Daniel Lisle

Homework 4

1. $2x - 1 = \sin x$

(a) Find interval $[a, b]$ on which eq'n has root r, use IVT to prove r exists.

Let $f(x) = 2x - 1 - \sin(x)$, continuous.

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

So, choose $[a,b] = [0, \pi]$. By IVT, since $f(x)$ is cts, $\exists$ at least one r on $[a, b]$.

Intuitively, the cts line must cross zero between the negative and positive values.

(b) Prove r is unique on $\mathbb{R}$.

Note $f'(x) = 2 - \cos(x) > 0$, so f is strictly increasing. Therefore it can only have one root.

It would need an extremum to cross zero again.

(c) Approx r to 8 decimal places using bisection code

See appended script.

Result approximation: 0.887862...
iterations: 24

The script found r between -1 and π as expected.

2. Use code from 1(c) approx the root at $x=5$ of

$$f(x) = (x-5)^9$$

w/ $a = 4.82$, $b = 5.2$, $tol = 1e-4$

(a) Result approximation: 5.00007324...
iterations: 11

Found root with desired accurracy.

(b) Result approximation: 5.12875
iterations: 3

Found a root near $x=5$, but further from true root.

(c) The reduced accurracy in (b) results from catastrophic cancellation from the subtraction in the expanded form.

The same error does not result from the multiplication in (a).

3(a) Use Theorem 2.1 to find upper bound on the # iterations in bisection needed to find the root of x^3+x-4 on $[1,4]$ w/ accurracy 10^{-3}.

We need n # iterations such that the actual error (L.H.S. of Theorem 2.1) does not exceed 10^{-3}:

$$10^{-3} \geq \left(\frac{1}{2}\right)^n (4-1)$$

Bound given by

$$n = \left\lceil \log_{1/2}\left(\frac{10^{-3}}{3}\right)\right\rceil = \lceil 11.55.. \rceil = 12.$$

After no more than 12 iterations, we have error at or below 10^{-3}.

(b) Approximate root with bisection code to same accurracy. How does # iterations compare to n?

See appended code.

Approximation: 1.378662...
iterations: 11

The number of iterations is below the upper bounded, as expected.

4. Using Defn 1, which of the followings converges to x_*? If it converges, give order and rate, for linear.

(a) $x_{n+1} = -16 + 6x_n + \frac{12}{x_n}$, $x_* = 2$

Let $f(x) = -16 + 6x + 12/x$

$f'(x) = 6 + \frac{-12}{x^2}$ and $|f'(2)| = 3$

Because $|f'(2)|>1$ iteration does not converge.

Successive iterations will not be closer to 2, because the slope of $F(x)$ will result in large differences between iterates. Theoretically, λ would have to exceed 1, meaning increasing error, which would then make convergence impossible.

(b) $x_{n+1} = \frac{2}{3}x_n + \frac{1}{x_n^2}$, $x_* = 3^{(\frac{1}{3})}$

Let $f(x) = \frac{2}{3}x + \frac{1}{x^2}$,

$f'(x) = \frac{2}{3} + \frac{-2}{x^3}$ and $|f'(3^{\frac{1}{3}})| = \frac{2}{3} - \frac{2}{(3^{\frac{1}{3}})^3} = 0$

Since $|f'(x_*)| = 0$, the iteration converges.

$f''(x) = \frac{6}{x^4}$ and $f''(x_*) = \frac{6}{3^{\frac{4}{3}}} = \frac{2}{\sqrt[3]{3}} \neq 0$

By 2.9, then, the order of convergence $\alpha = 2$.

The error reduces, meaning convergence, and the rate at which the error reduces, increases linearly.

(c) $x_{n+1} = \frac{12}{1+x_n}$, $x_* = 3$

Let $f(x) = \frac{12}{1+x}$

$f'(x) = \frac{-12}{(1+x)^2}$ and $f'(x_*) = \frac{-12}{4^2} = -\frac{3}{4} \neq 0$

$|f'(x_*)| < 1$, so it converges linearly ($\neq 0$) at rate $3/4$.

The error of each iterate reduces at a constant rate, making the order $\alpha = 1$.

5. Find roots of $x - 4\sin(2x) - 3 = 0$ to 10 accurate digits.

(a) Plot $f = x - 4\sin(2x) - 3$. How many roots?

See appended plot. There are 5 crossings, meaning f has five roots.

(b) Programatically find roots of $f(x)$ using

$$x_{n+1} = -\sin(2x_n) + 5x_n/4 - 3/4$$

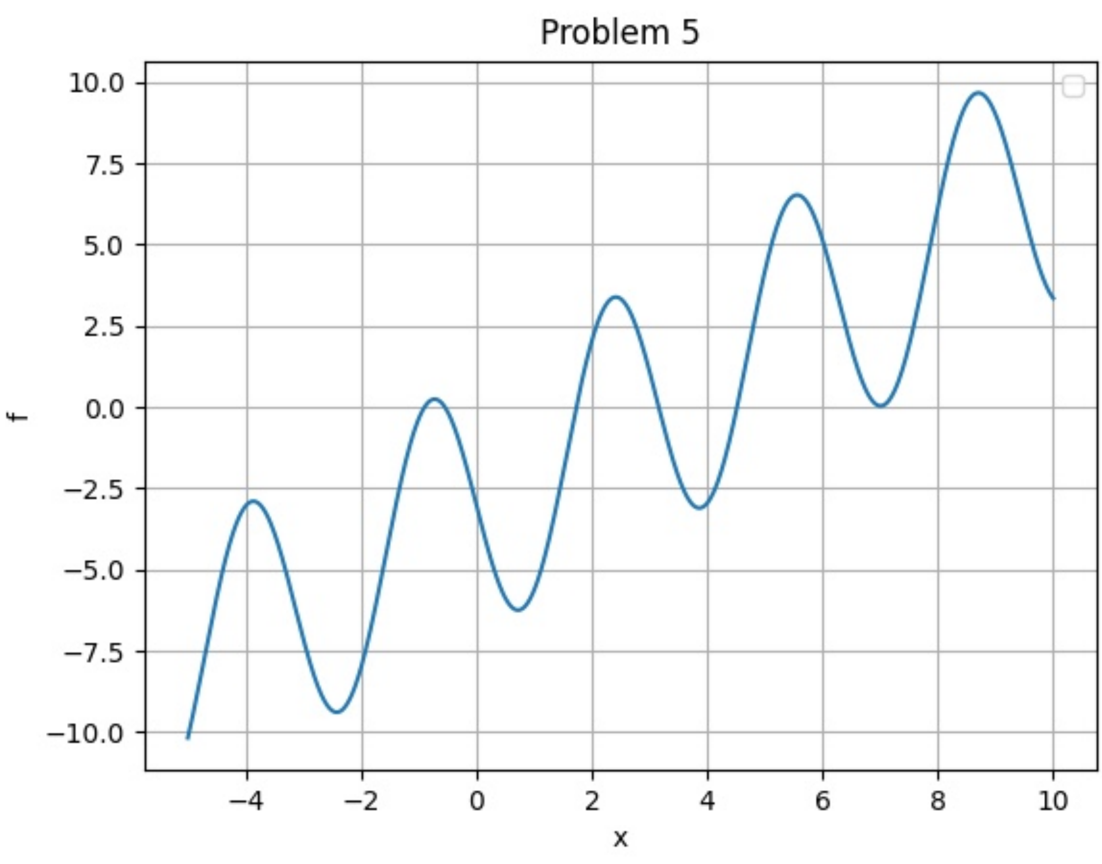
The code found $r_2 = -0.5444424006$8
and $r_4 = 3.1618264865\ldots$ (the roots where f is decreasing)

It could not find r_1, r_3, r_5, the roots where f is increasing.

Let $g = -\sin(2x) + 5x/4 - 3/4$. At r_2 and r_4, $|g'| < 1$ but at r_1, r_3, r_5, $|g'| > 1$. So, at the latter roots $\lim_{n\to\infty} \frac{|x_{n+1} - x_*|}{|x_n - p|} > 1$, meaning the error increases, and the iteration cannot converge.

```python
# PROBLEMS 1,2,3

import numpy as np

def driver():

# use routines

    f = lambda x: x**3 + x - 4

    #f = lambda x: x**9   - 45*x**8  +900*x**7  -10500*x**6
+78750*x**5 - 393750*x**4  +1312500*x**3   - 2812500*x**2  +3515625*x
- 1953125
    a = 1
    b = 4

    tol = 1e-3

    [astar,ier, count] = bisection(f,a,b,tol)
    print('the approximate root is',astar)
    print('the number of iterations was',count)
    print('the error message reads:',ier)
    print('f(astar) =', f(astar))

# define routines
def bisection(f,a,b,tol):

#    Inputs:
#     f,a,b       - function and endpoints of initial interval
#      tol  - bisection stops when interval length < tol

#    Returns:
#      astar - approximation of root
#      ier   - error message
#            - ier = 1 => Failed
#            - ier = 0 == success

#     first verify there is a root we can find in the interval

    fa = f(a)
    fb = f(b);
    if (fa*fb>0):
```

```
       ier = 1
       astar = a
       return [astar, ier, 0]

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

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

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

    astar = d
    ier = 0
    return [astar, ier, count]

driver()
```



Problem 5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
f
−4
−2
0
2
4
6
8
10
x

```python
# PROBLEM 5 (a)

import numpy as np
import matplotlib.pyplot as plt

f = lambda x: x - 4*np.sin(2*x) - 3

x = np.linspace(-5, 10, 1000)

y = f(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('f')
plt.title('Problem 5')
plt.grid(True)
plt.legend()
plt.show()
```

```python
# PROBLEM 5 (b)

import numpy as np

def fixed_pt(x):
    return -np.sin(2 * x) + 5 * x / 4 - 3 / 4

x0 = 4.5
tol = 1e-11
Nmax = 1000

for i in range(Nmax):
    x1 = fixed_pt(x0)

    if abs(x1 - x0) < tol:
        break

    x0 = x1

print("Approx:", x1)
```