APPM 4600 HW3

Ghulam Yahya Rajabi

Due Feb 16 2024

1 Question 1

1.1 part a:

Given the equation $2x - 1 = \sin(x)$, let $f(x) = 2x - \sin(x) - 1$. Then:

$$f(\pi) = 2\pi - 1 > 0$$
$$f(-\pi) = -2\pi - 1 < 0$$

Since f is continuous, $f(-\pi) < 0$ and $f(\pi) > 0$. By the Intermediate Value Theorem (IVT), there exists a root.

Also, by observing a simple plot, we can guess an interval of $[0, \frac{\pi}{2}]$. In that case:

$$f(0) = -1 < 0$$
$$f\left(\frac{\pi}{2}\right) = \pi - 2 > 0$$

By IVT, there exists a point $c \in [0, \frac{\pi}{2}]$ such that $f(c) = 0$.

1.2 part b:

To prove uniqueness, suppose there are two roots x_1 and x_2:

$$f(x_1) = 2x_1 - 1 - \sin(x_1) = 0$$
$$f(x_2) = 2x_2 - 1 - \sin(x_2) = 0$$

The following must also be true:

$$f(x_1) - f(x_2) = 0$$

$$\frac{f(x_1) - f(x_2)}{x_1 - x_2} = 0$$

The derivative of the function is:

$$f'(x) = 2 - \cos(x) = 0$$

We know by the Mean Value Theorem (MVT), since the function is continuous and differentiable, that there must exist $f'(c) = \frac{f(x_1)-f(x_2)}{x_1-x_2} = 0$. However, $\cos(x) = 2$ leads to a contradiction since the cosine function is bounded by $|1|$. As such, there can be no c such that $f'(c) = 0$, and therefore, there is only one root.

Another way of looking at the problem is to add the two equations:

$$2x_1 - 1 - \sin(x_1) + 2x_2 - 1 - \sin(x_2) = 0$$

After some algebra, we find that:

$$2(x_1 + x_2) = \sin(x_1) + \sin(x_2)$$

This is also a contradiction since $x_1 < x_1 + x_2 < 2x_1$ and $\sin(x_2) < x_2 < 2x_2$ are reconcilable with each other. Therefore, there must only be 1 root.

1.3 part c:

calling script in the appendix as well as the github repo
Number of iterations: 28
the approximate root is 0.8878622125563768

2 Question 2

2.1 part a:

Number of iterations: 11
the approximate root is 5.000073242187501

2.2 part b:

Number of iterations: 3
the approximate root is 5.12875

2.3 part c:

The first iteration of the expanded version gets a root of 5.105 while the non-expended form gets 4.915. As you can see, the first iteration is significantly above the actual root. Furthermore, all the subtractions in the expanded form continuously lead to greater errors.

3 Question 3

3.1 part a:

Applying the theorem to this specific case, we get:

$$10^{-3} \leq \frac{3}{2^n}$$

So,

$$2^n \leq 3 \times 10^3$$

$$n \leq \log_2(3 \times 10^3) \approx 11.55$$

Let us assume 12 iterations are needed.

3.2 part b:

Number of iterations: 11
the approximate root is 1.378662109375

Since the theorem is really conservative and considers the upper bound, we found the root in 1 less iteration.

4 Question 4

4.1 part a:

Let $g(x_n) = x_{n+1}$. The derivative of g at x_n is:

$$g'(x_n) = 6 - \frac{12}{{x_n}^2}$$

$$|g'(2)| = |6 - \frac{12}{4}| = 3 > 1$$

Since the absolute value of the derivative near the fixed point is greater than 1, the sequence diverges.

4.2 part b:

Let $g(x_n) = x_{n+1}$. The first derivative of g at x_n is:

$$g'(x_n) = \frac{2}{3} - \frac{2}{{x_n}^3}$$

So, $g'(3^{1/3}) = 0$.

The second derivative of g at x_n is:

$$g''(x_n) = \frac{6}{{x_n}^4}$$

Thus, $g''(3^{1/3}) = \frac{2}{3\sqrt{3}}$.

Taking the Taylor series of $g(x_n)$ centered at $x_* = 3^{1/3}$ and simplifying, we get:

$$\lim_{n\to\infty} \left| \frac{x_{n+1} - 3^{1/3}}{(x_n - 3^{1/3})^2} \right| = \lim_{n\to\infty} \left| \frac{g''(k)}{2!} \right|, \quad k \in [x_n, 3^{1/3}]$$

For $k \in [x_n, 3^{1/3}]$ as $x_n \to 3^{1/3}$, $k = 3^{1/3}$. Thus:

$$\lim_{n\to\infty} \frac{\frac{2}{3\sqrt{3}}}{2!} = \frac{1}{3^{1/3}}$$

Therefore, it converges quadratically.

4.3 part c:

Let $g(x_n) = x_{n+1}$. The derivative of g at x_n is:

$$g'(x_n) = \frac{-12}{(1+x_n)^2}$$

So, $|g'(3)| = \frac{3}{4} < 1$.

we already know the rate of linear convergence but doing it formally and taking the Taylor series of $g(x_n)$ centered at $x_* = 3$ and simplifying, we get:

$$\lim_{n\to\infty} \frac{x_{n+1} - 3}{|x_n - 3|} = |g'(3)| = \frac{3}{4}$$

Therefore, it converges linearly with a rate of $\frac{3}{4}$.

5 Question 5

5.1 part a:

generate figure:

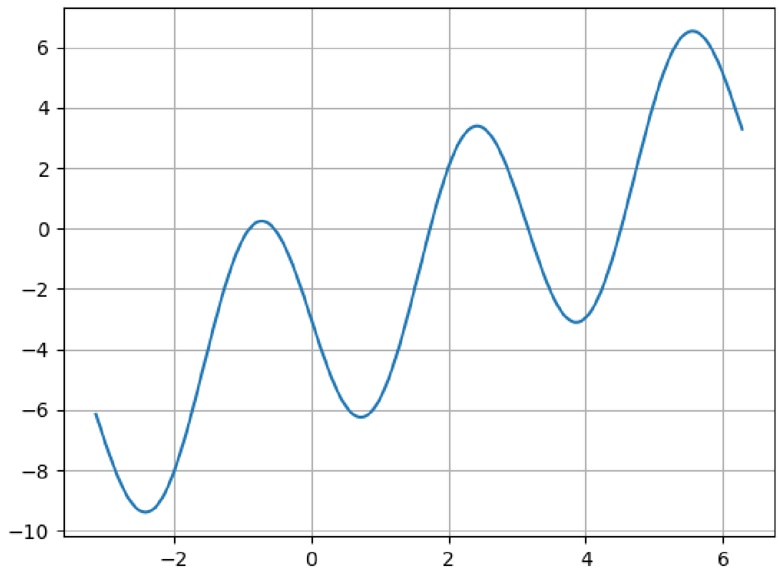

Figure 1: 5 zero crossings

5.2 part b:

Let $g(x_n) = x_{n+1}$. The derivative of g at x_n is given by:

$$g'(x_n) = \frac{5}{4} - 2\cos(x_n)$$

So, evaluating at some points:

$$\begin{aligned} g'(-0.89) &\approx 1.7 > 1 \\ g'(-0.54) &\approx 0.3 < 1 \\ g'(1.73) &\approx 3.2 > 1 \\ g'(3.16) &\approx 0.7 < 1 \\ g'(4.51) &\approx 3.1 > 1 \end{aligned}$$

Roots that the algorithm converges to satisfy $|g'(x)| < 1$, and the divergent ones don't.

6 Appendix (code also available in github repo)

```
# define routines
def bisection(f, a, b, tol):
    fa = f(a)
    fb = f(b)
    if fa * fb > 0:
        ier = 1
        astar = a
        return [astar, ier]

    #   verify end points are not a root
    if fa == 0:
        astar = a
        ier = 0
        return [astar, ier]

    if fb == 0:
        astar = b
        ier = 0
        return [astar, ier]

    count = 0
    d = 0.5 * (a + b)
    while abs(d - a) / abs(d) > tol:
        fd = f(d)
        if fd == 0:
            astar = d
            ier = 0
            return [astar, ier]
        if fa * fd < 0:
            b = d
        else:
            a = d
            fa = fd
        d = 0.5 * (a + b)
        count = count + 1
        print("iteration: ", count, " | curr_root = ", d)
    #       print('abs(d-a) = ', abs(d-a))

    print("Number of iterations: ", count, "\n")
    astar = d
    ier = 0
    return [astar, ier]
```

```
print("")

def fixedpt(f, x0, tol, Nmax):
    """x0 = initial guess"""
    """ Nmax = max number of iterations"""
    """ tol = stopping tolerance"""

    count = 0
    while count < Nmax:
        count = count + 1
        x1 = f(x0)
        if abs(x1 - x0) / abs(x0) < tol:
            xstar = x1
            ier = 0
            return [xstar, ier]
        x0 = x1

    xstar = x1
    ier = 1
    return [xstar, ier]



def driver():
    f = lambda x: 2 * x - 1 - np.sin(x)
    a = 0
    b = np.pi / 2
    tol = 0.5 * 10**-8
    [astar, ier] = bisection(f, a, b, tol)
    print("the approximate root is", astar)
   # print("the error message reads:", ier)
    print("f(root) =", f(astar))
    print("\n")

    return
```