hw5

Ankit Gupta

June 2023

1 Bisection in Matlab

1.1 a

Check bisection.m file

1.2 b

Error Root in [0, 1] is: NaN
Root in [0, 0.5] is: 0.33333
Root in [0.5, 1] is: 0.66667

1.3 c

The root is: 1.5708
The tangent function has vertical asymptotes at odd multiples of pi/2, where it goes to + infinity.
The root-finding algorithm such as the bisection method requires that the function is continuous in the interval where it is applied and that the function values at the endpoints of the interval have different signs. In this case, the function tan(x) - x is not continuous in the interval [1, 2], because it has a discontinuity at x = pi/2, where tan(x) goes to Infinity.

1.4 d

$$y_1 = x^3 - 2x + 1$$

$$y_2 = x^2$$

$$y_1 = y_2$$

$$x^3 - x^2 - 2x + 1 = 0$$

Looking at the graph, Intervals [0,1]
The curves intersect at x = 0.44504

2 Fixed Point Iteration

$$f(x) = e^{-x} - cos(x)$$

$$f(r) = 0$$

2.1 a

[1.1,1.6]
Using bisection.m file
Root is: 1.2927

2.2 b

$$x = g_1(x), g_1(x) = f(x) + x$$
$$x_0 = 1.6$$
$$x_1 = 1.8309$$
$$x_2 = 2.2482$$
$$x_3 = 2.9804$$
$$x_4 = 4.0181$$

This method doesn't converge.

2.3 c

$$x = g_2(x), g_1(x) = x - f(x)$$
$$x_0 = 1.6$$
$$x_1 = 1.4273$$
$$x_2 = 1.3304$$
$$x_3 = 1.3041$$
$$x_4 = 1.2962$$

This method does converge as the value comes close to the real root. 1.2927(part a)

3 On Newton's method

3.1 a

$$a(x) = x^3 - R$$
$$a'(x) = 3x^2$$
$$x_{n+1} = x_n - \frac{a(x_n)}{a'(x_n)}$$
$$x_{n+1} = \frac{1}{3}(2x_n + \frac{R}{x_n^2})$$

3.2 b

$$b(x) = x^2 - \frac{R}{x}$$
$$b'(x) = 2x + \frac{R}{x^2}$$
$$x_{n+1} = x_n - \frac{b(x_n)}{b'(x_n)}$$
$$x_{n+1} = x_n(\frac{x_n^3 + 2R}{2x_n^3 + R})$$

3.3 c

$$c(x) = 1 - \frac{R}{x^3}$$
$$c'(x) = \frac{3R}{x^4}$$
$$x_{n+1} = x_n - \frac{c(x_n)}{c'(x_n)}$$
$$x_{n+1} = \frac{x_n}{3R}(4R - x_n^3)$$

3.4 d

$$d(x) = \frac{1}{x^2} - \frac{x}{R}$$
$$d'(x) = \frac{-2}{x^3} + \frac{1}{R}$$
$$x_{n+1} = x_n - \frac{d(x_n)}{d'(x_n)}$$
$$x_{n+1} = \frac{3Rx_n}{2R + x_n^3}$$

4 Newton's Method in Matlab

Iteration 1: 1.500000
Iteration 2: 1.416667
Iteration 3: 1.414216
Iteration 4: 1.414214
Iteration 5: 1.414214
Iteration 6: 1.414214
Root is: 1.4142
Iteration 1: 1.296285
Iteration 2: 1.292701
Iteration 3: 1.292696
Iteration 4: 1.292696
Iteration 5: 1.292696
Root is: 1.2927

5 When Newton's Method Does Not Work Well.

5.1 i

$$f(x) = (x-1)^m$$
$$r = 1, m = 8$$

5.1.1 a

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$
$$f'(x) = m(x-1)^{m-1}$$
$$x_{n+1} = x_n - \frac{x_n - 1}{m}$$

5.1.2 b

$$x_{n+1} = x_n - \frac{x_n - 1}{m}$$

$$x_0 = 1.1$$

$$x_1 = 1.0875$$

$$x_2 = 1.0765$$

$$x_3 = 1.0669$$

$$x_4 = 1.0585$$

5.1.3 c

Newton's method usually shows quadratic convergence for roots without multiplicity. However, for roots with multiplicity greater than 1, the convergence is usually linear, not quadratic. This is because the assumption that the function can be well-approximated by a tangent line near the root is less accurate when the root has a multiplicity greater than 1.

5.1.4 d

$$m = 20$$

If m = 20, or any large value, the convergence of Newton's method would still be linear. The larger m is, the slower the convergence would be because the function

$$(x - 1)^m$$

becomes flatter near the root as m increases. This makes the tangent line a less accurate approximation of the function near the root, which slows down the convergence of Newton's method.

5.2 ii

Using mynewton.m function

Iteration 1: 1.785398
Iteration 2: 1.844562
Iteration 3: 1.870834
Iteration 4: 1.883346
Iteration 5: 1.889464
Iteration 6: 1.892490
Iteration 7: 1.893995
Iteration 8: 1.894745

x =

1.8947

As Newton's method relies on the derivative of the function at the current guess to estimate the next guess, if the derivative becomes very large, the step size in Newton's method can become very small, making convergence slow. On the other hand, if the derivative becomes very small, the step size can become very large, potentially causing the next guess to jump far away from the root, making convergence erratic, or even causing the method to diverge. This might happen even if the initial guess x0 is close to the root, which is unusual for Newton's method as it usually converges quickly with a good initial guess.

6 The Secant Method in Matlab

Iteration 1: 1.333333
Iteration 2: 1.400000
Iteration 3: 1.414634
Iteration 4: 1.414211
Iteration 5: 1.414214
Iteration 6: 1.414214
Iteration 7: 1.414214
Root: 1.4142
Iteration 1: 1.271572
Iteration 2: 1.290822
Iteration 3: 1.292712
Iteration 4: 1.292696
Iteration 5: 1.292696
Iteration 6: 1.292696
Root: 1.2927