Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Solution 1 - Theoretical Derivation of Order of Convergence

Part (a) - Secant method

Assumptions and well-known information:

1. The sequence of approximations $\{x_n\}$ generated by the Secant method converges to the root r.
2. The function $f(x)$ is sufficiently differentiable in the neighbourhood of the root r.
3. The root r has a known integer multiplicity $m \geq 1$. This allows us to write $f(x)$ in the form $f(x) = (x-r)^m g(x)$, where $g(r) \neq 0$.

The Secant method iteration is given as follows:

$$x_{n+1} = x_n - f(x_n)\left(\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right)$$

Let $r = x_{true\_root}$. In that case, because $e_n = x_n - r$, we can state the following:

$$x_n = e_n + r$$

$$x_{n-1} = e_{n-1} + r$$

$$x_{n+1} = e_{n+1} + r$$

Substituting this into the above equation, we get:

$$e_{n-1} + r = e_n + r - f(e_n + r)\left(\frac{(e_n + r) - (e_{n-1} + r)}{f(e_n + r) - f(e_{n-1} + r)}\right)$$

Let us now simplify this equation:

$$e_{n-1} + r = e_n + r - f(e_n + r)\left(\frac{(e_n + r) - (e_{n-1} + r)}{f(e_n + r) - f(e_{n-1} + r)}\right)$$

$$e_{n-1} = e_n - f(e_n + r)\left(\frac{e_n + r - e_{n-1} - r}{f(e_n + r) - f(e_{n-1} + r)}\right)$$

$$e_{n-1} = e_n - f(e_n + r)\left(\frac{e_n - e_{n-1}}{f(e_n + r) - f(e_{n-1} + r)}\right)$$

Name: Suhas Kamath
SR No: 06-18-01-10-51-25-1-25945
Email ID: suhaskamath@iisc.ac.in
Date: November 8, 2025

$$e_{n+1} = e_n - \frac{f(e_n + r)(e_n - e_{n-1})}{f(e_n + r) - f(e_{n-1} + r)}$$

$$e_{n+1} = e_n \left(\frac{f(e_n + r) - f(e_{n-1} + r)}{f(e_n + r) - f(e_{n-1} + r)} \right) - \frac{f(e_n + r)(e_n - e_{n-1})}{f(e_n + r) - f(e_{n-1} + r)}$$

$$e_{n+1} = \frac{e_n(f(e_n + r) - f(e_{n-1} + r)) - f(e_n + r)(e_n - e_{n-1})}{f(e_n + r) - f(e_{n-1} + r)}$$

$$e_{n+1} = \frac{e_n f(e_n + r) - e_n f(e_{n-1} + r) - e_n f(e_n + r) + e_{n-1} f(e_n + r)}{f(e_n + r) - f(e_{n-1} + r)}$$

$$e_{n+1} = \frac{e_{n-1} f(e_n + r) - e_n f(e_{n-1} + r)}{f(e_n + r) - f(e_{n-1} + r)}$$

Now, we can apply Taylor series expansion. Taylor series expansion states that:

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x - a)^k$$

In this case, however, we can modify the Taylor series expansion by taking the expansion about r, which is the true root, with multiplicity m. Let us split the summation as follows:

$$f(x) = \sum_{k=0}^{m} \frac{f^{(k)}(r)}{k!}(x - r)^k + \sum_{k=m}^{\infty} \frac{f^{(k)}(r)}{k!}(x - r)^k$$

Because the multiplicity of r is m, we can state that $f^{(k)}(r) = 0, \forall\, 0 \leq r \leq m$. Hence, $\sum_{k=0}^{m} \frac{f^{(k)}(r)}{k!}(x - r)^k = 0$. This simplifies the summation to:

$$f(x) = \sum_{k=m}^{\infty} \frac{f^{(k)}(r)}{k!}(x - r)^k$$

For simplicity sake, let us take $C_k = \frac{f^{(k)}(r)}{k!}$. Now, the summation is:

$$f(x) = \sum_{k=m}^{\infty} C_k(x - r)^k$$

Let us now obtain an expression for $f(r + e)$:

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

$$f(r+e) = \sum_{k=m}^{\infty} C_k (r+e-r)^k$$

$$f(r+e) = \sum_{k=m}^{\infty} C_k e^k$$

$$f(r+e) = C_m e^m + C_{m+1} e^{m+1} + O(e^{m+2})$$

Now, let us approximate $f(r+e_n)$ and $f(r+e_{n-1})$, so that we can simplify the fraction that we obtained using the secant iteration formula earlier:

$$f(r+e_n) = C_m e_n^m + C_{m+1} e_n^{m+1} + O(e_n^{m+2})$$

$$f(r+e_{n-1}) = C_m e_{n-1}^m + C_{m+1} e_{n-1}^{m+1} + O(e_{n-1}^{m+2})$$

Now, let us substitute these expressions into the fraction that we obtained using the secant iteration formula:

$$e_{n+1} = \frac{e_{n-1} f(e_n + r) - e_n f(e_{n-1} + r)}{f(e_n + r) - f(e_{n-1} + r)}$$

$$e_{n+1} = \frac{e_{n-1}(C_m e_n^m + C_{m+1} e_n^{m+1} + O(e_n^{m+2})) - e_n(C_m e_{n-1}^m + C_{m+1} e_{n-1}^{m+1} + O(e_{n-1}^{m+2}))}{(C_m e_n^m + C_{m+1} e_n^{m+1} + O(e_n^{m+2})) - (C_m e_{n-1}^m + C_{m+1} e_{n-1}^{m+1} + O(e_{n-1}^{m+2}))}$$

$$e_{n+1} = \frac{C_m e_n^m e_{n-1} + C_{m+1} e_n^{m+1} e_{n-1} + O(e_n^{m+2}) e_{n-1} - C_m e_{n-1}^m e_n - C_{m+1} e_{n-1}^{m+1} e_n - O(e_{n-1}^{m+2}) e_n}{C_m e_n^m + C_{m+1} e_n^{m+1} + O(e_n^{m+2}) - C_m e_{n-1}^m - C_{m+1} e_{n-1}^{m+1} - O(e_{n-1}^{m+2})}$$

$$e_{n+1} = \frac{C_m (e_n^m e_{n-1} - e_{n-1}^m e_n) + C_{m+1} (e_n^{m+1} e_{n-1} - e_{n-1}^{m+1} e_n) + \dots}{C_m (e_n^m - e_{n-1}^m) + C_{m+1} (e_n^{m+1} - e_{n-1}^{m+1}) + \dots}$$

$$e_{n+1} = \frac{C_m e_n e_{n-1} (e_n^{m-1} - e_{n-1}^{m-1}) + C_{m+1} e_n e_{n-1} (e_n^m - e_{n-1}^m) + \dots}{C_m (e_n^m - e_{n-1}^m) + C_{m+1} (e_n^{m+1} - e_{n-1}^{m+1}) + \dots}$$

$$e_{n+1} = \frac{e_n e_{n-1} (C_m (e_n^{m-1} - e_{n-1}^{m-1}) + C_{m+1} (e_n^m - e_{n-1}^m)) + \dots}{C_m (e_n^m - e_{n-1}^m) + C_{m+1} (e_n^{m+1} - e_{n-1}^{m+1}) + \dots}$$

Let us now start the convergence analysis. For simplicity, let us split the analysis into two cases: $m = 1$, and $m > 1$. Let us see what happens when $m = 1$:

$$e_{n+1} = \frac{e_n e_{n-1}(C_1(e_n^{1-1} - e_{n-1}^{1-1}) + C_{1+1}(e_n^1 - e_{n-1}^1) + \dots)}{C_1(e_n^1 - e_{n-1}^1) + C_{1+1}(e_n^{1+1} - e_{n-1}^{1+1}) + \dots}$$

$$e_{n+1} = \frac{e_n e_{n-1}(C_1(1-1) + C_2(e_n - e_{n-1})) + \dots}{C_1(e_n - e_{n-1}) + C_2(e_n^2 - e_{n-1}^2) + \dots}$$

$$e_{n+1} = \frac{e_n e_{n-1} C_2(e_n - e_{n-1}) + \dots}{C_1(e_n - e_{n-1}) + C_2(e_n^2 - e_{n-1}^2) + \dots}$$

$$e_{n+1} = \frac{e_n e_{n-1} C_2(e_n - e_{n-1}) + \dots}{(e_n - e_{n-1})[C_1 + C_2(e_n + e_{n-1})] + \dots}$$

$$e_{n+1} \approx \frac{e_n e_{n-1} C_2(e_n - e_{n-1})}{(e_n - e_{n-1})[C_1 + C_2(e_n + e_{n-1})]}$$

$$e_{n+1} \approx \frac{e_n e_{n-1} C_2}{C_1 + C_2(e_n + e_{n-1})}$$

$$e_{n+1} \approx \frac{C_2 e_n e_{n-1}}{C_1}$$

We know that α is the order of convergence. Hence, we can state that $e_{n+1} \approx \lambda e_n^\alpha$. Similarly, $e_n \approx \lambda e_{n-1}^\alpha$. This implies the following:

$$e_{n-1} \approx \left(\frac{e_n}{\lambda}\right)^{\frac{1}{\alpha}}$$

Substituting this, we get:

$$\lambda e_n^\alpha \approx \frac{C_2 e_n}{C_1} \times \left(\frac{e_n}{\lambda}\right)^{\frac{1}{\alpha}}$$

$$\lambda e_n^\alpha \approx \frac{C_2 e_n}{C_1} \times e_n^{\frac{1}{\alpha}} \times \lambda^{-\frac{1}{\alpha}}$$

$$\lambda e_n^\alpha \approx \frac{C_2}{C_1} \times e_n^{1+\frac{1}{\alpha}} \times \lambda^{-\frac{1}{\alpha}}$$

$$e_n^\alpha \approx \frac{C_2}{C_1} \times e_n^{1+\frac{1}{\alpha}} \times \lambda^{-\frac{1}{\alpha}-1}$$

In this equation, the exponents of e_n need to be equal on both sides of the inequality. This provides us with the following equation:

$$\alpha = 1 + \frac{1}{\alpha}$$

Let us solve the equation to obtain α:

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

$$\alpha^2 - \alpha - 1 = 0$$

$$\alpha = \frac{-(-1) \pm \sqrt{(-1)^2 - 4(1)(-1)}}{2(1)}$$

$$\alpha = \frac{1 \pm \sqrt{1+4}}{2}$$

$$\alpha = \frac{1 \pm \sqrt{5}}{2}$$

$$\alpha \approx 1.618$$

Now, let us look at the case with multiple roots, i.e. $m > 1$. Let us compare using only the leading terms:

$$e_{n+1} \approx \frac{e_n e_{n-1} C_m (e_n^{m-1} - e_{n-1}^{m-1})}{C_m (e_n^m - e_{n-1}^m)}$$

$$e_{n+1} \approx \frac{e_n e_{n-1} (e_n^{m-1} - e_{n-1}^{m-1})}{(e_n^m - e_{n-1}^m)}$$

$$e_{n+1} \approx e_n e_{n-1} \times \frac{e_n - e_{n-1}}{e_n - e_{n-1}} \times \frac{\sum_{j=0}^{m-2} e_n^{m-2-j} e_{n-1}^j}{\sum_{j=0}^{m-1} e_n^{m-1-j} e_{n-1}^j}$$

$$e_{n+1} \approx e_n e_{n-1} \times \frac{\sum_{j=0}^{m-2} e_n^{m-2-j} e_{n-1}^j}{\sum_{j=0}^{m-1} e_n^{m-1-j} e_{n-1}^j}$$

We can approximate the summation terms as follows:

$$\sum_{j=0}^{m-2} e_n^{m-2-j} e_{n-1}^j \approx (m-1) e_{n-1}^{m-2}$$

$$\sum_{j=0}^{m-1} e_n^{m-1-j} e_{n-1}^j \approx m e_{n-1}^{m-1}$$

Substituting these approximations back into the expression for e_{n+1}, we get:

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

$$e_{n+1} \approx e_n e_{n-1} \times \frac{(m-1)e_{n-1}^{m-2}}{m e_{n-1}^{m-1}}$$

$$e_{n+1} \approx e_n e_{n-1} \times \frac{m-1}{m e_{n-1}}$$

$$e_{n+1} \approx e_n \times \frac{m-1}{m}$$

$$e_{n+1} \approx e_n \times \left(\frac{m}{m} - \frac{1}{m}\right)$$

$$e_{n+1} \approx e_n \times \left(1 - \frac{1}{m}\right)$$

Because $e_{n+1} \approx \lambda e_n^{\alpha}$, we can state that $\lambda \approx 1 - \frac{1}{m}$. This shows us that convergence in Secant method is linear.
To conclude, the rate of convergence for the Secant method is as follows:

$$\alpha = \begin{cases} \frac{1+\sqrt{5}}{2} \approx 1.618 & \text{if } m = 1 \\ 1 & \text{if } m > 1 \end{cases}$$

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Part (b) - Muller's method

Assumptions and well-known information:

1. The sequence of approximations $\{x_n\}$ generated by the Muller method converges to the root r.
2. The function $f(x)$ is at least three times continuously differentiable in a neighbourhood of the root r.
3. The third derivative at the root, $f'''(r)$, is non-zero.

Muller's method finds the next approximation, i.e. x_{n+1} using a combination of three preceding points, i.e. $(x_n, f(x_n))$, $(x_{n-1}, f(x_{n-1}))$, and $(x_{n-2}, f(x_{n-2}))$. Mullers method basically obtains the unique parabola that passes through these three points. Let us call this parabola as $P_2(x)$. The error in polynomial interpolation is as follows:

$$f(x) - P_2(x) = \frac{f'''(\xi)}{3!}(x - x_n)(x - x_{n-1})(x - x_{n-2})$$

Let $r = x_{true\_root}$. We know that $e_n = x_n - r$.

The error is as follows:

$$f(r) - P_2(r) = \frac{f'''(\xi)}{3!}(r - x_n)(r - x_{n-1})(r - x_{n-2})$$

$$0 - P_2(r) = \frac{f'''(\xi)}{3!}(-e_n)(-e_{n-1})(-e_{n-2})$$

$$-P_2(r) = -\frac{f'''(\xi)}{3!}e_n e_{n-1} e_{n-2}$$

$$P_2(r) = \frac{f'''(\xi)}{3!}e_n e_{n-1} e_{n-2}$$

By definition, x_{n+1} is a root of the parabola $P_2(x)$, so $P_2(x_{n+1}) = 0$. Using a first-order Taylor expansion of $P_2(x)$ around the root r:

$$P_2(x_{n+1}) \approx P_2(r) + P_2'(r) \times (x_{n+1} - r)$$

$$0 \approx P_2(r) + P_2'(r)e_{n+1}$$

$$P_2(r) \approx -P_2'(r)e_{n+1}$$

As the sequence $\{x_k\}$ converges to r, the interpolating parabola $P_2(x)$ converges to the function $f(x)$ in the neighbourhood of r. Therefore, its derivative also converges:

$$\lim_{n\to\infty} P_2'(r) = f'(r)$$

We now have two expressions for $P_2(r)$. Let us equate these two equations and attempt to simplify:

$$\frac{f'''(\xi)}{3!}e_n e_{n-1} e_{n-2} = -f'(r)e_{n+1}$$

$$-f'(r)e_{n+1} = \frac{f'''(\xi)}{3!}e_n e_{n-1} e_{n-2}$$

$$e_{n+1} = -\frac{f'''(\xi)}{6 \times f'(r)}e_n e_{n-1} e_{n-2}$$

Letting $\lambda = \left|-\frac{f'''(r)}{6f'(r)}\right|$, the asymptotic error relation is:

$$|e_{n+1}| \approx \lambda e_n e_{n-1} e_{n-2}$$

Now, we can solve for order of convergence:

By definition, we expect a relationship of the form $|e_{k+1}| \approx C|e_k|^\alpha$, where C is a constant.

$$|e_{n+1}| \approx \lambda e_n e_{n-1} e_{n-2}$$

$$|e_n| \approx C|e_{n-1}|^\alpha$$

$$|e_{n-1}| = \left(\frac{|e_n|}{C}\right)^{\frac{1}{\alpha}}$$

Similarly, we can state:

$$|e_{n+1}| \approx \lambda e_n e_{n-1} e_{n-2}$$

$$|e_{n-2}| = \left(\frac{|e_{n-1}|}{C}\right)^{\frac{1}{\alpha}}$$

$$|e_{n-2}| = \left(\frac{\left(\frac{|e_n|}{C}\right)^{\frac{1}{\alpha}}}{C}\right)^{\frac{1}{\alpha}}$$

$$|e_{n-2}| = \left(\frac{|e_n|^{\frac{1}{\alpha}}}{C^{\frac{1}{\alpha}} \times C}\right)^{\frac{1}{\alpha}}$$

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

$$|e_{n-2}| = \left(\frac{|e_n|^{\frac{1}{\alpha}}}{C^{1+\frac{1}{\alpha}}} \right)^{\frac{1}{\alpha}}$$

$$|e_{n-2}| = \frac{|e_n|^{\frac{1}{\alpha}(\frac{1}{\alpha})}}{C^{\frac{1}{\alpha}(1+\frac{1}{\alpha})}}$$

$$|e_{n-2}| = \frac{|e_n|^{\frac{1}{\alpha^2}}}{C^{\frac{1}{\alpha}+\frac{1}{\alpha^2}}}$$

We can substitute the expressions for $|e_{n-1}|$ and $|e_{n-2}|$ into the original expression:

$$|e_{n+1}| \approx \lambda e_n e_{n-1} e_{n-2}$$

$$C|e_n|^\alpha \approx \lambda e_n \times \left(\frac{|e_n|}{C} \right)^{\frac{1}{\alpha}} \times \frac{|e_n|^{\frac{1}{\alpha^2}}}{C^{\frac{1}{\alpha}+\frac{1}{\alpha^2}}}$$

$$C|e_n|^\alpha \approx \lambda e_n \times \frac{|e_n|^{\frac{1}{\alpha}}}{C^{\frac{1}{\alpha}}} \times \frac{|e_n|^{\frac{1}{\alpha^2}}}{C^{\frac{1}{\alpha}+\frac{1}{\alpha^2}}}$$

$$C|e_n|^\alpha \approx \lambda \times \frac{|e_n|^{1+\frac{1}{\alpha}+\frac{1}{\alpha^2}}}{C^{\frac{2}{\alpha}+\frac{1}{\alpha^2}}}$$

$$|e_n|^\alpha \approx \lambda \times \frac{|e_n|^{1+\frac{1}{\alpha}+\frac{1}{\alpha^2}}}{C^{\frac{2}{\alpha}+\frac{1}{\alpha^2}+1}}$$

For the two equations to be equal, the powers of e_n must be equal on both sides of the equation. Using this fact, we can derive a new equation for α:

$$\alpha = 1 + \frac{1}{\alpha} + \frac{1}{\alpha^2}$$

$$\alpha^3 = \alpha^2 + \alpha + 1$$

$$\alpha^3 - \alpha^2 - \alpha - 1 = 0$$

Solving this equation, we get three solutions:

$$\alpha_1 = 1.839$$

$$\alpha_2 = -0.420 + 0.606i$$

$$\alpha_3 = -0.420 - 0.606i$$

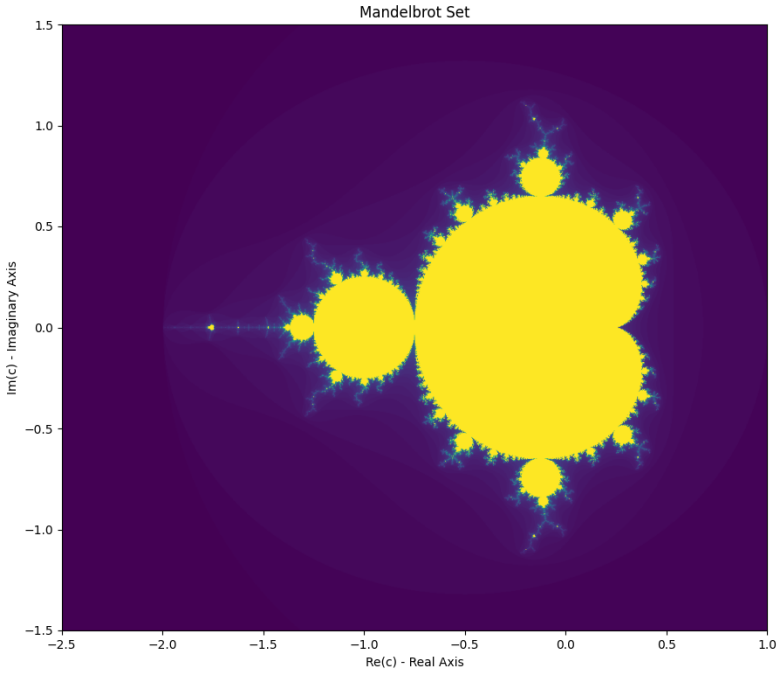
Because α has to be a positive real number, we know that $\alpha = 1.839$.

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Solution 2 - Chaotic world of fractals

Part (a) - Mandelbrot Set

The visualisation presented using the fixed-point iteration $z_{n+1} = z_n^2 + c$ is as follows:



Algorithm

The steps followed to obtain the above visualisation are as follows:

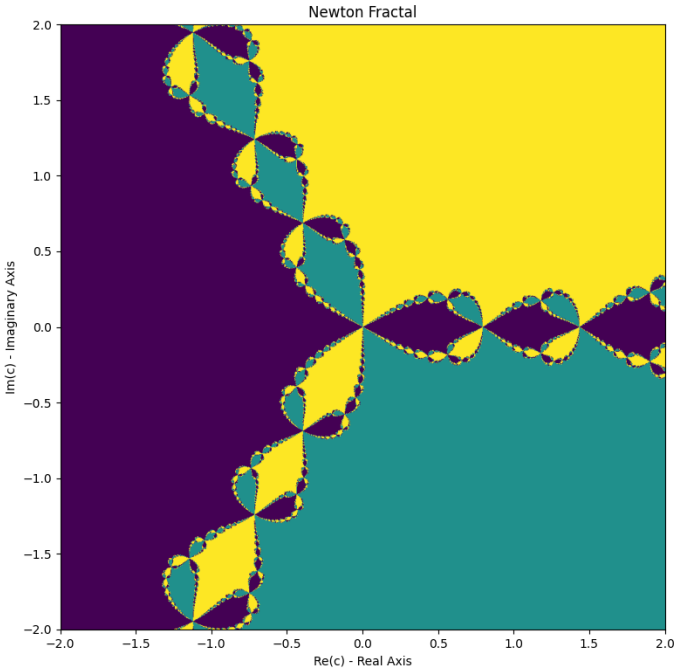
1. Initialise the parameters of the grid, this includes the following values:
   - width
   - height
   - minimum and maximum values of x
   - minimum and maximum values of y
2. Generate two arrays that correspond to the real and complex values of c respectively. We shall test every combination of these two arrays to generate the visualisation.
3. Create a two-dimensional array for storing the values of the Mandelbrot plot and initialise it with zeros.

4. Populate each element of the Mandelbrot plot using `mandelbrot()` function. The algorithm of the `mandelbrot()` function is:
   (a) Start with z = 0.
   (b) Iterate $z = z * z + c$ while $|z| <= 2$ and iteration count ¡ 100.
   (c) Return the number of iterations performed (escape-time). If the point does not escape within 100, the return value equals 100.
5. Plot the visualisation of the Mandelbrot plot using `matplotlib`.

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Part (b) - Newton's Fractal

The visualisation presented using the Newton-Raphson iteration of the equation $f(z) = z^3 + 1$ is as follows:



Algorithm

The steps followed to obtain the above visualisation are as follows:

1. Initialise the parameters of the grid, this includes the following values:
   - width
   - height
   - minimum and maximum values of x
   - minimum and maximum values of y
2. Generate two arrays that correspond to the real and complex values of x_0 respectively. We shall test every combination of these two arrays to generate the visualisation.
3. Create a two-dimensional array for storing the values of the fractal plot and initialise it with zeros.

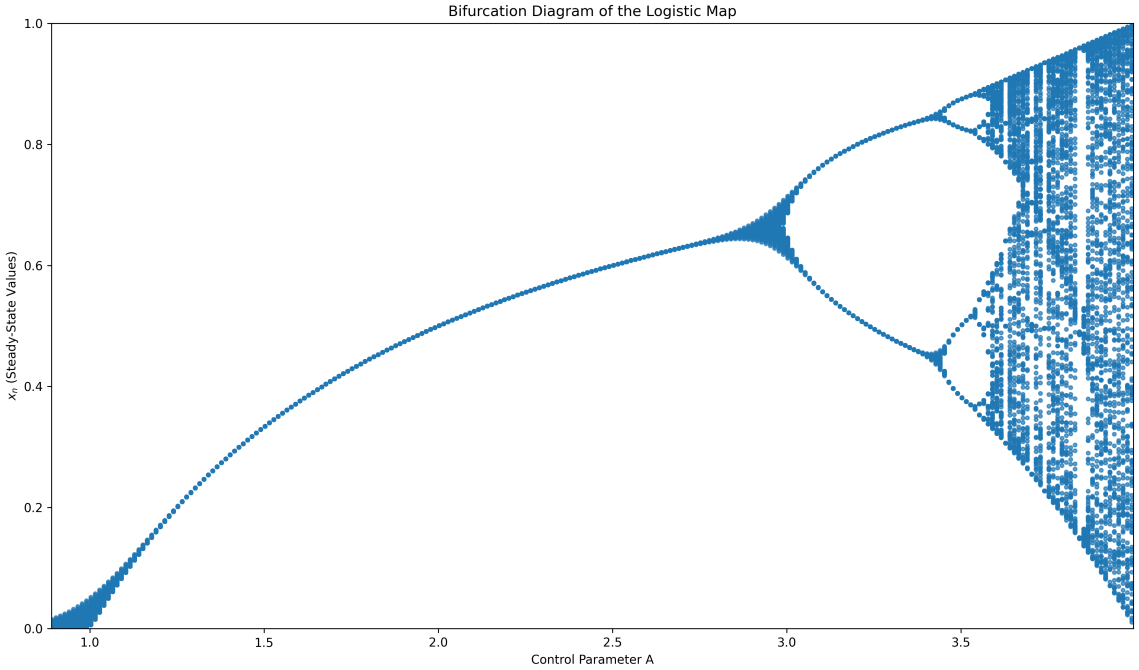
4. Populate each element of the fractal plot using the `newton_fractal()` function. This function returns the sequence of iterations of x_n, until either x_n is close enough to one of the three roots, or n has crossed the maximum number of iterations.

5. Plot the visualisation of the Mandelbrot plot using `matplotlib`.

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

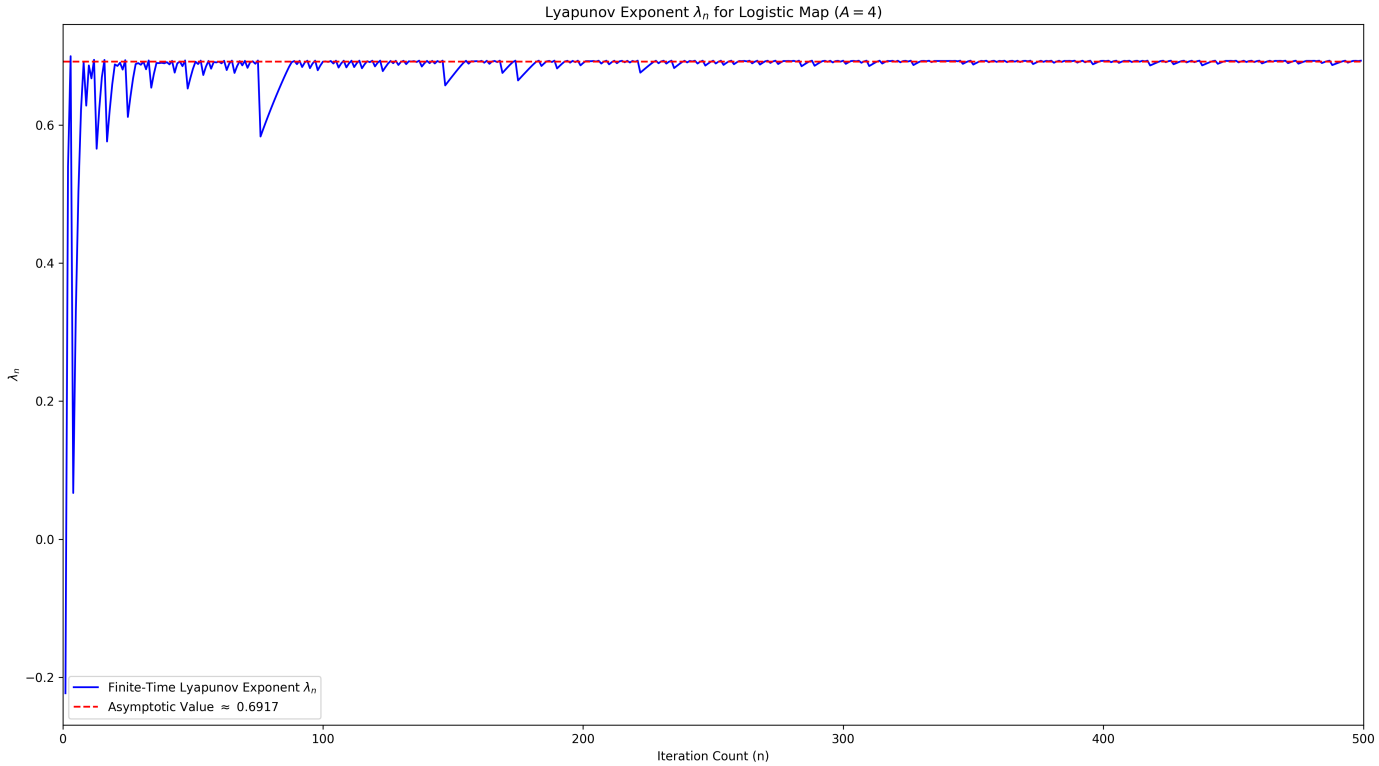
Part (c) - Logistic Map and the Onset of Chaos

The bifurcation diagram is as follows:



Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

The Lyapunov Logistic map is as follows:



The asymptotic value of λ comes out to approximately 0.692, as has been visualised by the red line in the above graph.

Algorithm

The steps followed to obtain the above visualisation are as follows:

1. Initialise the parameters as specified in the question. These parameters include:
    - minimum and maximum values of A
    - step size
    - maximum number of iterations
    - number of transient iterations to exclude from the plot
2. For each value of A, generate a sequence of x values. Discard the first 15 transient iterations.
3. Plot the bifurcation diagram.
4. Initialise the parameters required for the Lyapunov exponent. These parameters include:
    - the value of A

- x_0
- number of iterations

5. Calculate the values of λ using the sequence of x_n for the x_0 initialised above.

6. Calculate the value of λ_n using the formula, as provided in the question:

$$\lambda_n = \frac{1}{n} \sum_{k=0}^{n-1} \ln |f'(x_k)|$$

7. Plot the Lyapunov Logistic Map

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Solution 3 - Optimizing a multi-link robotic arm

In the given figure, there are four points. One of them is the origin, i.e. $(0, 0)$ and let us call the other points by (x_1, y_1), (x_2, y_2), and (x_3, y_3). Let us first calculate the co-ordinates of (x_1, y_1):

$$\begin{aligned} x_1 &= 0 + l_1 \cos\theta_1 \\ &= l_1 \cos\theta_1 \\ y_1 &= 0 + l_1 \sin\theta_1 \\ &= l_1 \sin\theta_1 \end{aligned}$$

Similarly, let us calculate the expressions for (x_2, y_2) and (x_3, y_3) also:

$$x_2 = x_1 + l_2 \cos(\theta_1 + \theta_2)$$

$$x_2 = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$

$$y_2 = y_1 + l_2 \sin(\theta_1 + \theta_2)$$

$$y_2 = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

$$x_3 = x_2 + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$

$$x_3 = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$

$$y_3 = y_2 + l_2 \sin(\theta_1 + \theta_2 + \theta_3)$$

$$y_3 = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3)$$

In conclusion:

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

$$x_3 = l_1 \cos\theta + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$

$$y_3 = l_1 \sin\theta + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3)$$

Hence, we have proven the expressions obtained in the question.

Now, let us apply the central difference theorem to compute the gradient of the loss function, i.e. $L(\theta)$. The loss function is defined by $L(\theta) = \frac{1}{2}||p(\theta) - p_t||^2$, where $p(\theta) = \begin{bmatrix} x(\theta) \\ y(\theta) \end{bmatrix}$.

The formula for the partial derivative of L with respect to any angle θ_i is:

$$\frac{\partial L}{\partial \theta_i} = \frac{L(\theta_1, ..., \theta_i + h, ..., \theta_3) - L(\theta_1, ..., \theta_i - h, ..., \theta_3)}{2h}$$

Using this formula we can calculate $\nabla L(\theta)$ as follows:

$$\nabla L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} & \frac{\partial L}{\partial \theta_2} & \frac{\partial L}{\partial \theta_3} \end{bmatrix}^T$$

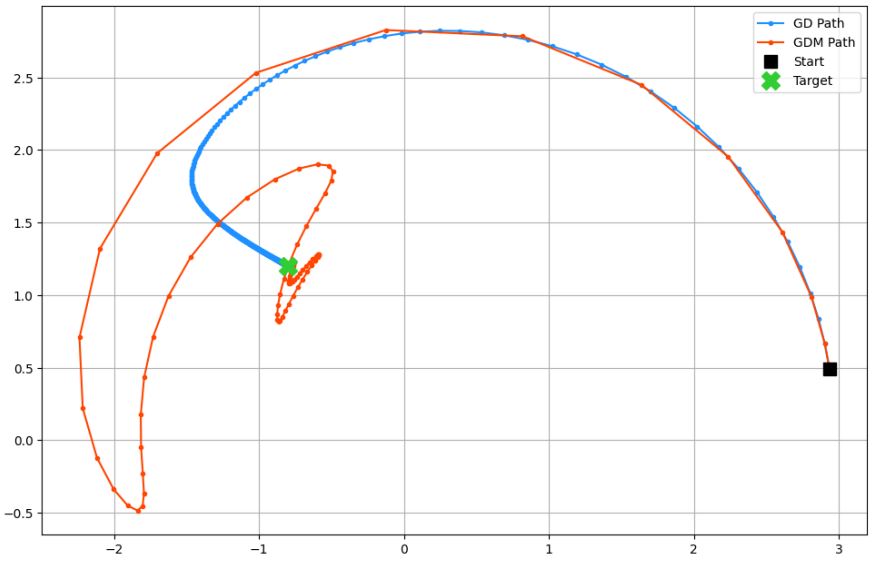
Using this formula, with the Gradient Descent and the Gradient Descent with Momentum formulae, we obtain the following results:

Table 1: Comparison of Gradient Descent Methods

Method	**Iterations**	**Angle**			**Final Position of Arm**		**Euclidean Distance between Target and Final Position**
		θ_1	θ_2	θ_3	x	y	
Gradient Descent	682	30.0877	102.3298	49.5672	-0.8087	1.2049	0.00998
Gradient Descent with Momentum	89	24.9835	109.9527	40.8387	-0.7972	1.2039	0.00484

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

The plot produced is as follows:



As we can see in the above plot, the Gradient Descent actually converges slower, but has relatively lower errors than the Gradient Descent with Momentum method. On the other hand the Gradient Descent with Momentum method converges much faster (in 13% of the iterations, to be precise). However, this method jumps all around the place, and has larger intermediate errors.

Algorithm

To obtain the results, the following set of steps was followed:

1. The variables were initialised as given in the question. These variables include:
    - target co-ordinates
    - lengths of the arms
    - step size, which is required to calculate first derivative using central difference method
    - initial values for the angle
    - the maximum tolerable distance between the endpoint of the arm, and the target
    - the learning rate for gradient descent
2. We then create a few functions. They are as follows:
    - `P(angles)`: Returns $P(\theta)$, i.e. the co-ordinates of the arm's endpoint based on the angles provided.
    - `two_norm_squared(vec)`: Returns the square of the two-norm (Euclidean norm) of the vector `vec`.

- `L(angles)`: Returns $L(\theta)$, i.e. the loss function as specified in the question.
- `nablaL(angles)`: Returns $\nabla L(\theta)$, i.e. the numerical first derivative of the loss function.
- `calculate_euclidean_distance(angles)`: Returns the Euclidean distance from the current position to the target position.

3. Initialise the iteration counter k, required for the loop.
4. Initialise the arrays in which the iteration data will be stored. This data shall be used for plotting the graph.
5. Apply the gradient descent method using the formula provided and iterate until the distance between the arm's endpoint and the target is less than the tolerance specified.
6. The details of the gradient descent are output. This includes:
   - No. of iterations
   - Final angles of the arms
   - Final co-ordinates of the arm's endpoint
   - Final distance between the arm's endpoint and the target
7. Reset the iteration counter k to 0.
8. Initialise the arrays in which the iteration data will be stored. This data shall be used for plotting the graph.
9. Apply the gradient descent with momentum method using the formula provided and iterate until the distance between the arm's endpoint and the target is less than the tolerance specified.
10. The details of the gradient descent with momentum are output. This includes:
    - No. of iterations
    - Final angles of the arms
    - Final co-ordinates of the arm's endpoint
    - Final distance between the arm's endpoint and the target
11. Plot the convergence of both methods and display it.

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Solution 4 - Fourier Series Reconstruction of Letter Outlines

Part 1 - Orthogonality Question [Theoretical]

Our objective is to prove that the set $\{e^{i2\pi kt}\}_{k\in\mathbb{Z}}$ is orthogonal with respect to the inner product $\langle f, g\rangle = \int_0^1 f(t)\overline{g(t)}dt$ on the interval $t \in [0, 1]$. In other words, we have to prove the following relationship:

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \begin{cases} 1, & \text{if } k = m \\ 0, & \text{if } k \neq m \end{cases}$$

We can do this by considering two cases: $k \neq m$ and $k = m$. Let us consider $k \neq m$ first:

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \int_0^1 e^{i2\pi kt}\overline{e^{i2\pi mt}}dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \int_0^1 e^{i2\pi kt}e^{-i2\pi mt}dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \int_0^1 e^{i2\pi kt-i2\pi mt}dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \int_0^1 e^{i2\pi(k-m)t}dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \left[\frac{e^{i2\pi(k-m)t}}{i2\pi(k-m)}\right]_0^1$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \frac{1}{i2\pi(k-m)}[e^{i2\pi(k-m)t}]_0^1$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \frac{1}{i2\pi(k-m)}(e^{i2\pi(k-m)1} - e^{i2\pi(k-m)0})$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \frac{1}{i2\pi(k-m)}(e^{i2\pi(k-m)} - e^{0})$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \frac{e^{i2\pi(k-m)} - 1}{i2\pi(k-m)}$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \frac{e^{i2\pi(k-m)} - 1}{i2\pi(k-m)}$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt}\rangle = \frac{\cos(2\pi(k-m)) + i\sin(2\pi(k-m)) - 1}{i2\pi(k-m)}$$

Because $\cos(2\pi(k-m)) = 1$ and $\sin(2\pi(k-m)) = 0$, as long as k and m are integers (as has been stated in the question), we can perform the following simplification.

$$\langle e^{i2\pi kt}, e^{i2\pi mt} \rangle = \frac{1 + i \cdot 0 - 1}{i2\pi(k-m)}$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt} \rangle = \frac{1 - 1}{i2\pi(k-m)}$$

$$\langle e^{i2\pi kt}, e^{i2\pi mt} \rangle = 0$$

Now, let us check the case where $k = m$:

$$\langle e^{i2\pi kt}, e^{i2\pi kt} \rangle = \int_0^1 e^{i2\pi kt} \overline{e^{i2\pi kt}} dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi kt} \rangle = \int_0^1 e^{i2\pi kt} e^{-i2\pi kt} dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi kt} \rangle = \int_0^1 e^{i2\pi kt - i2\pi kt} dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi kt} \rangle = \int_0^1 e^0 dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi kt} \rangle = \int_0^1 1 dt$$

$$\langle e^{i2\pi kt}, e^{i2\pi kt} \rangle = [t]_0^1$$

$$\langle e^{i2\pi kt}, e^{i2\pi kt} \rangle = 1$$

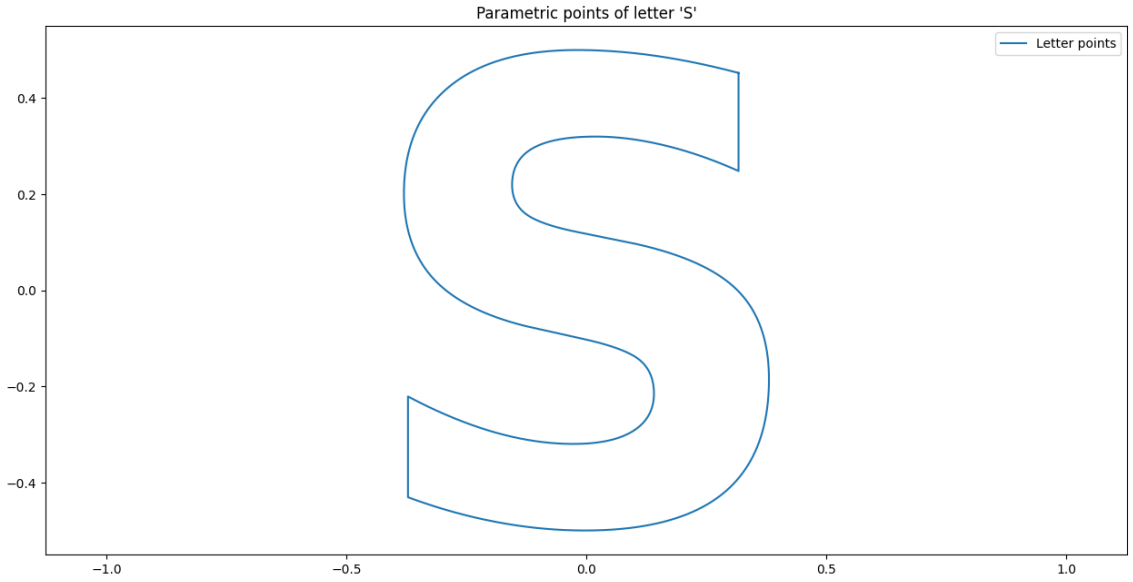
Hence, we have proved that:

$$\langle e^{i2\pi kt}, e^{i2\pi mt} \rangle = \begin{cases} 1, & \text{if } k = m \\ 0, & \text{if } k \neq m \end{cases}$$

Part 2 - Data Loading and Preprocessing

Upon loading the data and processing it (as explained in `Supplementary/Generator/readme.rtf`), the following diagram was constructed:
The first letter of my name (in capital) can be seen in the diagram above.

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025



Part 3 - Fourier Coefficient Calculation

Before we start integrating numerically, we have to choose one of the many methods taught in class. I decided to choose trapezium method due to the abundance of points. Let us derive a discrete formula to calculate the integral numerically by using the trapezoidal rule for N intervals, each with width $\Delta t = \frac{1}{N}$:

$$c_k = \int_0^1 z(t) \times e^{-i2\pi kt} dt$$

$$c_k \approx \sum_{j=0}^{N-1} z(t_j) \times e^{-i2\pi kt_j} \times \Delta t$$

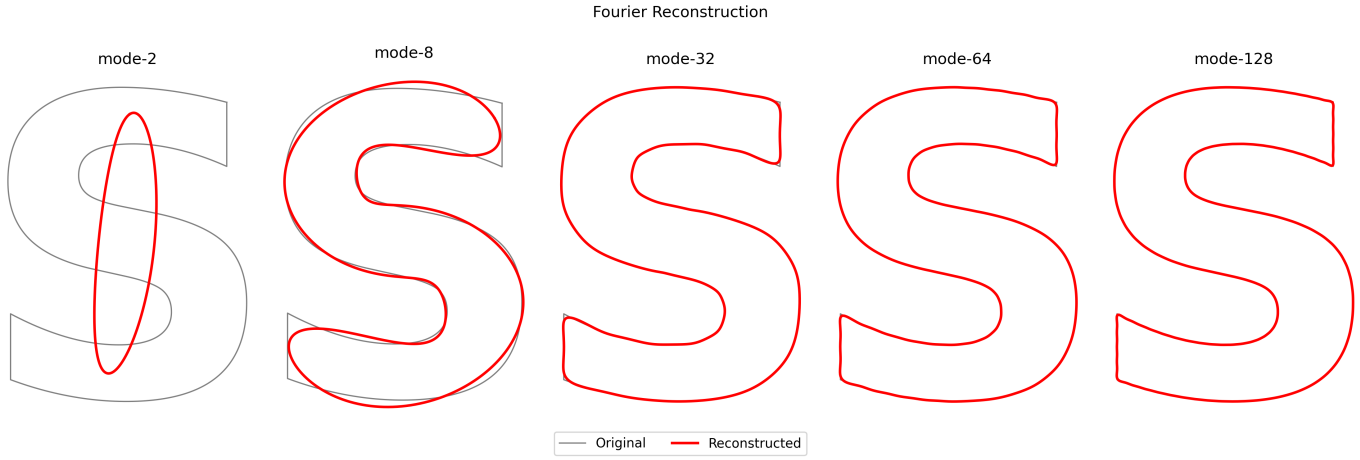
$$c_k \approx \frac{1}{N} \sum_{j=0}^{N-1} z_j \times e^{-i2\pi k\left(\frac{j}{N}\right)}$$

$$c_k \approx \frac{1}{N} \sum_{j=0}^{N-1} z_j \times e^{-\frac{i2\pi kj}{N}}$$

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Part 4 and 5 - Curve Reconstruction and Visualization

The plot obtained is as follows:

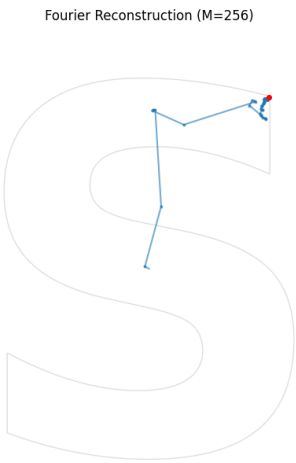


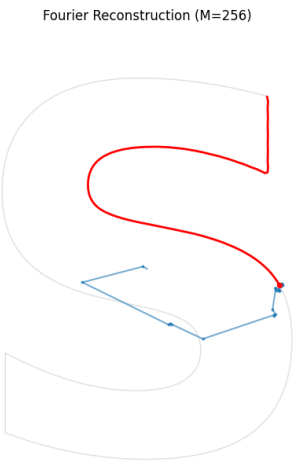
As can be seen in the above diagram, the letter is unrecognisable at $M = 2$. It becomes better at $M = 8$. Even if it does not match the original shape exactly, we can still see that it is an English letter. At $M = 32$, the reconstruction of the letter is very accurate. However, it is a bit inaccurate near the corners. At $M = 64$, even the corners have been smoothened, but the shape overall is a bit s̈haky.̇ Lastly, at $M = 128$, the reconstructed letter matches the original perfectly.

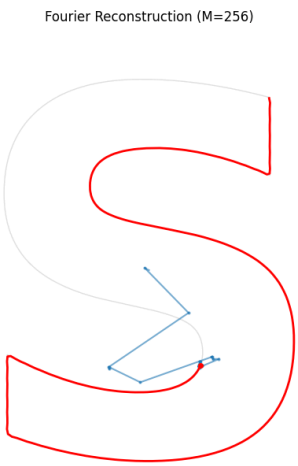
Part 6 - Animation

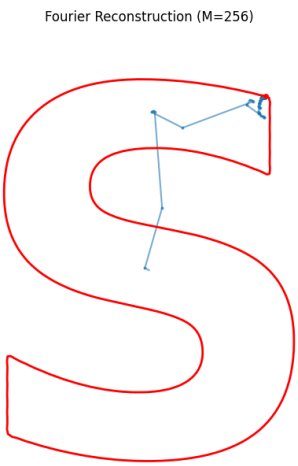
The GIF file generated cannot be attached here. However, the code is capable of generating it. Here are some frames from the GIF file produced:

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025









Algorithm

The set of steps used to produce the above visualisations are as follows:

1. Load the `csv` file produced by the application in `Supplementary/Generator/Windows/dist/letter.exe` using `numpy`.

2. Construct the complex values and plot the original.
3. Compute the Fourier coefficients using the values of M provided in the question. Create subplots for each value of M and plot accordingly.
4. Create the animation as provided in the `Supplementary/Outputs/anim.gif` file.

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Solution 5 - Damped, Driven Harmonic Oscillator

In this question, we are told to solve a second-order differential equation. The differential equation in question is as follows:

$$\frac{d^2x}{dt^2} + 2\gamma\frac{dx}{dt} + \omega_0^2 x = A_0\cos(\omega t)$$

As mentioned in the lecture slides, we have to split the second-order differential equation into two first-order differential equations. This can be done as follows:
Let $v(t) = \frac{dx}{dt}$. Substituting it into the second order differential equation, we get:

$$\begin{aligned}
\frac{d^2x}{dt^2} + 2\gamma\frac{dx}{dt} + \omega_0^2 x &= A_0\cos(\omega t) \\
\frac{d^2x}{dt^2} &= A_0\cos(\omega t) - 2\gamma\frac{dx}{dt} - \omega_0^2 x \\
\frac{d}{dt}\left(\frac{dx}{dt}\right) &= A_0\cos(\omega t) - 2\gamma\frac{dx}{dt} - \omega_0^2 x \\
\frac{d}{dt}(v(t)) &= A_0\cos(\omega t) - 2\gamma v(t) - \omega_0^2 x \\
\frac{dv}{dt} &= A_0\cos(\omega t) - 2\gamma v - \omega_0^2 x
\end{aligned}$$

Now, we have successfully reduced the second-order differential equation to two first-order differential equations:

$$\begin{aligned}
\frac{dx}{dt} &= v \\
\frac{dv}{dt} &= A_0\cos(\omega t) - 2\gamma v - \omega_0^2 x
\end{aligned}$$

Part 1 - Numerical Implementation

Euler Method

Euler method is a numerical method to solve first order equations. Euler method is defined as follows [1]:

$$y_{n+1} = y_n + hf(x_n, y_n)$$

Let us apply this into our system of differential equations. The first differential equation is transformed as follows:

$$x_{i+1} = x_i + \Delta t \cdot f(x_i, v_i)$$

$$x_{i+1} = x_i + \Delta t \cdot v_i$$

The second differential equation is transformed as follows:

$$v_{i+1} = v_i + \Delta t \cdot f(v_i, t_i, x_i)$$

$$v_{i+1} = v_i + \Delta t(A_0 \cos(\omega t_i) - 2\gamma v_i - \omega_0^2 x_i)$$

Runge-Kutta (RK-4) method

Let us put the differential equations into vector form:

$$f(t, \mathbf{y}) = \frac{d\mathbf{y}}{dt}, \text{where } y = \begin{bmatrix} x \\ v \end{bmatrix}, \text{and } f(t, \mathbf{y}) = \begin{bmatrix} v \\ A_0 \cos(\omega t) - 2\gamma v - \omega_0^2 x \end{bmatrix}$$

The RK-4 method is used to calculate the next iteration. The method is as follows:

$$k_1 = f(t_i, \mathbf{y}_i)$$

$$k_2 = f\left(t_i + \frac{\Delta t}{2}, \mathbf{y_i} + \frac{\Delta t}{2} k_1\right)$$

$$k_3 = f\left(t_i + \frac{\Delta t}{2}, \mathbf{y_i} + \frac{\Delta t}{2} k_2\right)$$

$$k_4 = f(t_i + \Delta t, \mathbf{y_i} + \Delta t k_3)$$

$$\mathbf{y_{i+1}} = \mathbf{y_i} + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Results

The table depicting the results from both methods is as follows:

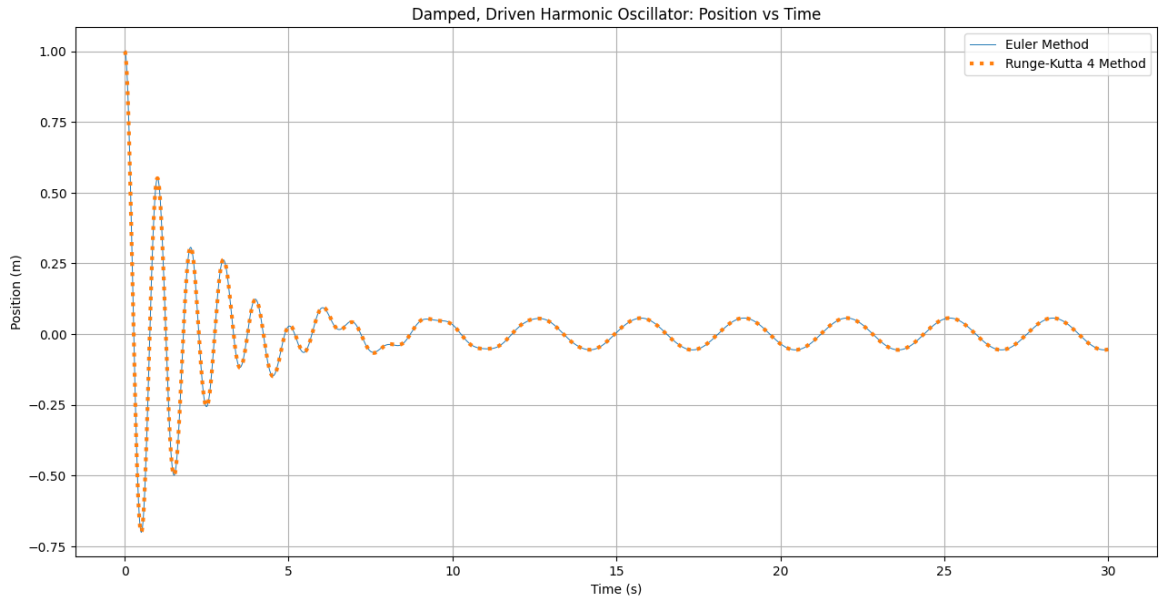
Name: Suhas Kamath
SR No: 06-18-01-10-51-25-1-25945
Email ID: suhaskamath@iisc.ac.in
Date: November 8, 2025

Assignment No: Assignment 4 (Final Project)
Course Code: DS288/UMC202
Course Name: Numerical Methods
Term: AUG 2025

Time	**Euler Method**		**Runge-Kutta 4 Method**	
t **(s)**	x **(m)**	v **(m/s)**	x **(m)**	v **(m/s)**
0.00	1.0000	0.0000	1.0000	0.0000
5.00	0.0270	0.0620	0.0276	0.1039
10.00	0.0330	−0.0982	0.0320	−0.0921
15.00	0.0049	0.1122	0.0060	0.1130
20.00	−0.0342	−0.0886	−0.0351	−0.0879
25.00	0.0530	0.0366	0.0534	0.0356
30.00	−0.0547	0.0271	−0.0545	0.0282

Part 2 - Visualization of Solution

The plot below shows the oscillator's displacement $x(t)$ versus time t for both the Euler and Runge-Kutta RK4 methods:



Part 3 - Analysis of Motion

By examining the plot, we can clearly identify two distinct phases of motion:

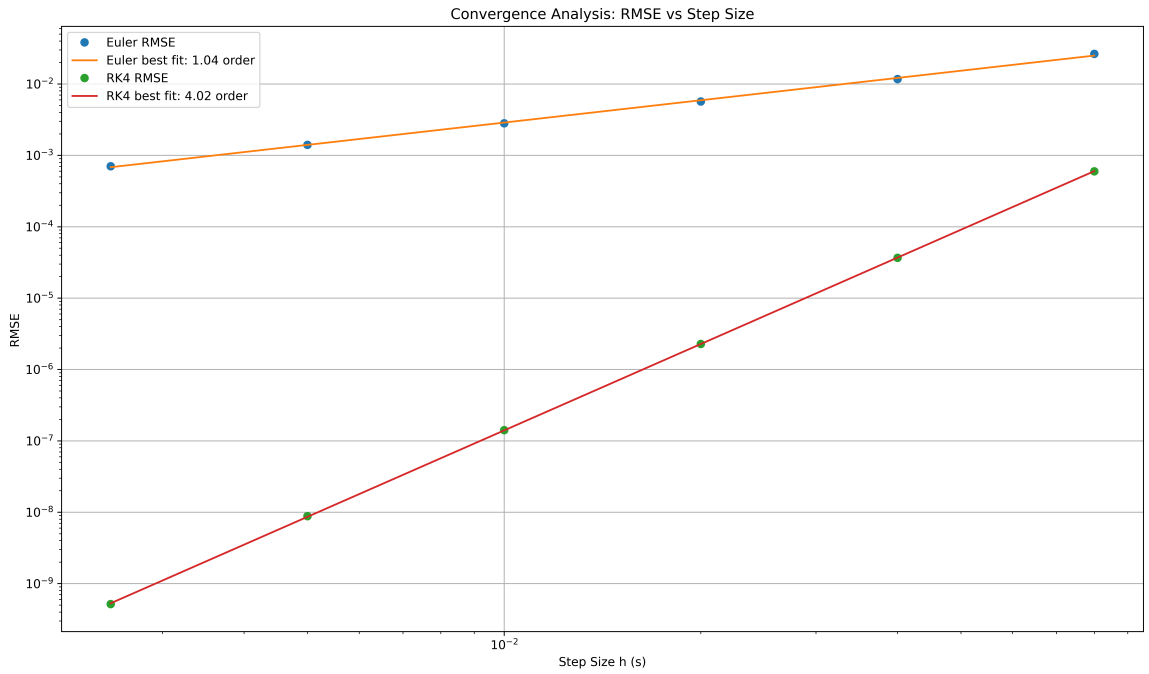
1. Transient Behaviour: For the first ≈ 10 seconds, the motion is irregular and influenced by the initial conditions ($x = 1, v = 0$).
2. Steady-State Behaviour: After the initial transients decay, the system settles into a stable, periodic oscillation. In this phase, the motion is oscillating at the driving frequency $\omega = 2.0$ rad/s.

Furthermore, we can also notice that the transient motion decays as the value of x increases. This is due to the damping term in the equation. The damping term is as follows:

$$2\gamma \frac{dx}{dt}$$

Part 4 - Convergence analysis

The convergence plot is as follows:



We can see in the plot that the Euler's method has a convergence rate of approximately 1.04, and the RK-4 method has a convergence rate of approximately 4.02.

Judging from the plot, it is obvious that the RK-4 method has a much better rate of convergence. However, RK-4 is computationally more expensive, because we have to calculate a lot of k-values, i.e. k_1, k_2, k_3 and k_4.

Hence, it is difficult to conclude which method exactly is better. the Euler method is computationally efficient, whereas the RK-4 method is fast but computationally expensive. Hence, we can state the b̈estälgorithm depends on the application.

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

Algorithm

The set of steps used to produce the results are as follows:

1. Initialise the constants of the differential equation, as provided in the question:
    - ω_0
    - γ
    - A_0
    - ω
    - x_0
    - v_0
2. Create some functions to calculate required parameters:
    - Create a function `get_next_x (x_i, v_i, delta_t)` to obtain the next value of x, i.e. x_{i+1} using the formula given above (used by Euler method).
    - Create a function `get_next_v (v_i, t_i, x_i, delta_t)` to obtain the next value of v, i.e. v_{i+1} using the formula given above (used by Euler method).
    - Create a function `f(t, y)` to obtain the next value of v, i.e. v_{i+1} using the formula given above (used by RK-4 method to calculate k-values).
    - Create a function `euler(delta_t)` to calculate the perform the Euler method iterations.
    - Create a function `runge_kutta_4(delta_t)` to calculate the perform the RK-4 method iterations.
3. Calculate the results for both Euler method and RK-4 method. Output the results (once every 500 iterations, as specified by the question).
4. Plot and save a graph comparing the Euler method and RK-4 method.
5. For convergence analysis, initialise the values of Δt that we have to use.
6. For each value of Δt:
    (a) Calculate the results for both Euler method and RK-4 method.
    (b) Get the root mean square error (R.M.S.E) for both methods using the ẗrue valueëcalculated using RK-4 method and a $\Delta t = 0.00125$.
    (c) Output the R.M.S.E.
7. The linear line of best fit for each method is calculated using a simple function.
8. Plot and save the convergence analysis with the line of best fit.

Name:	Suhas Kamath	**Assignment No:**	Assignment 4 (Final Project)
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS288/UMC202
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 8, 2025	**Term:**	AUG 2025

References

[1] BEHERA, R. *Lecture Notes*. Indian Institute of Science, Bengaluru, 2025.

[2] BURDEN, R., AND FAIRES, J. D. *Numerical Analysis*. PWS-Kent Publishing Company, 1993.