1.A)

Assume that the equation converges to x^*: $\lim_{n\to\infty} x_n = x^*$ where $x_{n+1} = g(x)$

Taylor series over $x^* \to g(x_n) = g(x^*) + (x_n - x^*)g'(x^*) + \frac{(x_n - x^*)^2 g''(\xi(x_n))}{2}$

$g'(x^*) = 0 \to g(x_n) - g(x^*) = \frac{(x_n - x^*)^2 g''(\xi(x_n))}{2}$

$\xi(x_n) = x^* + \alpha(x_n - x^*) \to \lim_{n\to\infty} \xi(x_n) = x^* + \alpha \lim_{n\to\infty}(x_n - x^*) = x^*$

Order of Convergence

Suppose $\{p_n\}_{n=0}^{\infty}$ is a sequence that converges to p, with $p_n \neq p$ for all n. If positive constants λ and α exist with

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lambda,$$

then $\{p_n\}_{n=0}^{\infty}$ **converges to p of order α, with asymptotic error constant λ.** ■

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = \lim_{n\to\infty} \frac{|g(x_n) - g(x^*)|}{|x_n - x^*|^2} = \lim_{n\to\infty} \frac{|(x_n - x^*)^2 g''(\xi(x_n))|}{2|x_n - x^*|^2} = \lim_{n\to\infty} \frac{|g''(\xi(x_n))|}{2} = \frac{|g''(x^*)|}{2}$$

Order of convergence is $\alpha = 2$ with $\lambda = \frac{|g''(x^*)|}{2}$

1.B)

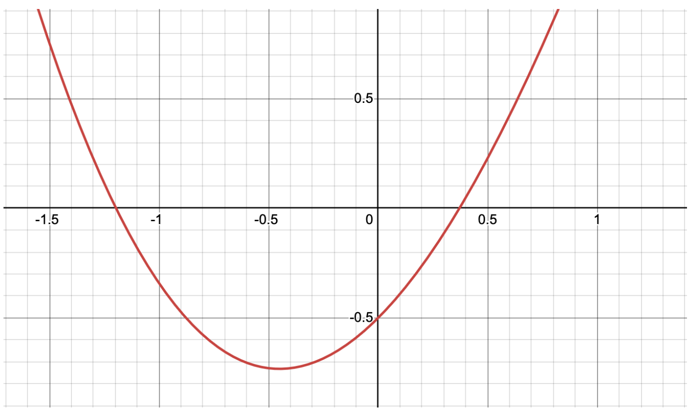
let $g(x) = \frac{2x^2 - 1}{4(x-1)}$

$g(x^*) = x^* \to 2x^{*2} - 4x^* + 1 = 0 \to x^* = 1 \pm \frac{\sqrt{2}}{2}$

$g'(x) = \frac{g(x)}{4(x-1)^2} \to g'(x^*) = 0$

Both roots of the equation have the requirements for part A $\to$ order of convergence is 2

2)



The equation has 2 roots, one near -1 and the other one near 0.5.

Newton – Raphson Method:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 + \mathrm{Sin}(x_n) - 0.5}{2x_n + \mathrm{Cos}(x_n)}$$

Use the python code below, starting from both -1 and 0.5, to find the roots.
Stop when the difference between 2 consecutive points is less than 1e-7.

```
def f(x):
    return x**2 + np.sin(x) - 0.5

def df(x):
    return 2*x + np.cos(x)

def newton_raphson(x0, tol=1e-7, max_iter=1000):
    x_n = x0
    for _ in range(max_iter):
        print(x_n)

        fx_n = f(x_n)
        dfx_n = df(x_n)

        if dfx_n == 0:
            return None

        x_next = x_n - fx_n / dfx_n

        if abs(f(x_next)) < tol:
            return x_next

        x_n = x_next

    return None
```

Result for $x_0 = 0.5$:

```
0.5
0.37780801587057
0.3709105514033993
Root found: 0.37088734037553595
```

Result for $x_0 = -1$:

```
-1
-1.2339326739917766
-1.1970672209676352
-1.1960827342846037
Root found: -1.1960820332974902
```

Fixed Point Method:

$$f(x) = x^2 + \mathrm{Sin}(x) - 0.5 = 0$$

$$g_2(x) = \frac{x^2 + \mathrm{Sin}(x) - 0.5 + 2x}{2} = x \to x_{n+1} = \frac{2x_n^2 + 4x_n - 1}{4} + \frac{\mathrm{Sin}(x)}{2} \to \text{use for finding the solution near} - 1$$

$$g_{-2}(x) = \frac{x^2 + \mathrm{Sin}(x) - 0.5 - 2x}{-2} = x \to x_{n+1} = \frac{2x_n^2 - 4x_n - 1}{-4} + \frac{\mathrm{Sin}(x)}{-2} \to \text{use for finding the solution near } 0.5$$

the choice of the g function is based on the absolute derivative of g being smaller than 1 near each point.

Use the python code below, starting from both -1 and 0.5, to find the roots.
Stop when the difference between 2 consecutive points is less than 1e-7.

```python
def f(x):
    return x**2 + np.sin(x) - 0.5

def g(x):
    return (2 * x**2 - 4 * x - 1) / (-4) + np.sin(x) / (-2)

def iterate_function(x0, tol=1e-7, max_iter=1000):
    x_n = x0
    for _ in range(max_iter):
        print(x_n)
        x_next = g(x_n)

        if abs(f(x_next)) < tol:
            return x_next

        x_n = x_next

    return None

# Initial guess
x0 = 0.5
```

Result for $x_0 = 0.5$:

```
0.5
0.3852872306978985
0.3731514425222818
0.37125453927579155
0.3709471786275893
0.3708970988829834
0.37088893182232063
0.3708875997342035
Fixed point found: 0.3708873824588418
```

```python
def f(x):
    return  (2 * x**2 + 4 * x - 1) / (+4) + np.sin(x) / (+2)

def iterate_function(x0, tol=1e-7, max_iter=1000):
    x_n = x0
    for _ in range(max_iter):
        print(x_n)
        x_next = f(x_n)

        if abs(x_next - x_n) < tol:
            return x_next

        x_n = x_next

    return None

# Initial guess
x0 = -1
```

Result for $x_0 = -1$:

```
-1
-1.1707354924039484
-1.195943346952303
-1.1960838330428523
Fixed point found: -1.1960820097564233
```

Based on the results, the number of iterations starting from -1 is the same for both methods.
However, starting from 0.5, Newton-Raphson finds the solution with 1e-7 error faster than fixed point.
Take note that the rate of convergence is dependent on the choice of function g in the fixed-point method.

3)

Secant Method:

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \text{ where } f(x) = x^3 - 13x^2 + 76$$

Use the python code below, starting with 2 and 3 to find the solution.

```python
def f(x):
    return x**3 - 13*x**2 + 76

def secant_method(f, x0, x1, tol=1e-2, max_iter=1000):
    for _ in range(max_iter):
        f_x0 = f(x0)
        f_x1 = f(x1)

        print(x1)

        if f_x1 - f_x0 == 0:  # Prevent division by zero
            return None

        x2 = x1 - f_x1 * (x1 - x0) / (f_x1 - f_x0)

        if abs(x2 - x1) < tol:
            return x2

        x0, x1 = x1, x2

    return None

# Initial guesses
x0 = 2
x1 = 3
```

Result:

```
3
2.6956521739130435
2.718253062466712
Root found: 2.7188564572581737
```

Bisection Method:
Binary search for the value by following these steps:
1) find an interval (a, b) where $f(a)f(b) < 0$
2) check if $f\left(\frac{a+b}{2}\right)$ is positive or negative
3) choose interval $\left(a, \frac{a+b}{2}\right)$ or $\left(\frac{a+b}{2}, b\right)$ based on which is $f(\text{start})f(\text{end}) < 0$

Use the python code below, with initial interval being (2,3) to find the solution.

```python
def f(x):
    return x**3 - 13*x**2 + 76


def bisection_method(f, a, b, tol=1e-2, max_iter=1000):
    if f(a) * f(b) >= 0:
        return None

    for _ in range(max_iter):
        print(a, b)

        c = (a + b) / 2
        f_c = f(c)


        if f(a) * f_c < 0:
            b = c
        else:
            a = c

        if abs(b - a) < tol:
            return (a + b) / 2

    return None

# Initial interval [a, b]
a = 2
b = 3
```

Result:

```
2 3
2.5 3
2.5 2.75
2.625 2.75
2.6875 2.75
2.71875 2.75
2.71875 2.734375
Root found: 2.72265625
```

The Secant Method is faster in convergence.

4)

$The\ following\ equation\ needs\ to\ be\ solved: x = \log_{10} 56 \rightarrow 10^x - 56 = 0 \rightarrow e^{\ln(10)x} - 56 = 0$

$$Newton - Raphson\ Method: x_{n+1} = x_n - \frac{10^{x_n} - 56}{\ln(10) \times 10^{x_n}}$$

Use the python code in question 2 of the assignment to calculate $\log_{10} 56$ with accuracy of tol=1e-3.

```python
def f(x):
    return 10**x - 56

def df(x):
    return np.log(10) * 10 ** x
```

Result, starting from $x_0 = 1$:

```
1
2.997754616754958
2.5879066929143026
2.2164275409696246
1.9298889474120062
1.7814083950339017
1.749426799236975
Root found: 1.7481897920512308
```

For solving the equation by fixed-point, we need to solve this:

$$e^{\ln(10)x} - 56 = 0 \rightarrow \frac{e^{\ln(10)x} - 56 + cx}{c} = x$$
$$g_c(x) = \frac{e^{\ln(10)x} - 56 + cx}{c} \rightarrow g_c'(x) = \frac{\ln(10)\,10^x}{c} + 1 < 1 \rightarrow choose\ c\ to\ be - 1000$$

Use the python code in question 2 of the assignment to calculate $\log_{10} 56$ with accuracy of tol=1e-3.

```python
def f(x):
    return 10**x - 56

def g(x):
    return (10**x - 56 - 1000 * x) / (-1000)
```

Result, starting from $x_0 = 1$:

```
1
1.046
1.0908826827271842
1.1345549649681486
1.1769231097469754
1.2178943511084985
.
.
.
1.7481775894846088
1.748178935332207
1.7481801076437045
Fixed point found: 1.748181128794732
Number of iterations: 91
```

As shown in the result, Newton-Raphson method converges much faster (with 6 iterations) to achieve 1e-3 accuracy. However, fixed-point takes 91 iterations to get to the same level of accuracy.

5)

let $f(x^*) = 0$ for some $x^* \in [0,1]$
Taylor series over $x^* \rightarrow f(x_i) = f(x^*) + (x_i - x^*)f'\big(\xi(x_i)\big) \rightarrow f(x_i) \le b|x_i - x^*|$

Convergence $\rightarrow \lim_{n\to\infty} |x_n - x^*| = 0 \rightarrow \lim_{n\to\infty} |x_n - x^*| = \lim_{n\to\infty} |x_{n-1} + Mf(x_{n-1}) - x^*| = 0$
$|x_{n-1} + Mf(x_{n-1}) - x^*| \le \big|x_{n-1} - x^* + Mb|x_{n-1} - x^*|\big| \le (Mb + 1)|x_{n-1} - x^*|$

$\rightarrow |x_n - x^*| \le (Mb + 1)|x_{n-1} - x^*|$
$\rightarrow |x_{n-1} - x^*| \le (Mb + 1)|x_{n-2} - x^*|$
…
$\rightarrow |x_1 - x^*| \le (Mb + 1)|x_0 - x^*|$

$$|x_n - x^*| \le (Mb + 1)^n\,|x_0 - x^*| \le (Mb + 1)^n \rightarrow \lim_{n\to\infty} (Mb + 1)^n = 0 \rightarrow 0 < Mb + 1 < 1 \rightarrow 0 > M > -\frac{1}{b}$$

The only part of proof left is to prove all $x_i \in [0,1]$ so that they are in the differentiable domain of f
There is only one x^* which $f(x^*) = 0$, because the function is strictly increasing on [0,1].

we prove by induction:

1) $x_i \geq 0$
$x_i = x_{i-1} + Mf(x_{i-1})$ for $i > 0 \rightarrow \; x_{i-1} + Mf(x_{i-1}) \geq 0$
1.1) $f(x_{i-1}) < 0: M \leq \frac{-x_{i-1}}{f(x_{i-1})} \rightarrow f(x_{i-1})$ is negative and x_{i-1} is positive so it gives $0 \geq M$
1.2) $f(x_{i-1}) > 0: M \geq \frac{-x_{i-1}}{f(x_{i-1})} \rightarrow$ by the proof bellow for all $f(x) > 0, -\frac{1}{b} \geq \frac{-x}{f(x)}$, so it gives $M \geq -\frac{1}{b}$

$$f(x) > 0 \rightarrow f(x) - f(x^*) = f(x) = \int_{x^*}^{x} f'(x)dx \rightarrow f(x) \leq b(x - x^*) \rightarrow \frac{1}{b} + \frac{x^*}{f(x)} \leq \frac{x}{f(x)} \rightarrow \; -\frac{1}{b} \geq -\frac{1}{b} - \frac{x^*}{f(x)} \geq -\frac{x}{f(x)}$$

2) $x_i \leq 1$
$x_i = x_{i-1} + Mf(x_{i-1})$ for $i > 0 \rightarrow \; x_{i-1} + Mf(x_{i-1}) \leq 1$
1.1) $f(x_{i-1}) > 0: M \leq \frac{1 - x_{i-1}}{f(x_{i-1})} \rightarrow f(x_{i-1})$ is positive and $1 - x_{i-1}$ is positive so it gives $0 \geq M$
1.2) $f(x_{i-1}) < 0: M \geq \frac{1 - x_{i-1}}{f(x_{i-1})} \rightarrow$ by the proof bellow for all $f(x) < 0, -\frac{1}{b} \geq \frac{1 - x}{f(x)}$, so it gives $M \geq -\frac{1}{b}$

$$f(x) < 0 \rightarrow f(x) - f(x^*) = f(x) = \int_{x^*}^{x} f'(x)dx \rightarrow f(x) \geq b(x - x^*) \rightarrow \frac{1}{a} + \frac{x^*}{f(x)} \leq \frac{x}{f(x)} \rightarrow \geq -\frac{1}{a} + \frac{1 - x^*}{f(x)} \geq \frac{1 - x}{f(x)}$$
$$\rightarrow -\frac{1}{b} \geq -\frac{1}{b} + \frac{1 - x^*}{f(x)} \; \geq -\frac{1}{a} + \frac{1 - x^*}{f(x)} \geq \frac{1 - x}{f(x)}$$

6)

$e_k = x_k - x^*$
$\tilde{e_k} = y_k - x^*$

$$f(x_k) = f(x^*) + e_k f'(x^*) + \frac{e_k^2 f''(x^*)}{2} + O(e_k^3) = e_k f'(x^*) + \frac{e_k^2 f''(x^*)}{2} + O(e_k^3)$$
$$f(y_k) = f(x^*) + \tilde{e_k} f'(x^*) + \frac{\tilde{e_k}^2 f''(x^*)}{2} + O\left(\tilde{e_k}^3\right) = \tilde{e_k} f'(x^*) + \frac{\tilde{e_k}^2 f''(x^*)}{2} + O\left(\tilde{e_k}^3\right)$$
$$f'(x_k) = f'(x^*) + e_k f''(x^*) + O(e_k^2)$$

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \left(\frac{f'(x^*) + \frac{e_k f''(x^*)}{2} + O(e_k^2)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) e_k$$
$$\rightarrow \tilde{e_k} = y_k - x^* = \; x_k - x^* - \left(\frac{f'(x^*) + \frac{e_k f''(x^*)}{2} + O(e_k^2)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) e_k = \left(1 - \frac{f'(x^*) + \frac{e_k f''(x^*)}{2} + O(e_k^2)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) e_k$$
$$= \left(\frac{\frac{e_k f''(x^*)}{2} + O(e_k^2)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) e_k = \left(\frac{\frac{f''(x^*)}{2} + O(e_k)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) e_k^2 \rightarrow \tilde{e_k} = \; Ce_k^2$$

$$x_{k+1} = y_k - \frac{f(y_k)}{f'(x_k)} = y_k - \left(\frac{f'(x^*) + \frac{\tilde{e_k} f''(x^*)}{2} + O\left(\tilde{e_k}^2\right)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) \tilde{e_k}$$
$$\rightarrow e_{k+1} = x_{k+1} - x^* = y_k - x^* - \left(\frac{f'(x^*) + \frac{\tilde{e_k} f''(x^*)}{2} + O\left(\tilde{e_k}^2\right)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) \tilde{e_k} = \left(1 - \frac{f'(x^*) + \frac{\tilde{e_k} f''(x^*)}{2} + O\left(\tilde{e_k}^2\right)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) \tilde{e_k}$$
$$= \left(\frac{e_k f''(x^*) + O(e_k^2) - \frac{\tilde{e_k} f''(x^*)}{2} - O\left(\tilde{e_k}^2\right)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) \tilde{e_k} = \left(\frac{e_k f''(x^*) + O(e_k^2) - \frac{Ce_k^2 f''(x^*)}{2} - O(e_k^4)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) Ce_k^2$$
$$= \left(\frac{f''(x^*) + O(e_k)}{f'(x^*) + e_k f''(x^*) + O(e_k^2)}\right) Ce_k^3 \rightarrow e_{k+1} = C' e_k^3$$