Name:	Aneesh Panchal	**Homework No:**	Homework 1
SR No:	06-18-01-10-12-24-1-25223	**Course Code:**	DS288
Email ID:	aneeshp@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	August 23, 2024	**Term:**	AUG 2024

Equations & Formulas Required

$$J_{n-1}(x) + J_{n+1}(x) = \frac{2n}{x} J_n(x) \tag{1}$$

$$Absolute\_Error = |J_{True} - J_{Predicted}| \tag{2}$$

$$Relative\_Error = \left| \frac{J_{True} - J_{Predicted}}{J_{True}} \right| \tag{3}$$

Solution 1

Forward Computation: $J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x)$

For $x = 1$, absolute error for $J_{10}(1)$ is 560.55331 and relative error is 2.13088E+12.

n	$J_{True}(1)$	$J_{Predicted}(1)$	$Absolute\_Error$	$Relative\_Error$
0 (IC1)	0.765197687	0.76519	7.68656E-06	1.00452E-05
1 (IC2)	0.440050586	0.44005	5.8574E-07	1.33107E-06
2	0.114903485	0.11491	6.51507E-06	5.67004E-05
3	0.019563354	0.01959	2.6646E-05	0.001362037
4	0.002476639	0.00263	0.000153361	0.061923049
5	0.000249758	0.00145	0.001200242	4.80562611
6	2.09383E-05	0.01187	0.011849062	565.902683
7	1.50233E-06	0.14099	0.140988498	93846.8181
8	9.42234E-08	1.96199	1.96198991	20822736.6
9	5.24925E-09	31.25085	31.25085	5953393140
10	2.63062E-10	560.55331	560.55331	2.13088E+12

For $x = 5$, absolute error for $J_{10}(5)$ is 0.000117453 and relative error is 0.080019827.

n	$J_{True}(5)$	$J_{Predicted}(5)$	$Absolute\_Error$	$Relative\_Error$
0 (IC1)	-0.177596771	-0.17759	6.77131E-06	3.81274E-05
1 (IC2)	-0.327579138	-0.32757	9.13759E-06	2.78943E-05
2	0.046565116	0.046562	3.11628E-06	6.6923E-05
3	0.364831231	0.3648196	1.16306E-05	3.18794E-05
4	0.39123236	0.39122152	1.08405E-05	2.77085E-05
5	0.261140546	0.261134832	5.71412E-06	2.18814E-05
6	0.131048732	0.131048144	5.8778E-07	4.4852E-06
7	0.05337641	0.053380714	4.30344E-06	8.06245E-05
8	0.018405217	0.018417854	1.26374E-05	0.000686622
9	0.005520283	0.005556419	3.61363E-05	0.006546099
10	0.001467803	0.001585256	0.000117453	0.080019827

For $x = 50$, absolute error for $J_{10}(50)$ is 8.86138E-07 and relative error is 7.78353E-06.

n	$J_{True}(50)$	$J_{Predicted}(50)$	$Absolute\_Error$	$Relative\_Error$
0 (IC1)	0.055812328	0.055812	3.27669E-07	5.87091E-06
1 (IC2)	-0.097511828	-0.097511	8.28125E-07	8.49256E-06
2	-0.059712801	-0.05971244	3.60794E-07	6.04216E-06
3	0.092734804	0.092734005	7.99262E-07	8.61879E-06
4	0.070840977	0.070840521	4.56706E-07	6.44692E-06
5	-0.081400248	-0.081399522	7.26189E-07	8.92122E-06
6	-0.087121027	-0.087120425	6.01943E-07	6.90928E-06
7	0.060491201	0.06049062	5.81723E-07	9.61665E-06
8	0.104058563	0.104057798	7.64822E-07	7.34992E-06
9	-0.027192461	-0.027192124	3.36978E-07	1.23923E-05
10	-0.113847849	-0.113846963	8.86138E-07	7.78353E-06

Name: Aneesh Panchal
SR No: 06-18-01-10-12-24-1-25223
Email ID: aneeshp@iisc.ac.in
Date: August 23, 2024

Homework No: Homework 1
Course Code: DS288
Course Name: Numerical Methods
Term: AUG 2024

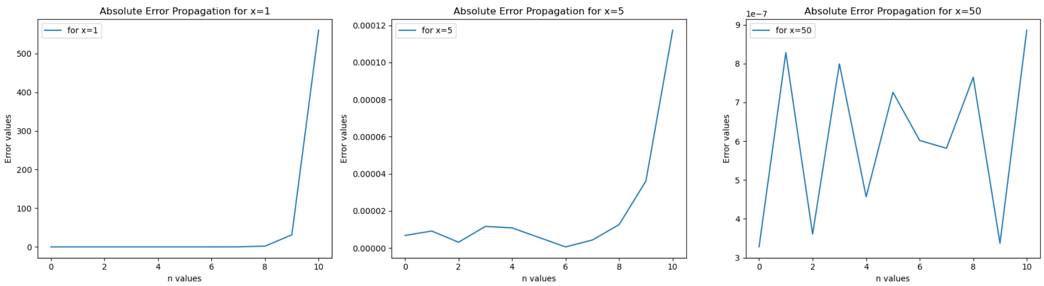


Figure 1: Absolute Error Propagation for computations in Forward direction.

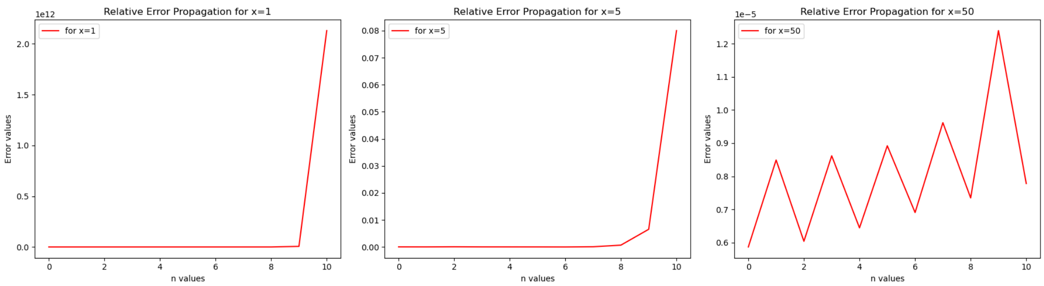


Figure 2: Relative Error Propagation for computations in Forward direction.

It is clear from the Tables for absolute errors (for $x = 1, 5, 50$) and Fig. 1 that in forward direction propagation, absolute error shows a predictable behaviour which is discussed as follows for every value of x given,

1. Error Propagation for $x = 1$,
   (a) for all n values, the error propagation shows the ***exponential growth*** (which can also be seen in Fig. 1) as $n = 1, 2, \ldots, 9 \geq 1 = x$ whose proof is given in Solution **3** with the specific equation (10).
2. Error Propagation for $x = 5$,
   (a) for $n = 1, 2, \ldots, 5 \leq 5 = x$ values, the error propagation shows ***oscillatory behaviour*** (which can also be seen in Fig. 1) for $n = 1, 2, \ldots, 5$ whose proof is given in Solution **3** with the specific equation (12).
   (b) for $n = 6, 7, \ldots, 9 > 5 = x$ values, the error propagation shows ***exponential growth*** (which can also be seen in Fig. 1) for $n = 6, 7, \ldots, 9$ whose proof is given in Solution **3** with the specific equation (10).
3. Error Propagation for $x = 50$,
   (a) for all n values, the error propagation shows the ***oscillatory behaviour*** (which can also be seen in Fig. 1) as $n = 1, 2, \ldots, 9 < 50 = x$ whose proof is given in Solution **3** with the specific equation (12).

Analysis [Forward Propagation]:

1. for $x = 1$, error propagation follows **Exponential Growth** for all n and hence, the absolute error for calculating $J_{10}(1)$ raises to 560.55331 from initial error of $5.8574E - 07$.
2. for $x = 5$, error propagation shows **Oscillatory Behaviour** upto $n = 5$ and afterwards error propagation follows **Exponential Growth**. Hence, the absolute error for calculating $J_{10}(5)$ raises to $1.17453E - 04$ from $9.13759E - 06$.
3. for $x = 50$, error propagation shows **Oscillatory Behaviour** for all n and hence, the absolute error for calculating $J_{10}(50)$ oscillates and finally reaches $8.86138E - 07$ from initial error of $8.28125E - 07$.

NOTE: 1). Initially difference equation is applied on $n = 1$ hence, initial error is δ_1.
2). Python code used for calculations and plotting forward propagation absolute and relative errors, is provided in Appendix.

Name:	Aneesh Panchal	**Homework No:**	Homework 1
SR No:	06-18-01-10-12-24-1-25223	**Course Code:**	DS288
Email ID:	aneeshp@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	August 23, 2024	**Term:**	AUG 2024

Solution 2

$$\text{Backward Computation: } J_{n-1}(x) = \frac{2n}{x} J_n(x) - J_{n+1}(x)$$

For $x = 1$, absolute error for $J_0(1)$ is 7.32304E-06 and relative error is 9.57012E-06.

n	$J_{True}(1)$	$J_{Predicted}(1)$	$Absolute\_Error$	$Relative\_Error$
0	0.765197687	0.765190364	7.32304E-06	9.57012E-06
1	0.440050586	0.440046374	4.21133E-06	9.57011E-06
2	0.114903485	0.114902385	1.09964E-06	9.57010E-06
3	0.019563354	0.019563167	1.87224E-07	9.57014E-06
4	0.002476639	0.002476615	2.37017E-08	9.57012E-06
5	0.000249758	0.000249755	2.39021E-09	9.57011E-06
6	2.09383E-05	2.09381E-05	2.00382E-10	9.57010E-06
7	1.50233E-06	1.50231E-06	1.43774E-11	9.57009E-06
8	9.42234E-08	9.42225E-08	9.01726E-13	9.57008E-06
9 (IC2)	5.24925E-09	5.24920E-09	5.01799E-14	9.55944E-06
10 (IC1)	2.63062E-10	2.63060E-10	1.51237E-15	5.74911E-06

For $x = 5$, absolute error for $J_0(5)$ is 2.88572E-06 and relative error is 1.62487E-05.

n	$J_{True}(5)$	$J_{Predicted}(5)$	$Absolute\_Error$	$Relative\_Error$
0	-0.177596771	-0.177593886	2.88572E-06	1.62487E-05
1	-0.327579138	-0.327573815	5.32214E-06	1.62469E-05
2	0.046565116	0.046564359	7.56867E-07	1.62539E-05
3	0.364831231	0.364825303	5.92763E-06	1.62476E-05
4	0.39123236	0.391226004	6.35630E-06	1.62469E-05
5	0.261140546	0.261136304	4.24244E-06	1.62458E-05
6	0.131048732	0.131046603	2.12858E-06	1.62427E-05
7	0.05337641	0.053375544	8.66156E-07	1.62273E-05
8	0.018405217	0.01840492	2.96655E-07	1.61180E-05
9 (IC2)	0.005520283	0.00552020	8.31385E-08	1.50606E-05
10 (IC1)	0.001467803	0.00146780	2.64730E-09	1.80358E-06

For $x = 50$, absolute error for $J_0(50)$ is 5.05209E-06 and relative error is 9.05193E-05.

n	$J_{True}(50)$	$J_{Predicted}(50)$	$Absolute\_Error$	$Relative\_Error$
0	0.055812328	0.055807276	5.05209E-06	9.05193E-05
1	-0.097511828	-0.097505885	5.94295E-06	6.09460E-05
2	-0.059712801	-0.059707511	5.28981E-06	8.85876E-05
3	0.092734804	0.092729284	5.51977E-06	5.95221E-05
4	0.070840977	0.070835025	5.95219E-06	8.40218E-05
5	-0.081400248	-0.08139568	4.56742E-06	5.61106E-05
6	-0.087121027	-0.087114161	6.86567E-06	7.88061E-05
7	0.060491201	0.060488282	2.91966E-06	4.82659E-05
8	0.104058563	0.10405088	7.68317E-06	7.38351E-05
9 (IC2)	-0.027192461	-0.027192	4.61044E-07	1.69548E-05
10 (IC1)	-0.113847849	-0.11384	7.84915E-06	6.89442E-05

It is clear from the Tables for absolute errors (for $x = 1, 5, 50$) and Fig. 3 that in backward direction propagation, absolute error shows a predictable behaviour which is discussed as follows for every value of x given,

1. Error Propagation for $x = 1$,

   (a) for all n values, the error propagation shows the ***exponential growth*** (which can also be seen in Fig. 3) as $n = 9, 8, \ldots, 1 \geq 1 = x$ whose proof is given in Solution **3** with the specific equation (19).

Name: Aneesh Panchal
SR No: 06-18-01-10-12-24-1-25223
Email ID: aneeshp@iisc.ac.in
Date: August 23, 2024

Homework No: Homework 1
Course Code: DS288
Course Name: Numerical Methods
Term: AUG 2024

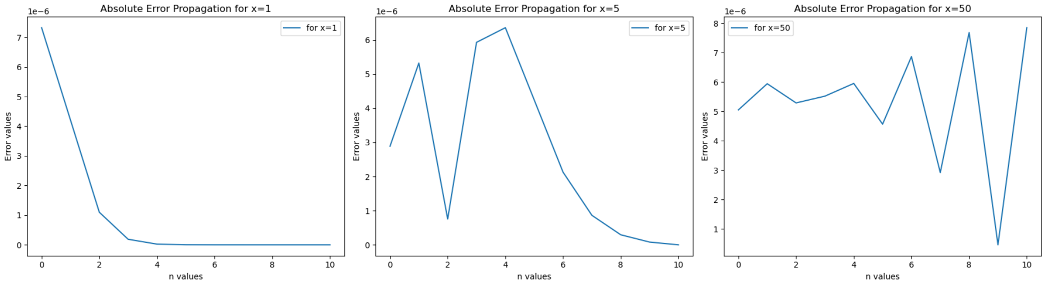


Figure 3: Absolute Error Propagation for computations in Backward direction.

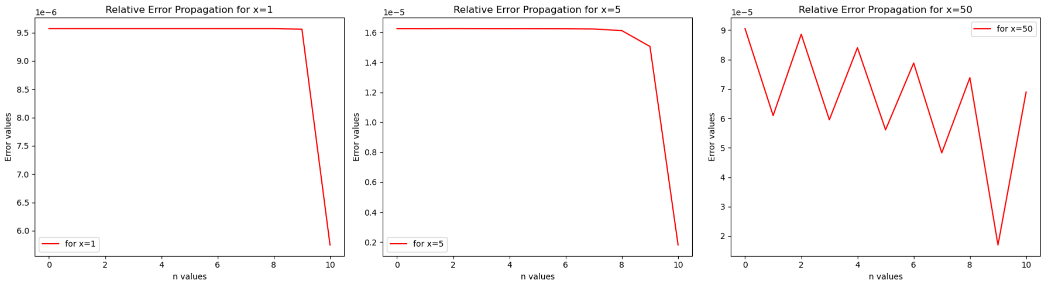


Figure 4: Relative Error Propagation for computations in Backward direction.

2. Error Propagation for $x = 5$,
   (a) for $n = 9, 8, \ldots, 6 > 5 = x$ values, the error propagation shows ***exponential growth*** (which can also be seen in Fig. 3) for $n = 9, 8, \ldots, 6$ whose proof is given in Solution **3** with the specific equation (19).
   (b) for $n = 5, 4, \ldots, 1 \leq 5 = x$ values, the error propagation shows ***oscillatory behaviour*** (which can also be seen in Fig. 3) for $n = 5, 4, \ldots, 1$ whose proof is given in Solution **3** with the specific equation (21).
3. Error Propagation for $x = 50$,
   (a) for all n values, the error propagation shows the ***oscillatory behaviour*** (which can also be seen in Fig. 3) as $n = 9, 8, \ldots, 1 < 50 = x$ whose proof is given in Solution **3** with the specific equation (21).

Analysis [Backward Propagation]:

1. for $x = 1$, error propagation follows **Exponential Growth** for all n and hence, the absolute error for calculating $J_0(1)$ raises to $7.32304E-06$ from initial error of $5.01799E-14$.
2. for $x = 5$, error propagation follows **Exponential Growth** upto $n = 5$ and afterwards error propagation shows **Oscillatory Behaviour**. Hence, the absolute error for calculating $J_0(5)$ raises to $2.88572E-06$ from $8.31385E-08$.
3. for $x = 50$, error propagation shows **Oscillatory Behaviour** for all n and hence, the absolute error for calculating $J_0(50)$ oscillates and finally reaches $5.05209E-06$ from initial error of $4.61044E-07$.

Forward vs Backward Propagation ? − Comparison using Relative Errors
Forward and Backward error propagation both shows same behaviour and comparable results for **oscillatory phases**. However, for **exponential growth phases** it is clear from relative errors values, Fig. 2 and Fig. 4 that backward error propagation outperformed forward error propagation. **It is due to the exponential growth factor terms in equation** (10) **and equation** (19)**, both of which tends to reduce as n reduces and increase as n increases.** Hence, in forward propagation n increases which results in increasing error with every successive term and in backward n decreases which results in somehow reducing the error with every successive iteration.

NOTE: 1). Initially difference equation is applied on $n = 9$ hence, initial error is δ_9.
2). Python code used for calculations and plotting backward propagation absolute and relative errors, is provided in Appendix.

Name: Aneesh Panchal
SR No: 06-18-01-10-12-24-1-25223
Email ID: aneeshp@iisc.ac.in
Date: August 23, 2024

Homework No: Homework 1
Course Code: DS288
Course Name: Numerical Methods
Term: AUG 2024

Solution 3 – Yes, the error propagation can be analyzed using difference equations

Findings for forward propagation and their approximate behaviour is explained in Solution **1** using Tables and Fig. 1 and 2.

Forward Propagation Analysis

Assume that the errors incorporated due to precision as ϵ and error due to propagation be δ. Hence, for forward propagation, we get equation (1) as (for a fixed x),

$$J_{n+1} + \delta_{n+1} = \frac{2n}{x}(J_n + \delta_n) - (J_{n-1} + \delta_{n-1}) + \epsilon \tag{4}$$

$$\implies \delta_{n+1} = \frac{2n}{x}\delta_n - \delta_{n-1} + \epsilon \tag{5}$$

Assumptions:

1. Assume full precision operations for a particular machine, i.e. $\epsilon = 0$. Hence, the error at every n is due to propagation of initial error which is δ_1.
2. Assume the solution of equation (5) of the form $\delta_n \propto \lambda^n$

Putting these assumptions we get,

$$\delta_{n+1} = \frac{2n}{x}\delta_n - \delta_{n-1} \tag{6}$$

$$\implies \lambda^{n+1} = \frac{2n}{x}\lambda^n - \lambda^{n-1} \implies \lambda^2 - \frac{2n}{x}\lambda + 1 = 0 \tag{7}$$

The solutions of the equation (7) are,

$$\lambda = \frac{n}{x} + \sqrt{\frac{n^2}{x^2} - 1}, \quad \frac{n}{x} - \sqrt{\frac{n^2}{x^2} - 1} \tag{8}$$

Hence, using the assumptions, the propagation error can be computed as,

$$\delta_n(x) = C_1\left(\frac{n}{x} + \sqrt{\frac{n^2}{x^2} - 1}\right)^n + C_2\left(\frac{n}{x} - \sqrt{\frac{n^2}{x^2} - 1}\right)^n \qquad \text{where, } C_1, C_2 \text{ are constants.} \tag{9}$$

Here, it is clear that 3 cases arises for the error propagation analysis,

1. **Case 1:** when $n > x$
   In this case both of the values of λ are positive. Hence, the error shows ***exponential behaviour***.

$$\delta_n(x) = C_1 \underbrace{\left(\frac{n}{x} + \sqrt{\frac{n^2}{x^2} - 1}\right)^n}_{\text{exponential growth}} + C_2 \underbrace{\left(\frac{n}{x} - \sqrt{\frac{n^2}{x^2} - 1}\right)^n}_{\text{decay}} \qquad \text{where, } n > x \tag{10}$$

2. **Case 2:** when $n = x$
   In this case both of the values of λ are 1. Hence the error shows ***constant behaviour***.

$$\delta_n(x) = C_1 + C_2 \qquad \text{where, } n = x \tag{11}$$

3. **Case 3:** when $n < x$
   In this case both of the values of λ are complex. Hence, the error shows ***oscillatory behaviour***.

$$\delta_n(x) = C_1\left(\frac{n}{x} + \iota\sqrt{1 - \frac{n^2}{x^2}}\right)^n + C_2\left(\frac{n}{x} - \iota\sqrt{1 - \frac{n^2}{x^2}}\right)^n \qquad \text{where, } n < x$$

$$\implies \delta_n(x) = (C_1 + C_2)\cos n\theta + (C_1 - C_2)\,\iota\sin n\theta \quad \text{where, } \theta = \cos^{-1}\left(\frac{n}{x}\right) \tag{12}$$

Analysis [Forward Propagation]:

1. when $n > x$, Error Propagation shows **Exponential** behaviour.
2. when $n = x$, Error Propagation shows **Constant** behaviour.
3. when $n < x$, Error Propagation shows **Oscillatory** behaviour.

Findings for backward propagation and their approximate behaviour is explained in Solution **2** using Tables and Fig. 3 and 4.

Backward Propagation Analysis

Assume that the errors incorporated due to precision as ϵ and error due to propagation be δ. Hence, for backward propagation, we get equation (1) as (for a fixed x),

$$J_{n-1} + \delta_{n-1} = \frac{2n}{x}(J_n + \delta_n) - (J_{n+1} + \delta_{n+1}) + \epsilon \tag{13}$$

$$\implies \delta_{n-1} = \frac{2n}{x}\delta_n - \delta_{n+1} + \epsilon \tag{14}$$

Assumptions:

1. Assume full precision operations for a particular machine, i.e. $\epsilon = 0$. Hence, the error at every n is due to propagation of initial error which is δ_9.
2. Assume the solution of equation (14) of the form $\delta_n \propto \left(\frac{1}{\lambda}\right)^n$

Putting these assumptions we get,

$$\delta_{n-1} = \frac{2n}{x}\delta_n - \delta_{n+1} \tag{15}$$

$$\implies \lambda^{1-n} = \frac{2n}{x}\lambda^{-n} - \lambda^{-1-n} \implies \lambda^2 - \frac{2n}{x}\lambda + 1 = 0 \tag{16}$$

The solutions of the equation (16) are,

$$\lambda = \frac{n}{x} + \sqrt{\frac{n^2}{x^2} - 1}, \quad \frac{n}{x} - \sqrt{\frac{n^2}{x^2} - 1} \tag{17}$$

Hence, using the assumptions, the propagation error can be computed as,

$$\delta_n(x) = C_1\left(\frac{n}{x} + \sqrt{\frac{n^2}{x^2} - 1}\right)^{-n} + C_2\left(\frac{n}{x} - \sqrt{\frac{n^2}{x^2} - 1}\right)^{-n} \qquad \text{where, } C_1, C_2 \text{ are constants.} \tag{18}$$

Here, it is clear that 3 cases arises for the error propagation analysis,

1. **Case 1:** when $n > x$
   In this case both of the values of λ are positive. Hence, the error shows ***exponential behaviour***.

$$\delta_n(x) = C_1\underbrace{\left(\frac{n}{x} + \sqrt{\frac{n^2}{x^2} - 1}\right)^{-n}}_{\text{decay}} + C_2\underbrace{\left(\frac{n}{x} - \sqrt{\frac{n^2}{x^2} - 1}\right)^{-n}}_{\text{exponential growth}} \qquad \text{where, } n > x \tag{19}$$

2. **Case 2:** when $n = x$
   In this case both of the values of λ are 1. Hence the error shows ***constant behaviour***.

$$\delta_n(x) = C_1 + C_2 \qquad \text{where, } n = x \tag{20}$$

3. **Case 3:** when $n < x$
   In this case both of the values of λ are complex. Hence, the error shows ***oscillatory behaviour***.

$$\delta_n(x) = C_1\left(\frac{n}{x} + \iota\sqrt{1 - \frac{n^2}{x^2}}\right)^{-n} + C_2\left(\frac{n}{x} - \iota\sqrt{1 - \frac{n^2}{x^2}}\right)^{-n} \qquad \text{where, } n < x$$

$$\implies \delta_n(x) = (C_1 + C_2)\cos n\theta + (C_2 - C_1)\,\iota\sin n\theta \quad \text{where, } \theta = \cos^{-1}\left(\frac{n}{x}\right) \tag{21}$$

Analysis [Backward Propagation]:

1. when $n > x$, Error Propagation shows **Exponential** behaviour.
2. when $n = x$, Error Propagation shows **Constant** behaviour.
3. when $n < x$, Error Propagation shows **Oscillatory** behaviour.

1 Appendix: Assignment 1 Programming

1.1 Ques 1

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Forward Propagation function
def forward(num1, num2, x, iter):
    val = (2 * iter / x) * num1 - num2
    return val

# Function for calculation and plotting of absolute & relative errors
def plot_err(J, J_true):
    abs_err = np.abs(J - J_true)
    rel_err = np.abs((J - J_true) / J_true)

    print("\n")
    print("Absolute Error:\n")
    print(abs_err.astype(float))
    print("\n")
    print("Relative Error:\n")
    print(rel_err.astype(float))
    print("\n")

    # Plotting the absolute error graphs
    fig, axes = plt.subplots(1, 3, figsize=(18, 5))
    for i in range(3):
        axes[i].plot(abs_err[:, i], label=f'for x={x[i]}')
        axes[i].set_xlabel('n values')
        axes[i].set_ylabel('Error values')
        axes[i].set_title(f'Absolute Error Propagation for x={x[i]}')
        axes[i].legend()
    plt.tight_layout()
    plt.show()

    # Plotting the relative error graphs
    fig, axes = plt.subplots(1, 3, figsize=(18, 5))
    for i in range(3):
        axes[i].plot(rel_err[:, i], "r-", label=f'for x={x[i]}')
        axes[i].set_xlabel('n values')
        axes[i].set_ylabel('Error values')
        axes[i].set_title(f'Relative Error Propagation for x={x[i]}')
        axes[i].legend()
    plt.tight_layout()
    plt.show()

# True Values of Bessels Function (Ji(x)) for x = 1,5,50 and i = 0,1,...,10
J_true = np.array([
    [7.6519768656e-01, -1.7759677131e-01, 5.5812327669e-02],
    [4.4005058574e-01, -3.2757913759e-01, -9.7511828125e-02],
    [1.1490348493e-01, 4.6565116278e-02, -5.9712800794e-02],
    [1.9563353983e-02, 3.6483123061e-01, 9.2734804062e-02],
    [2.4766389641e-03, 3.9123236046e-01, 7.0840977282e-02],
```

```python
    [2.4975773021e-04, 2.6114054612e-01, -8.1400247697e-02],
    [2.0938338002e-05, 1.3104873178e-01, -8.7121026821e-02],
    [1.5023258174e-06, 5.3376410156e-02, 6.0491201260e-02],
    [9.4223441726e-08, 1.8405216655e-02, 1.0405856317e-01],
    [5.2492501799e-09, 5.5202831385e-03, -2.7192461044e-02],
    [2.6306151237e-10, 1.4678026473e-03, -1.1384784915e-01]
])

# Predicted Values of Bessels Function
J = np.empty((11, 3), dtype=object)
J[0, 0] = 7.6519e-01
J[1, 0] = 4.4005e-01
J[0, 1] = -1.7759e-01
J[1, 1] = -3.2757e-01
J[0, 2] = 5.5812e-02
J[1, 2] = -9.7511e-02

# x vector
x = [1, 5, 50]

# Number of iterations required
max_iter = 10

# Main loop for predicting J value
for j in range(3):
    for i in range(2, max_iter + 1):
        J[i, j] = forward(J[i-1, j], J[i-2, j], x[j], i-1)

# Printing actual and predicted values of Bessels function
print("\n")
print("True Values:\n")
print(J_true.astype(float))
print("\n")
print("Predicted Values:\n")
print(J.astype(float))

# Plotting the errors
plot_err(J.astype(float), J_true)
```

```
True Values:

[[ 7.65197687e-01 -1.77596771e-01  5.58123277e-02]
 [ 4.40050586e-01 -3.27579138e-01 -9.75118281e-02]
 [ 1.14903485e-01  4.65651163e-02 -5.97128008e-02]
 [ 1.95633540e-02  3.64831231e-01  9.27348041e-02]
 [ 2.47663896e-03  3.91232360e-01  7.08409773e-02]
 [ 2.49757730e-04  2.61140546e-01 -8.14002477e-02]
 [ 2.09383380e-05  1.31048732e-01 -8.71210268e-02]
 [ 1.50232582e-06  5.33764102e-02  6.04912013e-02]
 [ 9.42234417e-08  1.84052167e-02  1.04058563e-01]
 [ 5.24925018e-09  5.52028314e-03 -2.71924610e-02]
 [ 2.63061512e-10  1.46780265e-03 -1.13847849e-01]]
```

Predicted Values:

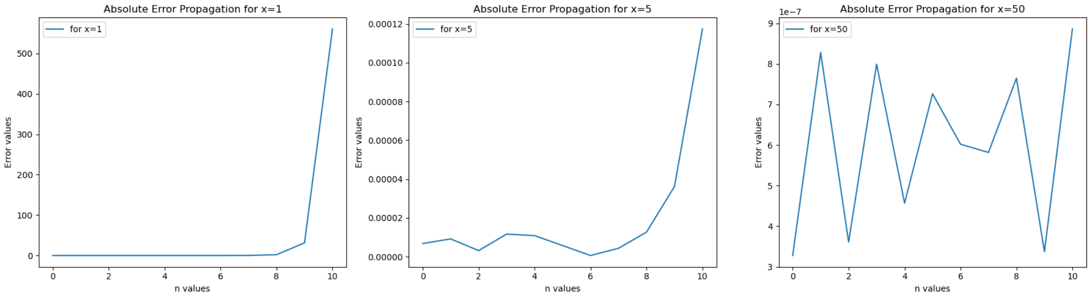
```
[[ 7.65190000e-01 -1.77590000e-01  5.58120000e-02]
 [ 4.40050000e-01 -3.27570000e-01 -9.75110000e-02]
 [ 1.14910000e-01  4.65620000e-02 -5.97124400e-02]
 [ 1.95900000e-02  3.64819600e-01  9.27340048e-02]
 [ 2.63000000e-03  3.91221520e-01  7.08405206e-02]
 [ 1.45000000e-03  2.61134832e-01 -8.13995215e-02]
 [ 1.18700000e-02  1.31048144e-01 -8.71204249e-02]
 [ 1.40990000e-01  5.33807136e-02  6.04906195e-02]
 [ 1.96199000e+00  1.84178541e-02  1.04057798e-01]
 [ 3.12508500e+01  5.55641946e-03 -2.71921241e-02]
 [ 5.60553310e+02  1.58525596e-03 -1.13846963e-01]]
```

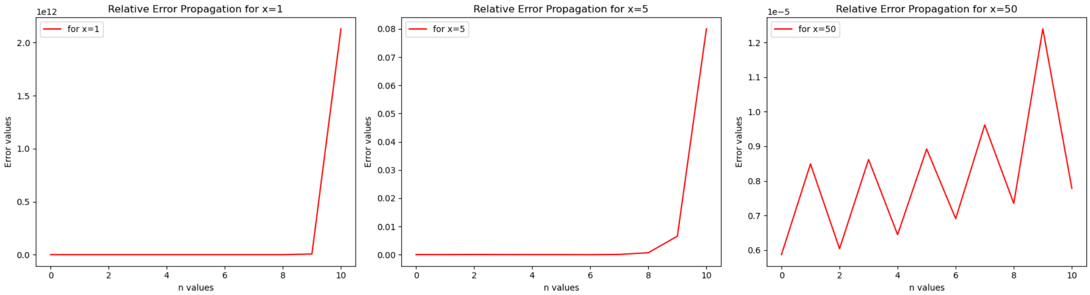
Absolute Error:

```
[[7.68656000e-06 6.77131000e-06 3.27669000e-07]
 [5.85740000e-07 9.13759000e-06 8.28125000e-07]
 [6.51507000e-06 3.11627800e-06 3.60794000e-07]
 [2.66460170e-05 1.16306100e-05 7.99262000e-07]
 [1.53361036e-04 1.08404600e-05 4.56706000e-07]
 [1.20024227e-03 5.71412000e-06 7.26189160e-07]
 [1.18490617e-02 5.87780000e-07 6.01943432e-07]
 [1.40988498e-01 4.30344400e-06 5.81722776e-07]
 [1.96198991e+00 1.26374250e-05 7.64822009e-07]
 [3.12508500e+01 3.61363175e-05 3.36978133e-07]
 [5.60553310e+02 1.17453314e-04 8.86138297e-07]]
```

Relative Error:

```
[[1.00451950e-05 3.81274386e-05 5.87090727e-06]
 [1.33107424e-06 2.78942977e-05 8.49255948e-06]
 [5.67003690e-05 6.69230155e-05 6.04215504e-06]
 [1.36203726e-03 3.18794254e-05 8.61879214e-06]
 [6.19230490e-02 2.77084952e-05 6.44691840e-06]
 [4.80562611e+00 2.18813971e-05 8.92121560e-06]
 [5.65902683e+02 4.48520174e-06 6.90927844e-06]
 [9.38468181e+04 8.06244554e-05 9.61665109e-06]
 [2.08227366e+07 6.86621909e-04 7.34991899e-06]
 [5.95339314e+09 6.54609856e-03 1.23923367e-05]
 [2.13088302e+12 8.00198273e-02 7.78353130e-06]]
```





1.2 Ques 2

[2]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Backward Propagation function
def backward(num1, num2, x, iter):
    val = (2 * iter / x) * num1 - num2
    return val

# Function for calculation and plotting of absolute & relative errors
def plot_err(J, J_true):
    abs_err = np.abs(J - J_true)
    rel_err = np.abs((J - J_true) / J_true)

    print("\n")
    print("Absolute Error:\n")
    print(abs_err.astype(float))
    print("\n")
    print("Relative Error:\n")
    print(rel_err.astype(float))
    print("\n")

    # Plotting the absolute error graphs
    fig, axes = plt.subplots(1, 3, figsize=(18, 5))
    for i in range(3):
        axes[i].plot(abs_err[:, i], label=f'for x={x[i]}')
```

```python
        axes[i].set_xlabel('n values')
        axes[i].set_ylabel('Error values')
        axes[i].set_title(f'Absolute Error Propagation for x={x[i]}')
        axes[i].legend()
    plt.tight_layout()
    plt.show()

    # Plotting the relative error graphs
    fig, axes = plt.subplots(1, 3, figsize=(18, 5))
    for i in range(3):
        axes[i].plot(rel_err[:, i], "r-", label=f'for x={x[i]}')
        axes[i].set_xlabel('n values')
        axes[i].set_ylabel('Error values')
        axes[i].set_title(f'Relative Error Propagation for x={x[i]}')
        axes[i].legend()
    plt.tight_layout()
    plt.show()

# True Values of Bessels Function (Ji(x)) for x = 1,5,50 and i = 0,1,...,10
J_true = np.array([
    [7.6519768656e-01, -1.7759677131e-01, 5.5812327669e-02],
    [4.4005058574e-01, -3.2757913759e-01, -9.7511828125e-02],
    [1.1490348493e-01, 4.6565116278e-02, -5.9712800794e-02],
    [1.9563353983e-02, 3.6483123061e-01, 9.2734804062e-02],
    [2.4766389641e-03, 3.9123236046e-01, 7.0840977282e-02],
    [2.4975773021e-04, 2.6114054612e-01, -8.1400247697e-02],
    [2.0938338002e-05, 1.3104873178e-01, -8.7121026821e-02],
    [1.5023258174e-06, 5.3376410156e-02, 6.0491201260e-02],
    [9.4223441726e-08, 1.8405216655e-02, 1.0405856317e-01],
    [5.2492501799e-09, 5.5202831385e-03, -2.7192461044e-02],
    [2.6306151237e-10, 1.4678026473e-03, -1.1384784915e-01]
])

# Predicted Values of Bessels Function
J = np.empty((11, 3), dtype=object)
J[9, 0] = 5.2492e-09
J[10, 0] = 2.6306e-10
J[9, 1] = 5.5202e-03
J[10, 1] = 1.4678e-03
J[9, 2] = -2.7192e-02
J[10, 2] = -1.1384e-01

# x vector
x = [1, 5, 50]

# Number of iterations required
max_iter = 10

# Main loop for predicting J value
for j in range(3):
    for i in range(max_iter-2,-1,-1):
        J[i, j] = backward(J[i+1, j], J[i+2, j], x[j], i+1)
```

```
# Printing actual and predicted values of Bessels function
print("\n")
print("True Values:\n")
print(J_true.astype(float))
print("\n")
print("Predicted Values:\n")
print(J.astype(float))

# Plotting the errors
plot_err(J.astype(float), J_true)
```

```
True Values:

[[ 7.65197687e-01 -1.77596771e-01  5.58123277e-02]
 [ 4.40050586e-01 -3.27579138e-01 -9.75118281e-02]
 [ 1.14903485e-01  4.65651163e-02 -5.97128008e-02]
 [ 1.95633540e-02  3.64831231e-01  9.27348041e-02]
 [ 2.47663896e-03  3.91232360e-01  7.08409773e-02]
 [ 2.49757730e-04  2.61140546e-01 -8.14002477e-02]
 [ 2.09383380e-05  1.31048732e-01 -8.71210268e-02]
 [ 1.50232582e-06  5.33764102e-02  6.04912013e-02]
 [ 9.42234417e-08  1.84052167e-02  1.04058563e-01]
 [ 5.24925018e-09  5.52028314e-03 -2.71924610e-02]
 [ 2.63061512e-10  1.46780265e-03 -1.13847849e-01]]


Predicted Values:

[[ 7.65190364e-01 -1.77593886e-01  5.58072756e-02]
 [ 4.40046374e-01 -3.27573815e-01 -9.75058852e-02]
 [ 1.14902385e-01  4.65643594e-02 -5.97075110e-02]
 [ 1.95631668e-02  3.64825303e-01  9.27292843e-02]
 [ 2.47661526e-03  3.91226004e-01  7.08350251e-02]
 [ 2.49755340e-04  2.61136304e-01 -8.13956803e-02]
 [ 2.09381376e-05  1.31046603e-01 -8.71141612e-02]
 [ 1.50231144e-06  5.33755440e-02  6.04882816e-02]
 [ 9.42225400e-08  1.84049200e-02  1.04050880e-01]
 [ 5.24920000e-09  5.52020000e-03 -2.71920000e-02]
 [ 2.63060000e-10  1.46780000e-03 -1.13840000e-01]]


Absolute Error:

[[7.32303506e-06 2.88571998e-06 5.05209415e-06]
 [4.21133160e-06 5.32214296e-06 5.94295451e-06]
 [1.09963814e-06 7.56866800e-07 5.28981233e-06]
 [1.87223960e-07 5.92763400e-06 5.51977005e-06]
 [2.37017200e-08 6.35630000e-06 5.95218530e-06]
 [2.39021000e-09 4.24244000e-06 4.56742052e-06]
 [2.00382000e-10 2.12858000e-06 6.86566900e-06]
 [1.43774000e-11 8.66156000e-07 2.91966000e-06]
 [9.01726000e-13 2.96655000e-07 7.68317000e-06]
```

```
 [5.01799000e-14 8.31385000e-08 4.61044000e-07]
 [1.51237000e-15 2.64730000e-09 7.84915000e-06]]
```

```
Relative Error:

[[9.57012180e-06 1.62487187e-05 9.05193236e-05]
 [9.57010793e-06 1.62468923e-05 6.09459860e-05]
 [9.57010260e-06 1.62539442e-05 8.85875769e-05]
 [9.57013609e-06 1.62476057e-05 5.95220975e-05]
 [9.57011512e-06 1.62468667e-05 8.40217841e-05]
 [9.57011420e-06 1.62458112e-05 5.61106465e-05]
 [9.57010055e-06 1.62426600e-05 7.88061074e-05]
 [9.57009447e-06 1.62273184e-05 4.82658625e-05]
 [9.57008132e-06 1.61179847e-05 7.38350575e-05]
 [9.55944150e-06 1.50605500e-05 1.69548464e-05]
 [5.74911163e-06 1.80358034e-06 6.89442098e-05]]
```