CRISTIAN B. JETOMO
MATH 175 - H4L

Exercise 1.1: The Bisection Method in Root-finding

In this exercise, we are to place a ball of cork with radius $R = 5$ cm and density $\rho = 0.12$ g/cm^3 into a motor oil with density $\rho = 0.89$ g/cm^3. It is provided that the depth h to which the ball will sink to is given as the solution of the equation

$$\frac{\rho_f}{3}h^3 - R\rho_f h^2 + \frac{4}{3}R^3\rho_0 = 0 \tag{1.1.1}$$

Our goal is to estimate how far will the cork ball sink. Suppose we let

$$f(h) = \frac{\rho_f}{3}h^3 - R\rho_f h^2 + \frac{4}{3}R^3\rho_0 \tag{1.1.2}$$

Hence, our new goal is to solve this problem by finding the root of the function $f(h)$. We do this by applying the **Bisection Method**. In this method, we first need to set an initial interval in which we can find a root of the function. Note that h represents the estimated depth or height for which the ball will sink to. Since this is a measured value, it is impossible that h is negative. Thus, we assume that $h \geq 0$.

Intermediate Value Theorem guarantees that if $f(a)f(b) < 0$ for a function $f \in \mathcal{C}[a, b]$, then a root of f is within the interval $[a, b]$. With this and using the assumption above, we can set $a = 0$ as the left endpoint of our initial interval. We perform an iterative process to find the right endpoint b. To do that, we define a small value Δ and set the right endpoint to be $b = a + i\Delta$ for the i^{th} iteration. In each iteration, we check whether $f(a)f(b) < 0$ is satisfied. If it does, we end the iteration and let the final value for a and b as our initial endpoints in the Bisection method. Otherwise, we proceed with the next iteration.

Performing this iterative process in MATLAB and setting $\Delta = 0.1$, we obtained that our initial endpoint which will be used in the Bisection method is $[0, 2.4]$. This implies that we can find the root for $f(h)$ within this interval.

```
37  %% Bisection Method Proper
38  iteration_count = 0;
39
40  for i = 1:max_iteration
41
42      % interval-based halting criterion
43      if b-a < ErrorTol
44
45          % current estimate
46          curr_est = (a+b)/2;
47
48          % update table
49          n(iteration_count+1) = iteration_count;
50          A(iteration_count+1) = a;
51          B(iteration_count+1) = b;
52          C(iteration_count+1) = curr_est;
53          D(iteration_count+1) = b-a;
54
55      else
56          % current estimate
57          curr_est = (a+b)/2;
58
59          % update table
60          n(iteration_count+1) = iteration_count;
61          A(iteration_count+1) = a;
62          B(iteration_count+1) = b;
63          C(iteration_count+1) = curr_est;
64          D(iteration_count+1) = b-a;
65
66          % function evaluation at a, b, and curr_est
67          f_a = double(subs(func, a));
68          f_b = double(subs(func, b));
69          f_c = double(subs(func, curr_est));
70
71          %% choosing new interval
72          if f_c == 0
73              break
74          elseif f_a*f_c < 0
75              b = curr_est;
76          else
77              a = curr_est;
78          end
79
80          % updating iteration count
81          iteration_count = iteration_count + 1;
82      end
83  end
84
```

Figure 1.1.1: Program used in MATLAB to perform Bisection Method.

From the code above, a *for* loop is constructed which iterates from 1 to the defined maximum iteration which is 100. If $b - a$ which is the length of the currrent interval is smaller than the defined error tolerance 10^{-6}, then we halt the iteration, solve the midpoint of the current interval, and add our required values in a table. This process is seen in lines $43 - 53$ of the code. Otherwise, we continue the iteration process. For this, we do the same process with above as

observed from lines $56-64$. After which, we evaluate our function at the current endpoints and midpoint of our interval and label them as $f_a, f_b,$ and f_c. For lines $71-78$, we now choose our new interval. If $f_c = 0$, then we stop the process as it indicates that c is the root we are looking for. If $f(a)f(c) < 0$, it implies that the root can be found in the interval $[a, c]$. So we let our current estimate as b to be used in the next iteration. If on the otherhand $f(a)f(c) > 0$, then the root can be found on the interval $[c, b]$ and so we let our current estimate to be a. The iteration counter is then updated in line 81 to proceed to the next iteration. Using these series of code, we obtained the results below.

```
Command Window
  [   0,       0,     1.2,     2.4,          2.4]
  [ 1.0,     1.2,     1.8,     2.4,          1.2]
  [ 2.0,     1.8,     2.1,     2.4,          0.6]
  [ 3.0,     2.1,    2.25,     2.4,          0.3]
  [ 4.0,    2.25,   2.325,     2.4,         0.15]
  [ 5.0,    2.25,  2.2875,   2.325,        0.075]
  [ 6.0,  2.2875, 2.30625,   2.325,       0.0375]
  [ 7.0,  2.2875, 2.29688, 2.30625,      0.01875]
  [ 8.0, 2.29688, 2.30156, 2.30625,     0.009375]
  [ 9.0, 2.30156, 2.30391, 2.30625,    0.0046875]
  [10.0, 2.30391, 2.30508, 2.30625,    0.00234375]
  [11.0, 2.30391, 2.30449, 2.30508,    0.00117188]
  [12.0, 2.30391,  2.3042, 2.30449,   0.000585937]
  [13.0,  2.3042, 2.30435, 2.30449,   0.000292969]
  [14.0, 2.30435, 2.30442, 2.30449,   0.000146484]
  [15.0, 2.30435, 2.30438, 2.30442,  0.0000732422]
  [16.0, 2.30435, 2.30436, 2.30438,  0.0000366211]
  [17.0, 2.30436, 2.30437, 2.30438,  0.0000183105]
  [18.0, 2.30437, 2.30438, 2.30438, 0.00000915527]
  [19.0, 2.30437, 2.30438, 2.30438, 0.00000457764]
  [20.0, 2.30438, 2.30438, 2.30438, 0.00000228882]
  [21.0, 2.30438, 2.30438, 2.30438, 0.00000114441]
  [22.0, 2.30438, 2.30438, 2.30438,    5.72205e-7]

  The estimated root is
      2.3044
```

Figure 1.1.2: Summarized table of the iterations and the obtained estimated root.

In the table, the respective columns are the n^{th} iteration, a, $c = \frac{a+b}{2}$, b, and $b-a$. As observed in the last column, the values for every iteration are decreasing, indicating that the length of the interval becomes smaller. Note also that the error tolerance criteria is met after 22 iterations since $b-a$ in this iteration is 5.272205×10^{-7} which is less than our defined error tolerance 10^{-6}. Therefore, it is concluded that the root of the function is the midpoint of the final interval which is 2.3044. Evaluating the function f at this value, we obtain that $f(2.3044) = 2.1721 \times 10^{-7}$ which is very close to 0, hence a good approximation. Therefore, in the sense of the given situation, this means that the ball of cork will sink at a depth of $h = 2.3044$ cm when placed into the motor oil.

Now, suppose that we instead place the same cork ball on water with a density of 0.998 g/cm^3. By changing some values in the code we used in Figure 1.1.1, we obtained the result provided in Figure 1.1.3 in the next page.

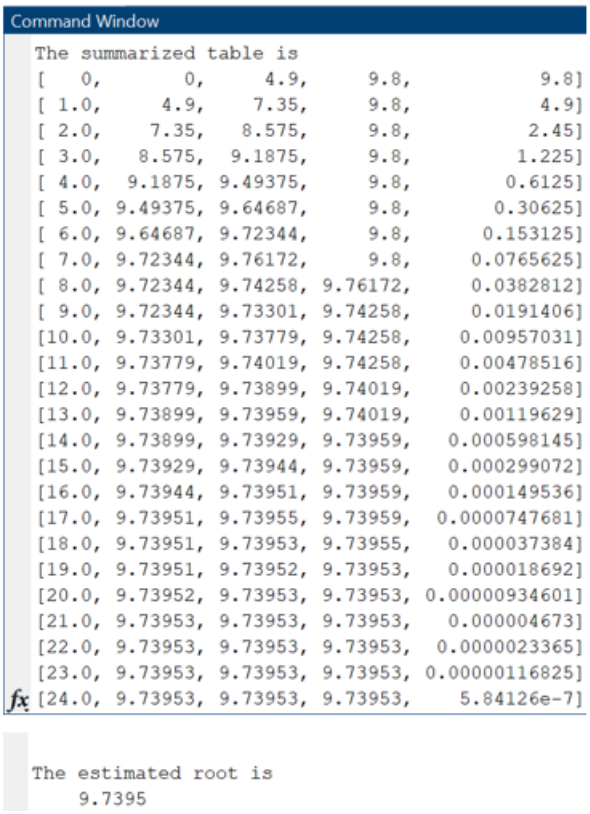

Figure 1.1.3: Summarized table of the iterations and obtained estimated root for 2nd scenario.

Note that in this scenario with the same ball placed on water, there are 24 iterations performed in order to obtain the root of the function. That is, the error tolerance criteria is met after 24 iterations. The obtained root after these iterations is 9.7395 with $f(9.7395) = -3.8498 \times 10^{-7}$. This implies that the ball of cork will sink at a depth of $h = 9.7395$ cm when placed into the water. Comparing the two scenarios, the results imply that the same ball of cork will sink deeper when placed in water than in motor oil. This is expected since the motor oil is more viscous in nature.

Note however that the scenarios are not realistic. The ball of cork has a density with value smaller than the densities of the motor oil and water. This means that the ball should instead float and not sink into these fluids. It is only the case that we obtained a depth value for the two scenarios because of how our function in Equation 1.1.2 is defined.

Remarks: *The program used to solve this problem is made in MATLAB and is made solely by the author of this paper.*