',i,t1(i));
     end
     fprintf(fileID, str);
end
fclose(fileID);

end
```

Listing 4: Wrtite to disk1

```
function t_writer2(filename, absT, relT, rel2T)
% Writes a table to disk
%Input:
```

```
%   filename
%   absT = a matrix of absolut error.
%   relT = a matrix of relative errors.
%   rel2T = a matrix of square relative erros
    fileID = fopen(filename, 'w');
    lenMax = max(length(absT), length(relT));
    lenMax = max(lenMax, length(rel2T));

    str = sprintf(',abs,rel,rel2\n');
    fprintf(fileID, str);
    for i = 1:lenMax
        n = sprintf('%d,', i)
        if (i <= length(absT))
            a = sprintf('%d,',absT(i));
        else
            a = ',';
        end

        if (i <= length(relT))
            b = sprintf('%d,',relT(i));
        else
            b = ',';
        end

        if (i <= length(rel2T))
            c = sprintf('%d',rel2T(i));
        else
            c = '';
        end
        d = '\n'
        row = strcat(n,a,b,c,d);
        fprintf(fileID, row);
    end
    fclose(fileID);
end
```

Listing 5: Write to disk2

1.4 Results

1.4.1 Table with iterations.

	x1	x2
1	2.000000e-01	1.900000e+00
2	5.337616e-01	1.934728e+00
3	7.363210e-01	1.957057e+00
4	8.320773e-01	1.971612e+00
5	8.557439e-01	1.981179e+00
6	8.571382e-01	1.987498e+00
7	8.571429e-01	1.991685e+00
8		1.994465e+00
9		1.996314e+00
10		1.997544e+00
11		1.998364e+00
12		1.998909e+00
13		1.999273e+00
14		1.999515e+00

Table 1: Table with iterations

1.4.2 Plot of $f(x)$

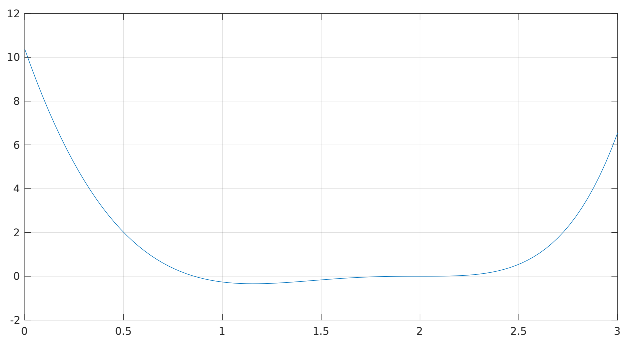

Figure 1: Plot of figure

1.4.3 Tables with absolute and relative error ratio

	abs	rel	rel2
1	6.571429e-01	4.921019e-01	7.488508e-01
2	3.233813e-01	3.736204e-01	1.155356e+00
3	1.208218e-01	2.074585e-01	1.717061e+00
4	2.506552e-02	5.581234e-02	2.226658e+00
5	1.398965e-03	3.361128e-03	2.402582e+00
6	4.702102e-06		
7	5.336898e-11		

Table 2: Table for root 1.

	abs	rel	rel2
1	1.000000e-01	6.527153e-01	6.527153e+00
2	6.527153e-02	6.579172e-01	1.007970e+01
3	4.294326e-02	6.610504e-01	1.539358e+01
4	2.838766e-02	6.630116e-01	2.335563e+01
5	1.882135e-02	6.642677e-01	3.529331e+01
6	1.250241e-02	6.650836e-01	5.319642e+01
7	8.315149e-03	6.656184e-01	8.004888e+01
8	5.534716e-03	6.659709e-01	1.203261e+02
9	3.685960e-03	6.662042e-01	1.807410e+02
10	2.455602e-03	6.663589e-01	2.713628e+02
11	1.636312e-03	6.664618e-01	4.072951e+02
12	1.090539e-03	6.665284e-01	6.111914e+02
13	7.268755e-04		
14	4.845176e-04		

Table 3: Table for root 2

1.5 Analysis of results

1.5.1 Problem 1

1. Fig 1 show the plot of $f(x)$ and we chose $x_1 = 0.2$ and $x_2 = 1.9$ as our initial guesses to approximate the roots for the function. To get a considerable amount of iterations we had to make "bad guesses", otherwise it would even for low tolerances yield only 2-3 iterations before finding an approximation of the root which was to little data to analyse.

2. Applying the Newton-Raphson method yields an solution for $x_1 = 0.857142857089488$ after 7 iterations and solution for $x_2 = 1.999515482374499$ after 14 iterations with a tolerance of $T_{tol} = 10^{-9}$.

3. The data is shown in the tables in 1.4.3.

1.5.2 Problem 2

1. From table1 we see that the root $x = 2$ seems to converge when$R = 1$ very steadily. For $x_1 \approx 0.857$ however with so few iterations it is hard to see any convergence, but we conclude that for $R = 2$ it seems to converge to a value. Our conclusion is that $x_2 = 2$ has a linear convergence and $x_1 \approx 0.857$ has a quadratic convergence. Using the formula 2we get $M = 2.413850894567771$ which correlates with the value x_1 seems to converge to. This is also the asymptotic error constant. For $x = 2, A = 0.666528351347989$ and for $x_1 = 0.857$, $A = 2.402581739389790$.

$$M = \frac{f''(x)}{2 * f(x)} \tag{2}$$

1. For the root x = 2 the method doesn't show quadratic convergence, this is because x = 2 is a root but also an extreme point for the function so the derivative at this point is zero which yields division by zero using the formula to compute using formula 2.

2. The formula which calculates the asymptotic error constant $\frac{E_{(i+1)}}{E_i} = A$ rewritten to compute the next absolute error is $E_{(i+1)} = A * E_i^R$. For $R = 1$ this is a linear equation, for $R = 2$ a quadratic equation and for $R = 3$ a cubic equation and so on. For our case the root $x = 2$ converges when $R = 1$ and therefore follows linear convergence.