Birla Institute of Technology and Science, Pilani-K K Birla Goa Campus

First Semester 2018-2019

Numerical Analysis
(MATH F313)
Assignment - 1

Note: Solve all the problems by implementing the corresponding method using MATLAB software.

1. Mechanical engineers, as well as most engineers use thermodynamics extensively in their work. The following polynomial can be used to relate the zero-pressure specific heat of dry air c_p to temperature

   $$c_p = 0.99403 + 1.671 \times 10^{-4}\, T + 9.7215 \times 10^{-8}\, T^2 - 9.5838 \times 10^{-11}\, T^3 + 1.9520 \times 10^{-14}\, T^4.$$

   Determine the temperature that corresponds to a specific heat of 1.1 using secant method.

2. The upward velocity of a rocket can be computed by the following formula:

   $$v = u \ln \frac{m_0}{m_0 - qt} - gt,$$

   where v = upward velocity, u = the velocity at which fuel is expelled relative to the rocket, m_0 = the initial mass of the rocket at time $t = 0$, q = the fuel consumption rate, and g = the downward acceleration of gravity (assumed constant = $9.81\, m/s^2$). If $u = 2000\, m/s, m_0 = 150000\, kg$, and $q = 2700\, kg/s$. Compute the time at which $v = 750\, m/s$ using bisection method.

3. Water is flowing in a channel at a rate of $Q = 20 m^3/s$. The critical depth y for such a channel must satisfy the equation

   $$1 - \frac{Q^2}{g\, A_c^3} B = 0,$$

   where $g = 9.81\, m/s^2$, A_c = the cross-sectional area (m^2), and B = the width of the channel at the surface (m). For this case, the width and cross-sectional area can be related to depth y by

   $$B = 3 + y, \quad \text{and} \quad A_c = 3y + \frac{y^2}{2}.$$

   Solve for the critical depth using

   - The graphical method
   - Bisection method and
   - Method of false position.

   Use initial interval $[0.5, 2.5]$.

4. Use Newton-Raphson method to determine the mass m of the bungee jumper with a drag coefficient of $c_d = 0.25\, kg/m$ to have a velocity v of $36\, m/s$ after $4s$ of free fall. The acceleration of gravity is $9.81\, m/s^2$. The governing model is

   $$f(m) = \sqrt{\frac{g\, m}{c_d}} \tanh\left(\sqrt{\frac{g\, c_d}{m}} t\right) - v(t).$$

5. The volume of liquid V in a hollow horizontal cylinder of radius r and length L is related to the depth of the liquid h by

$$V = \left[r^2 \cos^{-1}\left(\frac{r-h}{r}\right) - (r-h)\sqrt{2rh - h^2} \right] L.$$

Determine h given $r = 2\,m, L = 5\,m^3$, and $V = 8.5\,m^3$ using regula-falsi method.

6. An oscillating current in an electric circuit is described by

$$I = 9e^{-t}\cos(2\pi t),$$

where t is in seconds. Determine all values of t such that $I = 3$ using secant method.

7. Use the Newton-Raphson method to find the root of

$$f(x) = e^{-0.5x}(4 - x) - 2.$$

Take initial guesses of (i) 2, (ii) 6, and (iii) 8. Explain your results.

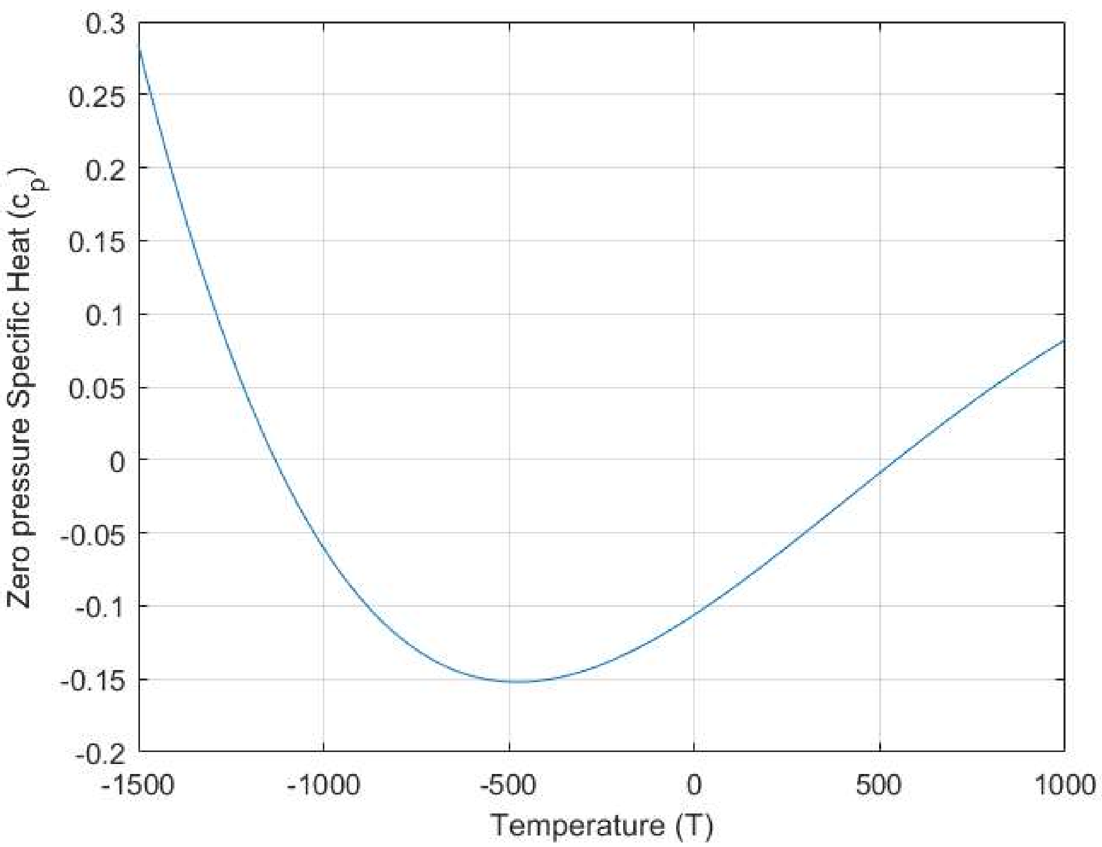
Contents


```
clear all;
close all;
```

Plotting graph of function

```
g = @(x) -1.1+0.99403+(1.671*10^-4)*x+(9.7215*10^-8)*x^2-(9.5838*10^-11)*x^3+(1.9520*10^-1
4)*x^4;
X = -1500:1000; Y = -1500:1000; n = 1;
for x = -1500:1000
Y(n) = g(x);
n = n + 1;
end
plot(X,Y); grid on;
xlabel('Temperature (T)');
ylabel('Zero pressure Specific Heat (c_p)');

fprintf("Considering the interval [-1500, -1000]\n"); SecantMethod(-1500,-1000);
fprintf("Considering the interval [400, 600]\n"); SecantMethod(400,600);
```

```
Considering the interval [-1500, -1000]
```



Secant Method applied with different intervals.

```
x0,x1   initial approximations to location of root
```

```
function y = SecantMethod(x0,x1)
    f = inline('-1.1 + 0.99403 + (1.671*10^-4)*x + (9.7215*10^-8)*x^2 - (9.5838*10^-11)*x^
3 + (1.9520*10^-14)*x^4','x');
    TOL=10^(-10); %  absolute error convergence tolerance
    Nmax=100;%   maximum number of iterations to be performed
    flag=0;
    older = x0;   old = x1;
    folder = feval(f,older);
    for i = 2 : Nmax
        fold = feval(f,old);
        dx = fold * ( old - older ) / ( fold - folder );
        new = old - dx;
        fprintf('\t\t %3d \t %.15f \n', i, new )
        if ( abs(dx) < TOL )
            flag=1;
            break
        else
            older = old;
            old = new;
            folder = fold;
        end
    end
    if flag == 0
        disp('Maximum number of iterations exceeded')
    end
end
```

```
                   2     -1087.706301249115800
                   3     -1138.029509542062000
                   4     -1130.693760893146000
                   5     -1131.025602676162600
                   6     -1131.028101492426900
                   7     -1131.028100596541900
                   8     -1131.028100596544200
Considering the interval [400, 600]
                   2     544.867997840782210
                   3     544.081498954654420
                   4     544.087538233961940
                   5     544.087537655509440
                   6     544.087537655508980
```

Published with MATLAB® R2017a

Contents

Bisection Method for finding zeros of a function

```
clear all;
close all;
```

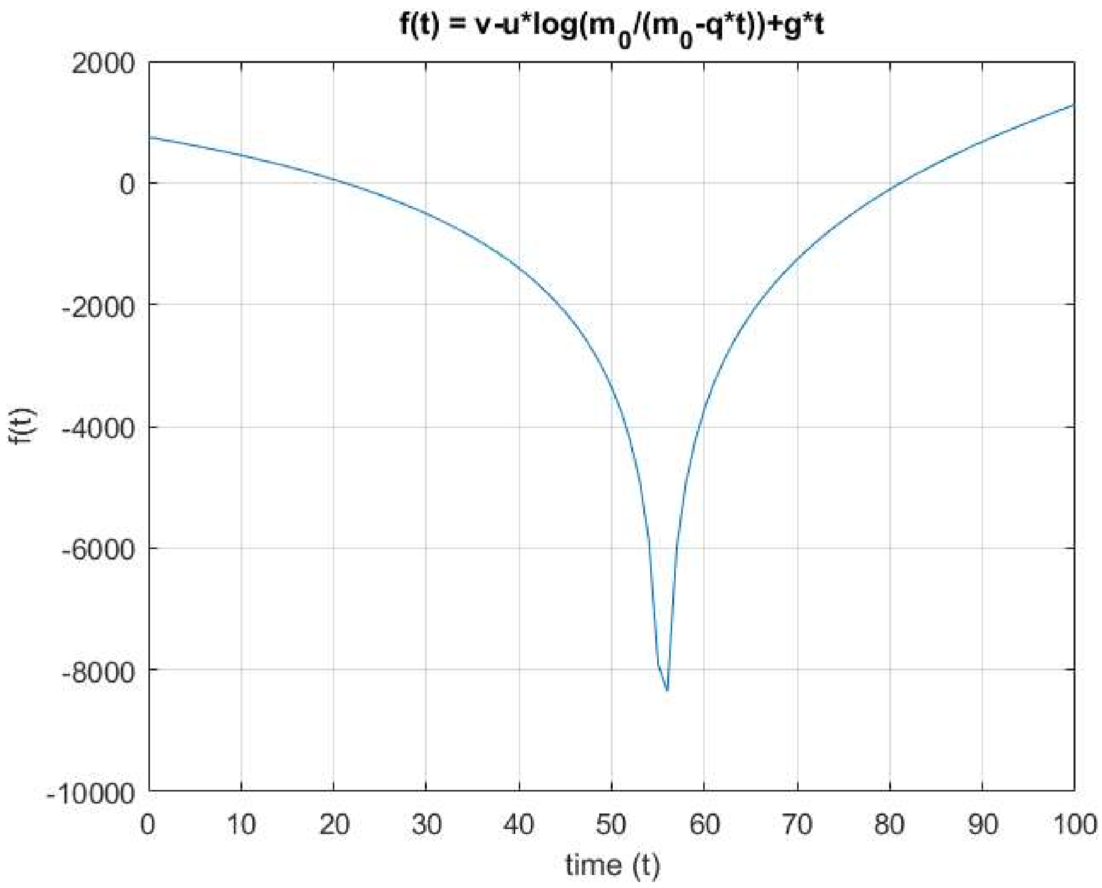
Function definition

```
u = 2000; v = 750; m_0 = 150000;
g = 9.81; q = 2700;
f = @(t) v-u*log(m_0/(m_0-q*t))+g*t;
a=10;  b=30;
```

Plotting function f(t)

```
X = 0:100;
Y = 0:100;
n = 1;
for x = 0:100
Y(n) = f(x);
n = n + 1;
end
plot(X,Y); grid on;
xlabel("time (t)");
ylabel("f(t)");
title("f(t) = v-u*log(m_0/(m_0-q*t))+g*t");
```

```
Warning: Imaginary parts of complex X and/or
Y arguments ignored
```



Stopping criterium

```
TOL = 10^(-6);
Nmax = floor ( log((b-a)/TOL) / log(2) ) + 1
pvalues=zeros(Nmax,1);
```

```
Nmax =

    25
```

Main loop

```
for i = 1 : Nmax
    p = (a+b)/2;
        pvalues(i)=p;
    sfa = f(a);
    sfp = sign(f(p));
    fprintf( '\t\t %3d \t (%.6f,%.6f) \t %.10f \n', i, a, b, p)
        if ( (b-a)<2*TOL || sfp == 0 )
           break
        elseif ( sfa * sfp < 0 )
           b = p;
        else
           a = p;
           sfa = sfp;
    end
end
```

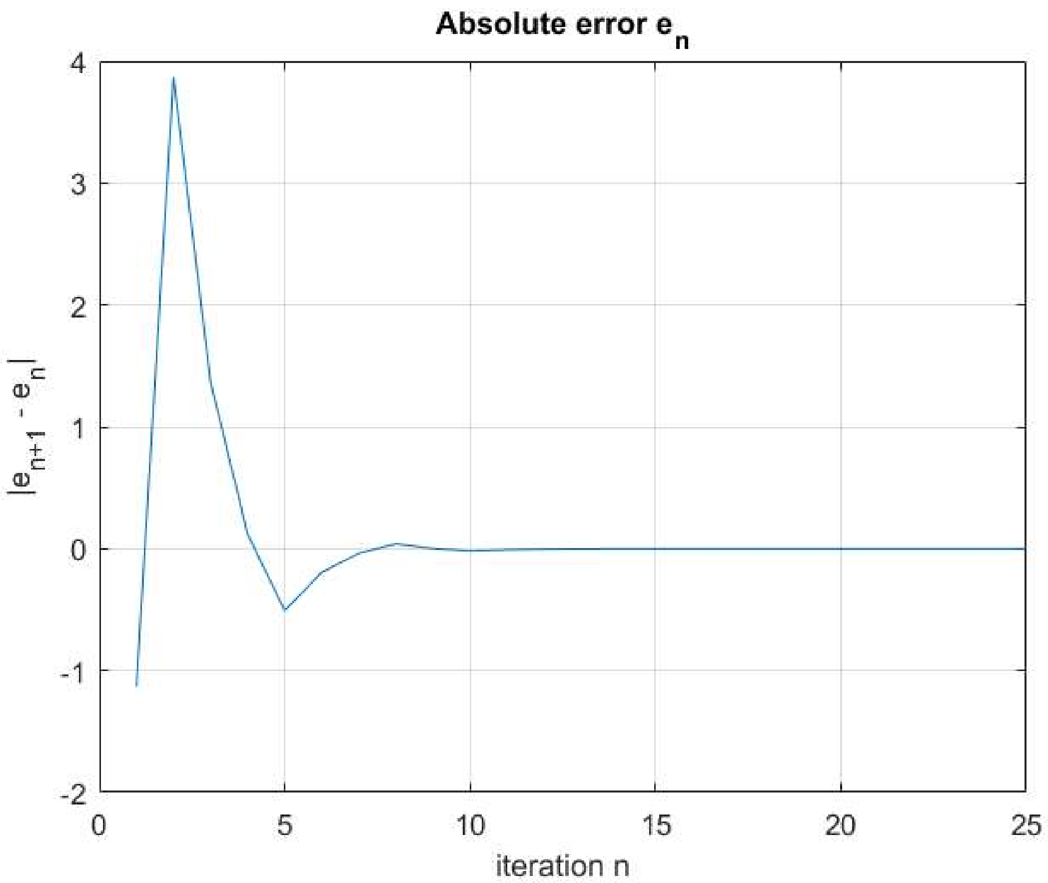
```
   1      (10.000000,30.000000)   20.0000000000
   2      (20.000000,30.000000)   25.0000000000
   3      (20.000000,25.000000)   22.5000000000
   4      (20.000000,22.500000)   21.2500000000
   5      (20.000000,21.250000)   20.6250000000
   6      (20.625000,21.250000)   20.9375000000
   7      (20.937500,21.250000)   21.0937500000
   8      (21.093750,21.250000)   21.1718750000
   9      (21.093750,21.171875)   21.1328125000
  10      (21.093750,21.132813)   21.1132812500
  11      (21.113281,21.132813)   21.1230468750
  12      (21.123047,21.132813)   21.1279296875
  13      (21.127930,21.132813)   21.1303710938
  14      (21.130371,21.132813)   21.1315917969
  15      (21.131592,21.132813)   21.1322021484
  16      (21.132202,21.132813)   21.1325073242
  17      (21.132202,21.132507)   21.1323547363
  18      (21.132355,21.132507)   21.1324310303
  19      (21.132355,21.132431)   21.1323928833
  20      (21.132393,21.132431)   21.1324119568
  21      (21.132412,21.132431)   21.1324214935
  22      (21.132412,21.132421)   21.1324167252
  23      (21.132412,21.132417)   21.1324143410
  24      (21.132414,21.132417)   21.1324155331
  25      (21.132414,21.132416)   21.1324149370
```

Absolute Error computation

```
plast =p;
errors=pvalues-plast*ones(Nmax,1);
fprintf('Approximate value |  Absolute Error\n\n')
fprintf(' %.10f     |  %.10f\n',[pvalues errors]');

figure; plot(errors); grid on;
xlabel("iteration n");
ylabel("|e_{n+1} - e_n|");
title('Absolute error e_n')
```

```
Approximate value |  Absolute Error

 20.0000000000     |  -1.1324149370
 25.0000000000     |  3.8675850630
 22.5000000000     |  1.3675850630
 21.2500000000     |  0.1175850630
 20.6250000000     |  -0.5074149370
 20.9375000000     |  -0.1949149370
 21.0937500000     |  -0.0386649370
 21.1718750000     |  0.0394600630
 21.1328125000     |  0.0003975630
 21.1132812500     |  -0.0191336870
 21.1230468750     |  -0.0093680620
 21.1279296875     |  -0.0044852495
 21.1303710938     |  -0.0020438433
 21.1315917969     |  -0.0008231401
 21.1322021484     |  -0.0002127886
 21.1325073242     |  0.0000923872
 21.1323547363     |  -0.0000602007
 21.1324310303     |  0.0000160933
```

```
21.1323928833      |  -0.0000220537
21.1324119568      |  -0.0000029802
21.1324214935      |  0.0000065565
21.1324167252      |  0.0000017881
21.1324143410      |  -0.0000005960
21.1324155331      |  0.0000005960
21.1324149370      |  0.0000000000
```



Contents

Bisection Method for finding zeros of a function

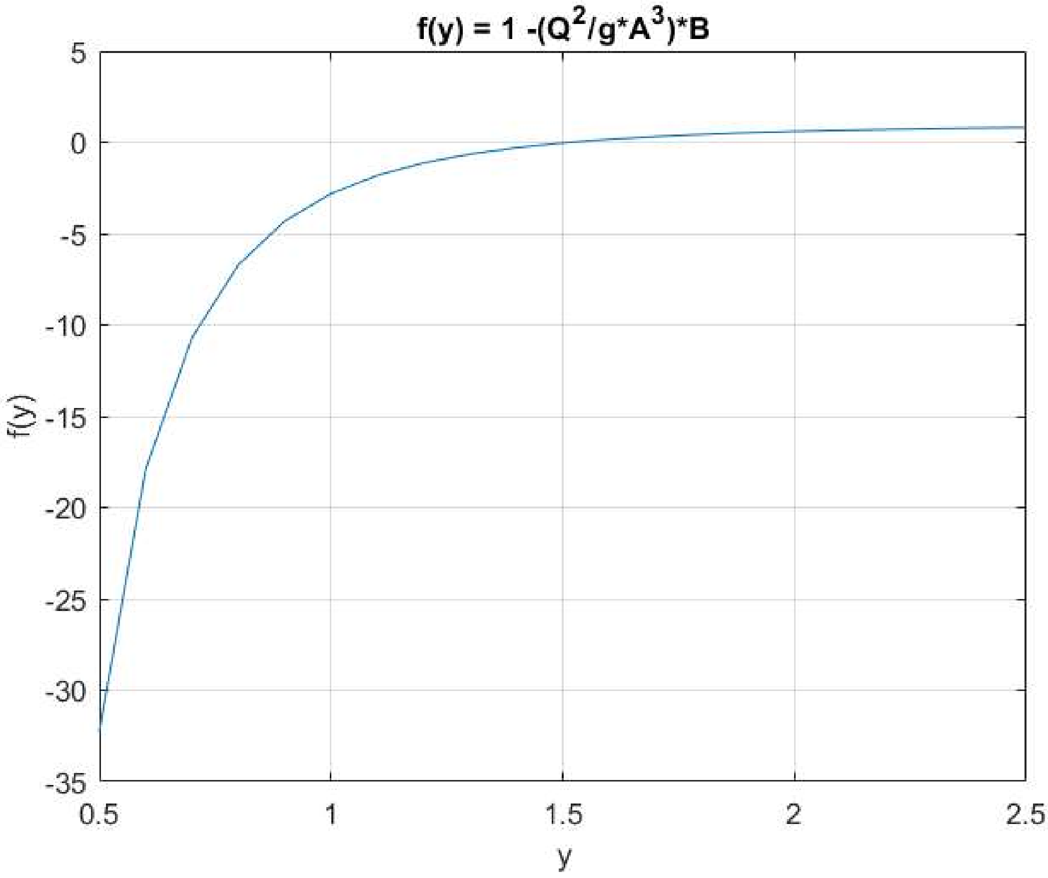
```
clear all;
close all;
```

Function definition

```
Q = 20; g = 9.81;
A = @(y) 3*y + y^2/2;
B = @(y) 3 + y;
f = @(y) 1 -(Q^2/(g*(A(y)^3)))*B(y);
a=0.5;  b=2.5;
```

Plotting graph of f(y)

```
X = a:0.1:b;
Y = a:0.1:b;
n = 1;
for x = a:0.1:b
Y(n) = f(x);
n = n + 1;
end
plot(X,Y); grid on;
xlabel("y");
ylabel("f(y)");
title("f(y) = 1 -({Q^2}/{g*A^3})*B")
```



Stopping criterium

```
TOL = 10^(-6);
Nmax = floor ( log((b-a)/TOL) / log(2) ) + 1
pvalues=zeros(Nmax,1);
```

```
Nmax =

    21
```

Iteration Scheme

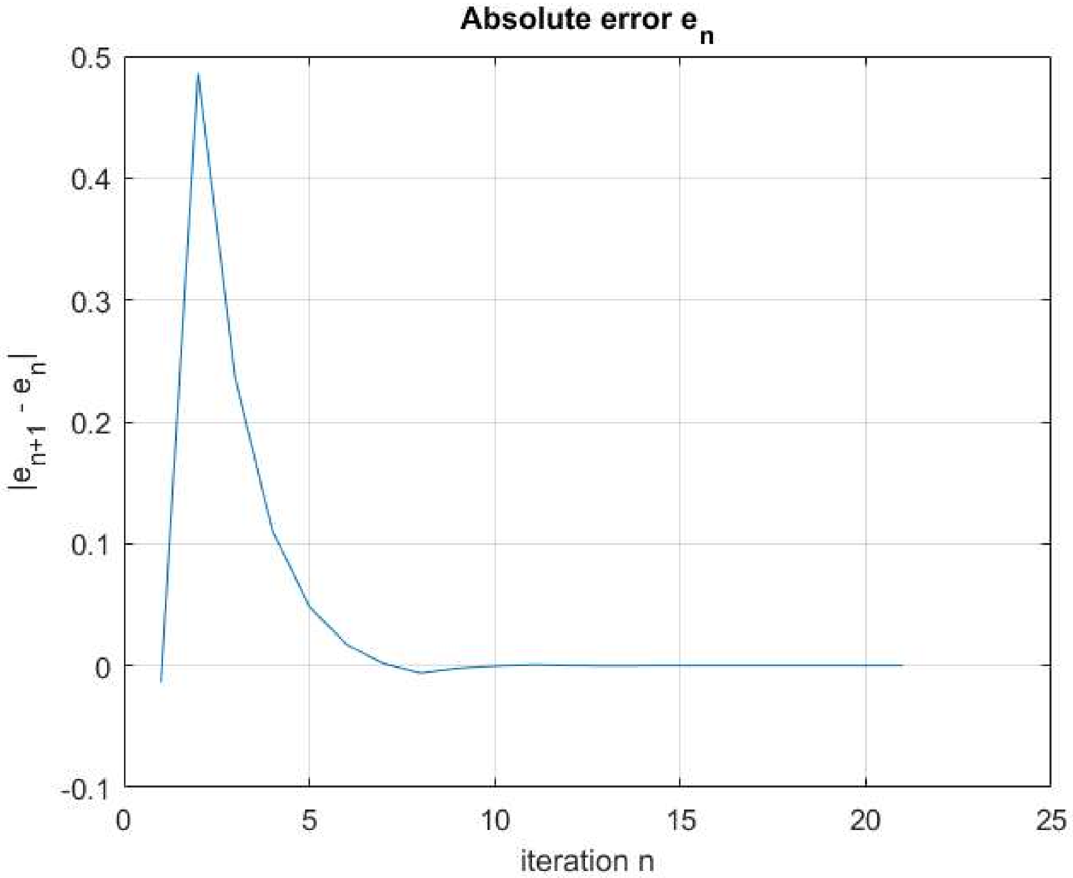
```
for i = 1 : Nmax
    p = (a+b)/2;
        pvalues(i)=p;
    sfa = f(a);
    sfp = sign(f(p));
    fprintf( '\t\t %3d \t (%.6f,%.6f) \t %.10f \n', i, a, b, p)
        if ( (b-a)<2*TOL || sfp == 0 )
           break
        elseif ( sfa * sfp < 0 )
           b = p;
        else
           a = p;
           sfa = sfp;
    end
end
```

```
 1     (0.500000,2.500000)     1.5000000000
 2     (1.500000,2.500000)     2.0000000000
 3     (1.500000,2.000000)     1.7500000000
 4     (1.500000,1.750000)     1.6250000000
 5     (1.500000,1.625000)     1.5625000000
 6     (1.500000,1.562500)     1.5312500000
 7     (1.500000,1.531250)     1.5156250000
 8     (1.500000,1.515625)     1.5078125000
 9     (1.507813,1.515625)     1.5117187500
10     (1.511719,1.515625)     1.5136718750
11     (1.513672,1.515625)     1.5146484375
12     (1.513672,1.514648)     1.5141601563
13     (1.513672,1.514160)     1.5139160156
14     (1.513916,1.514160)     1.5140380859
15     (1.514038,1.514160)     1.5140991211
16     (1.514038,1.514099)     1.5140686035
17     (1.514038,1.514069)     1.5140533447
18     (1.514053,1.514069)     1.5140609741
19     (1.514053,1.514061)     1.5140571594
20     (1.514053,1.514057)     1.5140552521
21     (1.514053,1.514055)     1.5140542984
```

Absolute Error computation

```
plast =p;
errors=pvalues-plast*ones(Nmax,1);
fprintf('Approximate value |  Absolute Error\n\n')
fprintf(' %.10f     |  %.10f\n',[pvalues errors]');

figure; plot(errors); grid on;
xlabel("iteration n");
ylabel("|e_{n+1} - e_n|");
title('Absolute error e_n')
```

```
Approximate value |  Absolute Error

 1.5000000000     |  -0.0140542984
 2.0000000000     |  0.4859457016
 1.7500000000     |  0.2359457016
 1.6250000000     |  0.1109457016
 1.5625000000     |  0.0484457016
 1.5312500000     |  0.0171957016
 1.5156250000     |  0.0015707016
 1.5078125000     |  -0.0062417984
 1.5117187500     |  -0.0023355484
 1.5136718750     |  -0.0003824234
 1.5146484375     |  0.0005941391
 1.5141601563     |  0.0001058578
 1.5139160156     |  -0.0001382828
 1.5140380859     |  -0.0000162125
 1.5140991211     |  0.0000448227
 1.5140686035     |  0.0000143051
 1.5140533447     |  -0.0000009537
 1.5140609741     |  0.0000066757
 1.5140571594     |  0.0000028610
 1.5140552521     |  0.0000009537
 1.5140542984     |  0.0000000000
```



Published with MATLAB® R2017a

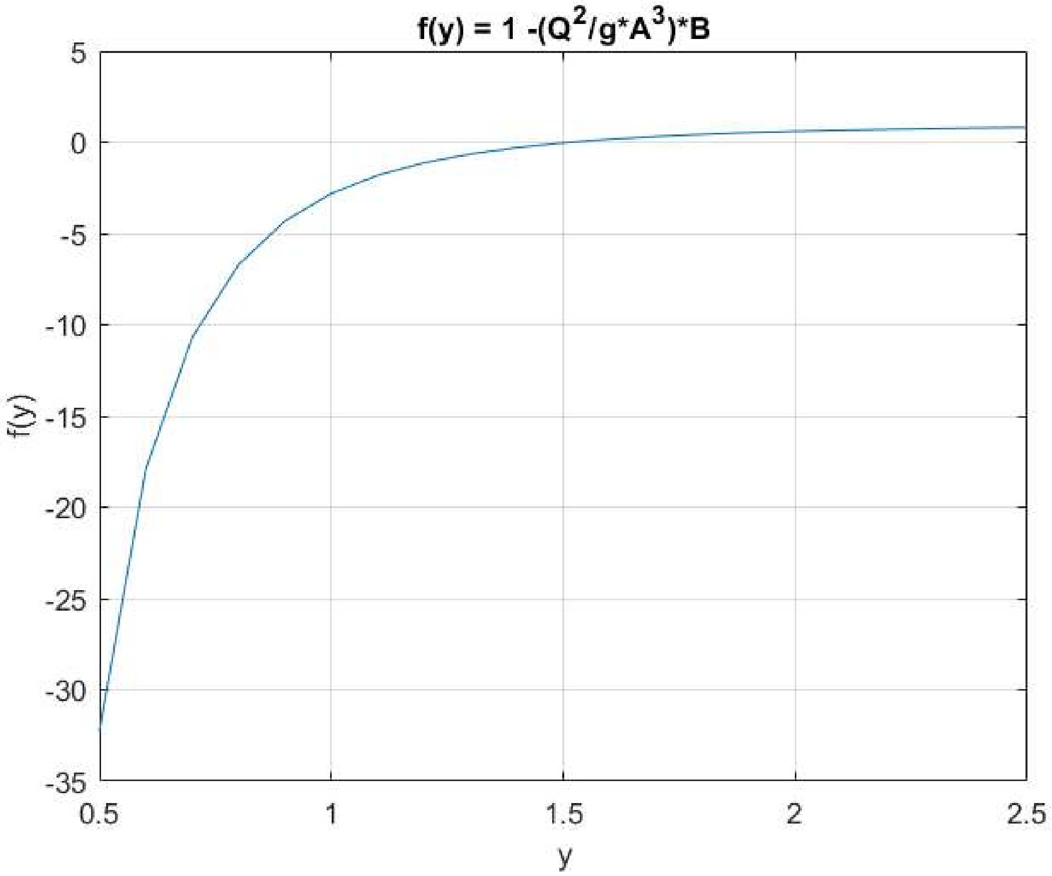
Contents

Function defination

```
Q = 20; g = 9.81;
A = @(y) 3*y + y^2/2;
B = @(y) 3 + y;
f = @(y) 1 -(Q^2/(g*(A(y)^3)))*B(y);
a=0.5;  b=2.5;
```

Plotting graph of f(y)

```
X = a:0.1:b;
Y = a:0.1:b;
n = 1;
for x = a:0.1:b
Y(n) = f(x);
n = n + 1;
end
plot(X,Y); grid on;
xlabel("y");
ylabel("f(y)");
title("f(y) = 1 -({Q^2}/{g*A^3})*B")
```



Stopping criterium

```
TOL=10^(-6);
format long;
old = b;
fa = feval(f,a);
fb = feval(f,b);
Nmax = 100;
```

Iteration scheme

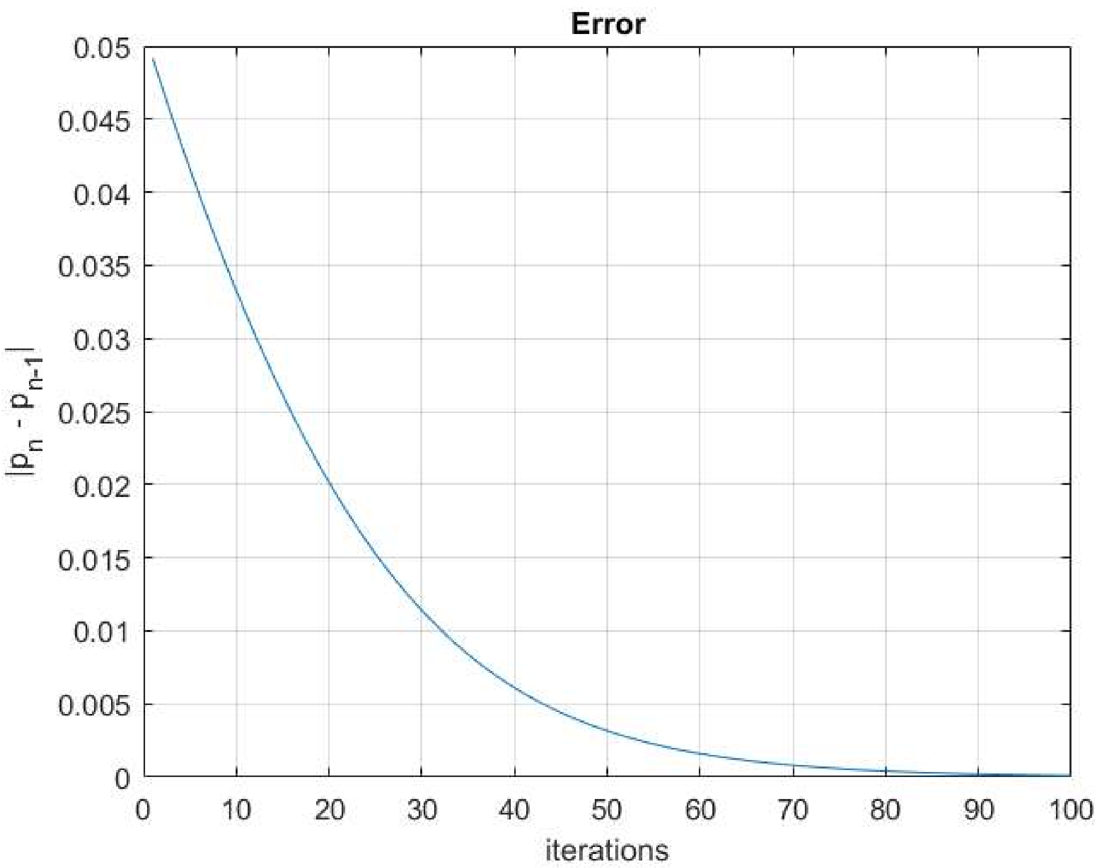
```
pvalues=[]; flag =0;
for i = 1 : Nmax
        new = b - fb * ( b - a ) / ( fb - fa );
        fnew = feval(f,new);
        fprintf ( '\t\t %3d \t (%.10f,%.10f) \t %.10f \n', i, a, b, new )

        if ( abs(new-old) < TOL )
        flag=1;
           break
        elseif ( fa * fnew < 0 )
           b = new;
           fb = fnew;
        else
           a = new;
           fa = fnew;
    end
    error(i) = abs(new-old);
        old = new;
        pvalues = [pvalues;old];

end
```

1	(0.5000000000,2.5000000000)	2.4508314769
2	(0.5000000000,2.4508314769)	2.4036291706
3	(0.5000000000,2.4036291706)	2.3583419190
4	(0.5000000000,2.3583419190)	2.3149191727
5	(0.5000000000,2.3149191727)	2.2733109175
6	(0.5000000000,2.2733109175)	2.2334676049
7	(0.5000000000,2.2334676049)	2.1953400929
8	(0.5000000000,2.1953400929)	2.1588795962
9	(0.5000000000,2.1588795962)	2.1240376472
10	(0.5000000000,2.1240376472)	2.0907660677
11	(0.5000000000,2.0907660677)	2.0590169517
12	(0.5000000000,2.0590169517)	2.0287426587
13	(0.5000000000,2.0287426587)	1.9998958179
14	(0.5000000000,1.9998958179)	1.9724293424
15	(0.5000000000,1.9724293424)	1.9462964529
16	(0.5000000000,1.9462964529)	1.9214507099
17	(0.5000000000,1.9214507099)	1.8978460542
18	(0.5000000000,1.8978460542)	1.8754368537
19	(0.5000000000,1.8754368537)	1.8541779556
20	(0.5000000000,1.8541779556)	1.8340247435
21	(0.5000000000,1.8340247435)	1.8149331968
22	(0.5000000000,1.8149331968)	1.7968599526
23	(0.5000000000,1.7968599526)	1.7797623671
24	(0.5000000000,1.7797623671)	1.7635985779
25	(0.5000000000,1.7635985779)	1.7483275630
26	(0.5000000000,1.7483275630)	1.7339091977
27	(0.5000000000,1.7339091977)	1.7203043082
28	(0.5000000000,1.7203043082)	1.7074747200
29	(0.5000000000,1.7074747200)	1.6953833021
30	(0.5000000000,1.6953833021)	1.6839940055
31	(0.5000000000,1.6839940055)	1.6732718953
32	(0.5000000000,1.6732718953)	1.6631831780
33	(0.5000000000,1.6631831780)	1.6536952215
34	(0.5000000000,1.6536952215)	1.6447765697
35	(0.5000000000,1.6447765697)	1.6363969518
36	(0.5000000000,1.6363969518)	1.6285272841
37	(0.5000000000,1.6285272841)	1.6211396688
38	(0.5000000000,1.6211396688)	1.6142073856
39	(0.5000000000,1.6142073856)	1.6077048800
40	(0.5000000000,1.6077048800)	1.6016077471
41	(0.5000000000,1.6016077471)	1.5958927111
42	(0.5000000000,1.5958927111)	1.5905376021
43	(0.5000000000,1.5905376021)	1.5855213296
44	(0.5000000000,1.5855213296)	1.5808238536
45	(0.5000000000,1.5808238536)	1.5764261538
46	(0.5000000000,1.5764261538)	1.5723101972
47	(0.5000000000,1.5723101972)	1.5684589036
48	(0.5000000000,1.5684589036)	1.5648561116
49	(0.5000000000,1.5648561116)	1.5614865425
50	(0.5000000000,1.5614865425)	1.5583357652
51	(0.5000000000,1.5583357652)	1.5553901595
52	(0.5000000000,1.5553901595)	1.5526368813
53	(0.5000000000,1.5526368813)	1.5500638265
54	(0.5000000000,1.5500638265)	1.5476595967
55	(0.5000000000,1.5476595967)	1.5454134645
56	(0.5000000000,1.5454134645)	1.5433153408
57	(0.5000000000,1.5433153408)	1.5413557415
58	(0.5000000000,1.5413557415)	1.5395257565

59	(0.5000000000,1.5395257565)	1.5378170192
60	(0.5000000000,1.5378170192)	1.5362216769
61	(0.5000000000,1.5362216769)	1.5347323625
62	(0.5000000000,1.5347323625)	1.5333421676
63	(0.5000000000,1.5333421676)	1.5320446161
64	(0.5000000000,1.5320446161)	1.5308336399
65	(0.5000000000,1.5308336399)	1.5297035546
66	(0.5000000000,1.5297035546)	1.5286490375
67	(0.5000000000,1.5286490375)	1.5276651054
68	(0.5000000000,1.5276651054)	1.5267470952
69	(0.5000000000,1.5267470952)	1.5258906436
70	(0.5000000000,1.5258906436)	1.5250916693
71	(0.5000000000,1.5250916693)	1.5243463557
72	(0.5000000000,1.5243463557)	1.5236511342
73	(0.5000000000,1.5236511342)	1.5230026691
74	(0.5000000000,1.5230026691)	1.5223978427
75	(0.5000000000,1.5223978427)	1.5218337418
76	(0.5000000000,1.5218337418)	1.5213076446
77	(0.5000000000,1.5213076446)	1.5208170085
78	(0.5000000000,1.5208170085)	1.5203594589
79	(0.5000000000,1.5203594589)	1.5199327779
80	(0.5000000000,1.5199327779)	1.5195348948
81	(0.5000000000,1.5195348948)	1.5191638761
82	(0.5000000000,1.5191638761)	1.5188179169
83	(0.5000000000,1.5188179169)	1.5184953323
84	(0.5000000000,1.5184953323)	1.5181945498
85	(0.5000000000,1.5181945498)	1.5179141017
86	(0.5000000000,1.5179141017)	1.5176526184
87	(0.5000000000,1.5176526184)	1.5174088219
88	(0.5000000000,1.5174088219)	1.5171815196
89	(0.5000000000,1.5171815196)	1.5169695990
90	(0.5000000000,1.5169695990)	1.5167720222
91	(0.5000000000,1.5167720222)	1.5165878207
92	(0.5000000000,1.5165878207)	1.5164160914
93	(0.5000000000,1.5164160914)	1.5162559917
94	(0.5000000000,1.5162559917)	1.5161067357
95	(0.5000000000,1.5161067357)	1.5159675903
96	(0.5000000000,1.5159675903)	1.5158378720
97	(0.5000000000,1.5158378720)	1.5157169431
98	(0.5000000000,1.5157169431)	1.5156042091
99	(0.5000000000,1.5156042091)	1.5154991154
100	(0.5000000000,1.5154991154)	1.5154011449

Error plot "p_n - p_{n-1}"

```
figure; plot([1:i], error); grid on;
xlabel("iterations"); ylabel("|p_n - p_{n-1}|");
title("Error")
```



Printing results

```
fprintf('The approximate root is %.10f',new)
if flag == 0
    disp(' Maximum number of iterations exceeded')
end
```

```
The approximate root is 1.5154011449 Maximum number of iterations exceeded
```

Published with MATLAB® R2017a

Contents

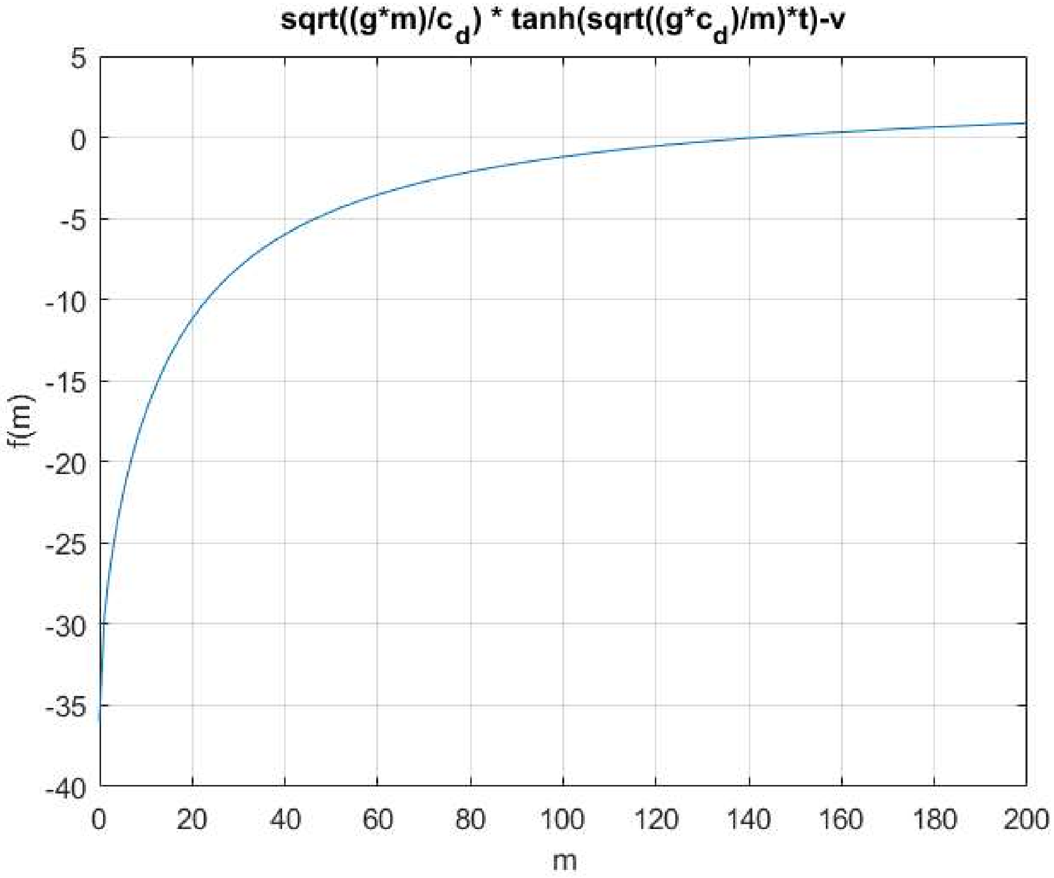

```
clear all;
close all;
```

Function Defination

```
syms m;
c_d = 0.25; v = 36; t = 4; g = 9.81;
f = @(m)sqrt((g*m)/c_d)*tanh(sqrt((g*c_d)/m)*t)-v;
fprime = eval(['@(m)' char(diff(f(m)))]);
```

Plotting Graph

```
M = 0:200;
f_M = 0:200;
n = 1;
for m = 0:200;
    f_M(n) = f(m);
    n = n+1;
end
plot(M, f_M); grid on;
xlabel("m"); ylabel("f(m)");
title("sqrt((g*m)/c_d) * tanh(sqrt((g*c_d)/m)*t)-v");
```



Stopping criterium

```
fprintf("(i) Taking the initial guess as x0 = 50 \n");
x0 = 50 ; %  initial approximation to location of root
TOL = 10^(-5); % absolute error convergence tolerance
Nmax =100; % maximum number of iterations to be performed
```

```
(i) Taking the initial guess as x0 = 50
```

Iteration Scheme

```
flag=0;
for i = 1 : Nmax
    fold=f(x0);
    fprimeold=fprime(x0);
        dx = fold / fprimeold;
    x0 = x0 - dx;
        fprintf ( '\t\t %3d \t %.10f \n', i, x0 );

        if ( abs(dx) < TOL )
           flag=1;
           break
        end
end
```

```
        1      88.3992785546
        2      124.0896500820
        3      140.5416962708
```

```
                    4       142.7071837659
                    5       142.7376272539
                    6       142.7376331084
```

```
if flag == 0
    disp('Maximum number of iterations exceeded.')
end
```

```
fprintf('\n The approximate solution is %f \n', x0)
```

```
 The approximate solution is 142.737633
```

Published with MATLAB® R2017a

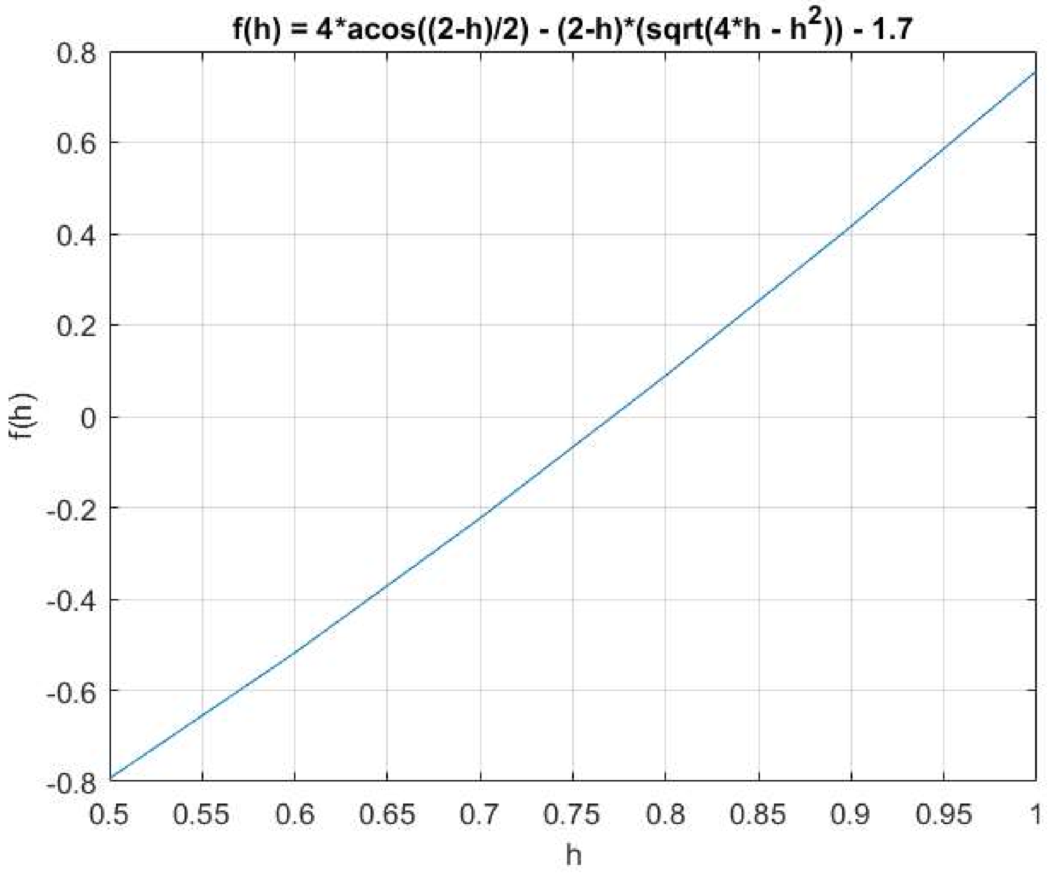
Contents

Function defination

```
f = @(h) 4*acos((2-h)/2) - (2-h)*(sqrt(4*h - h^2)) - 1.7;
a=0.5;  b=1;
```

Plotting graph of f(y)

```
X = a:0.1:b;
Y = a:0.1:b;
n = 1;
for x = a:0.1:b
Y(n) = f(x);
n = n + 1;
end
plot(X,Y); grid on;
xlabel("h");
ylabel("f(h)");
title("f(h) = 4*acos((2-h)/2) - (2-h)*(sqrt(4*h - h^2)) - 1.7")
```



Stopping criterium

```
TOL=10^(-6);
format long;
old = b;
fa = feval(f,a);
fb = feval(f,b);
Nmax = 100;
```

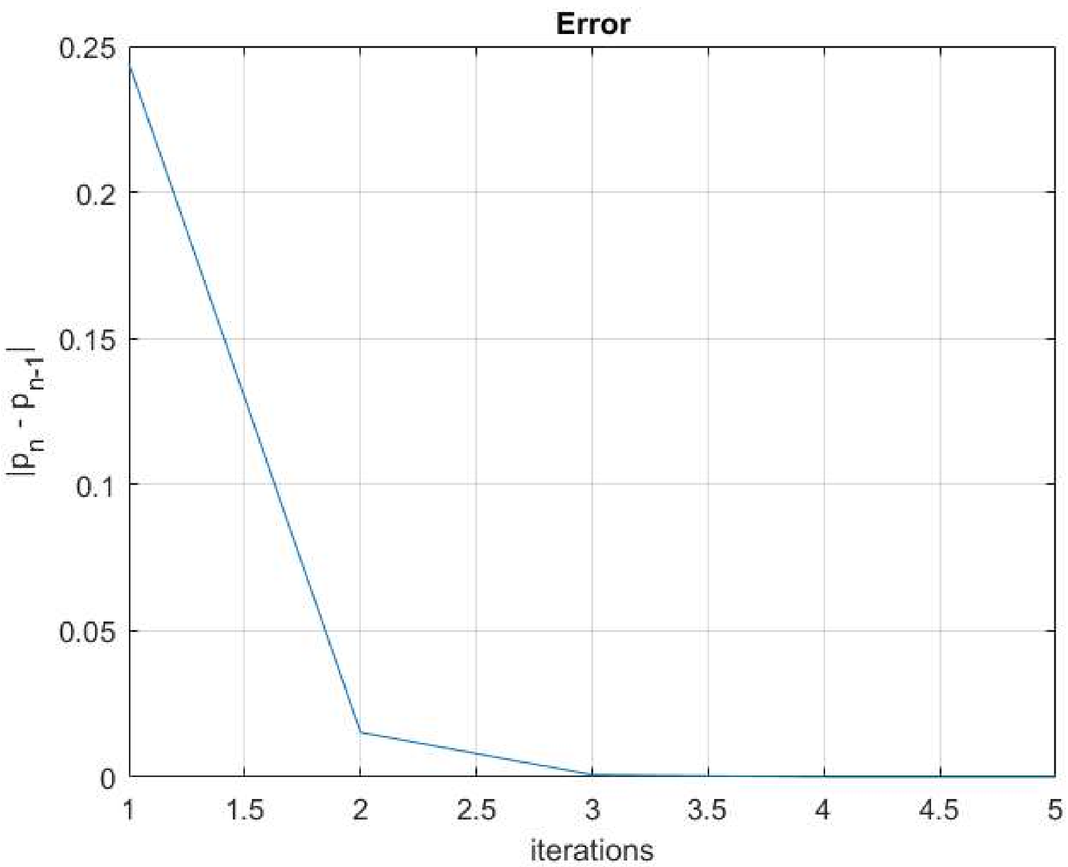
Iteration scheme

```
pvalues=[]; flag =0;
for i = 1 : Nmax
        new = b - fb * ( b - a ) / ( fb - fa );
        fnew = feval(f,new);
        fprintf ( '\t\t %3d \t (%.10f,%.10f) \t %.10f \n', i, a, b, new )

        if ( abs(new-old) < TOL )
        flag=1;
           break
        elseif ( fa * fnew < 0 )
           b = new;
           fb = fnew;
        else
           a = new;
           fa = fnew;
    end
    error(i) = abs(new-old);
        old = new;
        pvalues = [pvalues;old];

end
```

```
       1      (0.5000000000,1.0000000000)      0.7559087671
       2      (0.7559087671,1.0000000000)      0.7711574965
       3      (0.7711574965,1.0000000000)      0.7719060141
       4      (0.7719060141,1.0000000000)      0.7719423707
       5      (0.7719423707,1.0000000000)      0.7719441357
       6      (0.7719441357,1.0000000000)      0.7719442214
```

Error plot "p_n - p_{n-1}"

```
figure; plot([1:5], error); grid on;
xlabel("iterations"); ylabel("|p_n - p_{n-1}|");
title("Error")
```



Printing results

```
fprintf('The approximate root is %.10f',new)
if flag == 0
    disp(' Maximum number of iterations exceeded')
end
```

```
The approximate root is 0.7719442214
```

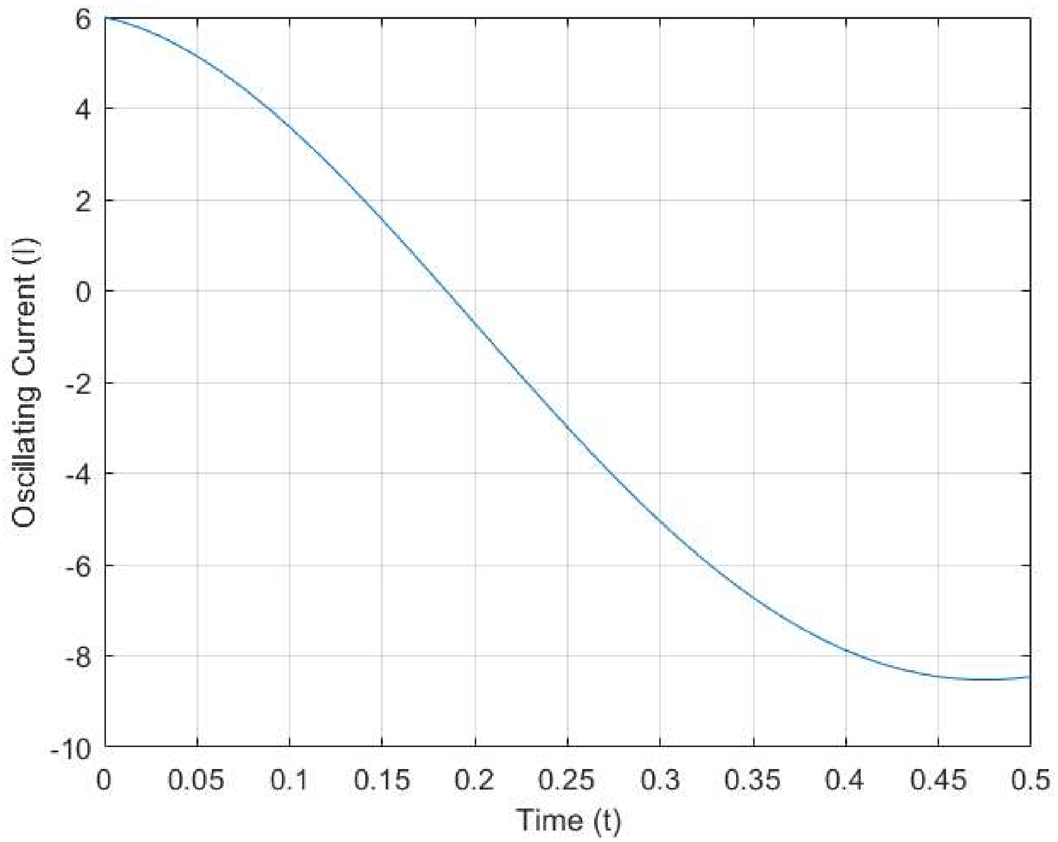
Contents


```
clear all;
close all;
```

Plotting graph of function

```
g = @(x) 9*exp(-x)*cos(2*pi*x) - 3;
X = 0:0.01:0.5; Y = -100:4:100;
n = 1;
for x = 0:0.01:0.5
Y(n) = g(x);
n = n + 1;
end
plot(X,Y); grid on;
xlabel('Time (t)');
ylabel('Oscillating Current (I)');

fprintf("Considering the interval [0,0.5]\n"); SecantMethod(0,0.5);
```

```
Considering the interval [0,0.5]
```



Secant Method applied with different intervals.

```
x0,x1   initial approximations to location of root
```

```
function y = SecantMethod(x0,x1)
    f = inline('9*exp(-x)*cos(2*pi*x) - 3','x');
    TOL=10^(-10); %  absolute error convergence tolerance
    Nmax=100;%   maximum number of iterations to be performed
    flag=0;
    older = x0;   old = x1;
    folder = feval(f,older);
    for i = 2 : Nmax
        fold = feval(f,old);
        dx = fold * ( old - older ) / ( fold - folder );
        new = old - dx;
        fprintf('\t\t %3d \t %.15f \n', i, new )
        if ( abs(dx) < TOL )
            flag=1;
            break
        else
            older = old;
            old = new;
            folder = fold;
        end
    end
    if flag == 0
        disp('Maximum number of iterations exceeded')
    end
end
```

```
          2      0.207486443733952
          3      0.165147507080017
          4      0.184233316589298
          5      0.184364220463863
          6      0.184363081087201
          7      0.184363081134278
```

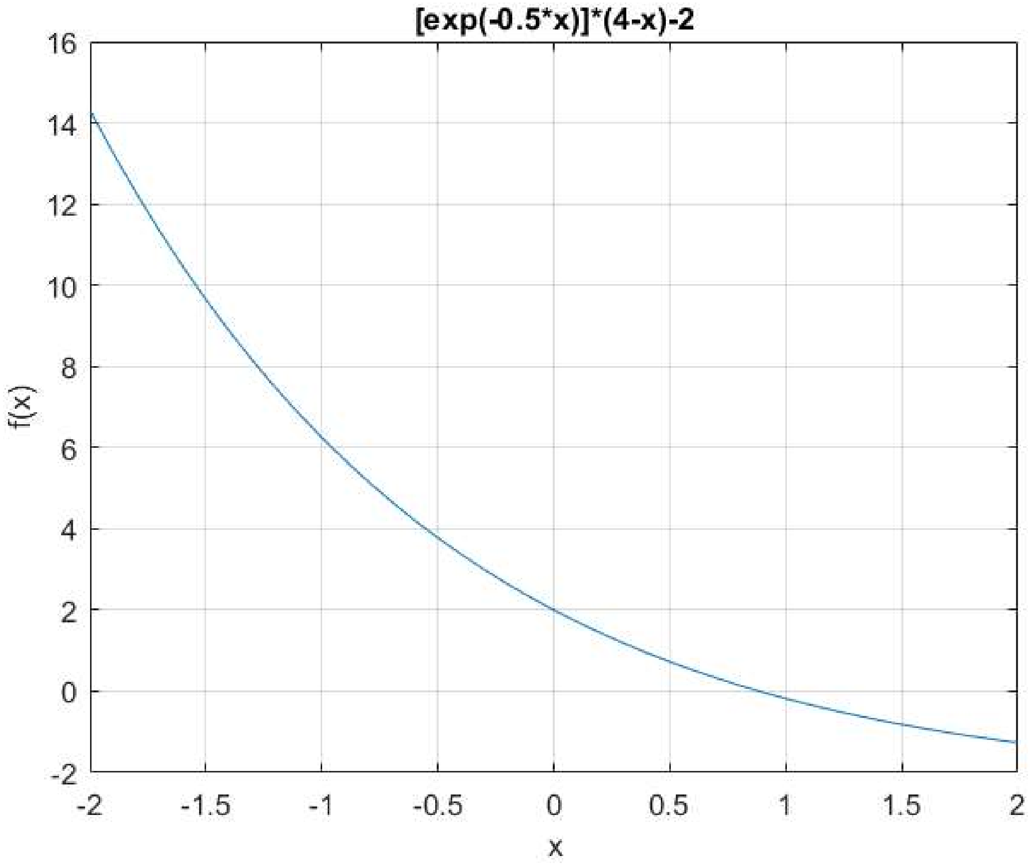
Published with MATLAB® R2017a

Contents


```
clear all;
close all;
```

Plotting Graph

```
f = @(x)[exp(-0.5*x)]*(4-x)-2 ;
X = -2:0.1:2;
Y = -2:0.1:2;
n = 1;
for x = -2:0.1:2;
Y(n) = f(x);
n = n + 1;
end
plot(X,Y); grid on;
xlabel("x"); ylabel("f(x)");
title("[exp(-0.5*x)]*(4-x)-2");
```



Analyzing results for different initial guess

```
NewtonRaphson(2);
NewtonRaphson(6);
NewtonRaphson(8);
```

The Newton Raphson Iteration Scheme

```
function y = NewtonRaphson(x0)
```

Function Defination

```
    syms x;
    f = @(x)[exp(-0.5*x)]*(4-x)-2 ;
    fprime = eval(['@(x)' char(diff(f(x)))]);
```

Stopping criterium

x0 = 2 ; % initial approximation to location of root

```
    TOL = 10^(-5); % absolute error convergence tolerance
    Nmax =50; % maximum number of iterations to be performed
```

Iteration Scheme

```
    flag=0;
    for i = 1 : Nmax
```

```
        fold=f(x0);
        fprimeold=fprime(x0);
        dx = fold / fprimeold;
        x0 = x0 - dx;
        fprintf ( '\t\t %3d \t %.10f \n', i, x0 );

        if ( abs(dx) < TOL )
            flag=1;
            break
        end
    end
```

```
		   1 	 0.2817181715
		   2 	 0.7768868450
		   3 	 0.8817078789
		   4 	 0.8857032412
		   5 	 0.8857088020
```

```
		   1 	 Inf
		   2 	 NaN
		   3 	 NaN
		   4 	 NaN
		   5 	 NaN
		   6 	 NaN
		   7 	 NaN
		   8 	 NaN
		   9 	 NaN
		  10 	 NaN
		  11 	 NaN
		  12 	 NaN
		  13 	 NaN
		  14 	 NaN
		  15 	 NaN
		  16 	 NaN
		  17 	 NaN
		  18 	 NaN
		  19 	 NaN
		  20 	 NaN
		  21 	 NaN
		  22 	 NaN
		  23 	 NaN
		  24 	 NaN
		  25 	 NaN
		  26 	 NaN
		  27 	 NaN
		  28 	 NaN
		  29 	 NaN
		  30 	 NaN
		  31 	 NaN
		  32 	 NaN
		  33 	 NaN
		  34 	 NaN
		  35 	 NaN
		  36 	 NaN
		  37 	 NaN
		  38 	 NaN
		  39 	 NaN
		  40 	 NaN
```

41	NaN
42	NaN
43	NaN
44	NaN
45	NaN
46	NaN
47	NaN
48	NaN
49	NaN
50	NaN

1	121.1963000663
2	7212131452880262800000000.0000000000
3	Inf
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
13	NaN
14	NaN
15	NaN
16	NaN
17	NaN
18	NaN
19	NaN
20	NaN
21	NaN
22	NaN
23	NaN
24	NaN
25	NaN
26	NaN
27	NaN
28	NaN
29	NaN
30	NaN
31	NaN
32	NaN
33	NaN
34	NaN
35	NaN
36	NaN
37	NaN
38	NaN
39	NaN
40	NaN
41	NaN
42	NaN
43	NaN
44	NaN
45	NaN
46	NaN
47	NaN
48	NaN

```
49      NaN
50      NaN
```

Results

```
fprintf("# Taking the initial guess as x_0 = %.1f \n", x0);
if flag == 0
    disp('Maximum number of iterations exceeded.')
end
fprintf('\n The approximate solution is %f \n', x0)
```

```
# Taking the initial guess as x_0 = 0.9

 The approximate solution is 0.885709



# Taking the initial guess as x_0 = NaN
Maximum number of iterations exceeded.

 The approximate solution is NaN



# Taking the initial guess as x_0 = NaN
Maximum number of iterations exceeded.

 The approximate solution is NaN
```

```
end
```

Published with MATLAB® R2017a