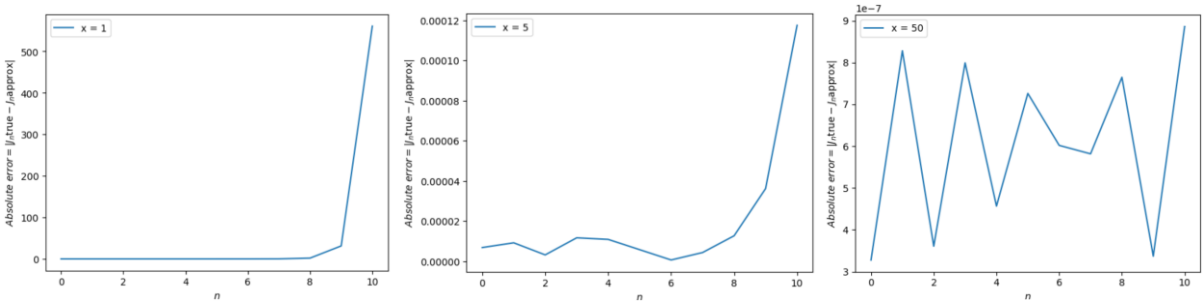
Problem 1. Forward recurrence approach: absolute error and relative error for different values of x are given in the Table 1. Propagation of absolute error with n can be seen in fig 1(a), (b), (c). absolute error and relative error given in Table 1 are calculated using python code given in **section 1(a) of Appendix.**

Table 1

	$J_{10}(x=1)$	$J_{10}(x=5)$	$J_{10}(x=50)$
Absolute Error	560.5533099961624	0.0001174533140389433	8.86138297351291e-07
Relative Error	2130883020271.455	0.08001982727034036	7.783531300479479e-06



1. (a) 1. (b) 1. (c)

Fig 1. (a), 1. (b) and 1. (c) shows how absolute error progress with n, when x =1, x = 5 and x= 50 respectively.

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

Error Calculation In forward recursion approach:

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

$$J_n(x) = \widehat{J_n}(x) + \varepsilon_n$$

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

Substituting equation (1) into equation (2)

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

Error recurrence relation in forward approach is as follows:

$$\varepsilon_{n+1}(x) = \frac{2n}{x}\left(\varepsilon_n(x)\right) - \varepsilon_{n-1}(x)(x) \quad (3)$$

Error Calculation for x = 1

Initial errors:

- ε_0, ε_1 are initial errors.
- Value of J_0 (x = 1) is 7.6519768656e-01, only five significant digits are taken, the maximum value of $\varepsilon_0(1)$, $\varepsilon_1(1)$ will be $0.00009999 \times 10^{-1} \approx 1\times10^{-5}$.

Let's assume $\varepsilon_0(1) = \varepsilon1(1) = \varepsilon_0 = 1\times10^{-4} \times10^{-1} = 10^{-5}$.

$$\varepsilon_{n+1}(1) = \frac{2n}{1}\left(\varepsilon_n(1)\right) - \varepsilon_{n-1}(1)$$

For n = 1:

$$\varepsilon_2(1) = 2 \times 1 \cdot \varepsilon_1 - \varepsilon_0 = 2\varepsilon_0 - \varepsilon_0 = \varepsilon_0$$

$$\varepsilon_3 = 2 \times 2\varepsilon_2 - \varepsilon_1 = 2 \times 2\varepsilon_0 - \varepsilon_1 = 4\varepsilon_0 - \varepsilon_0$$

$$\varepsilon_4 = 2 \times 3\varepsilon_3 - \varepsilon_2 = 2 \times 3 \times 2 \times 2\varepsilon_0 - 2 \times 3\varepsilon_0 - \varepsilon_0$$

$$\varepsilon_{10} = 62115169 \cdot \varepsilon_0 \leq 62115169 \cdot 10^{-5}$$

(Calculated using python code given in section 1(b) of appendix.)

$$\varepsilon_{10} \leq 621.15169$$

Error Calculation for x = 5

Initial errors:

- ε_0, ε_1 are initial errors.
- Value of J_0 (x = 5) is -1.7759677131e-01, only five significant digits are taken, the maximum value of $\varepsilon_0(1)$, $\varepsilon_1(1)$ will be $0.00009999 \times 10^{-1} \approx 1 \times 10^{-5}$.

Let's assume $\varepsilon_0(1) = \varepsilon 1(1) = \varepsilon_0 = 1 \times 10^{-4} \times 10^{-1} = 10^{-5}$.

For n = 1:

$$\varepsilon_2 = \frac{2}{5}\varepsilon_1 - \varepsilon_0$$

$$\varepsilon_3 = \frac{2 \times 2}{5}\varepsilon_2 - \varepsilon_0$$

$$\varepsilon_{10} = 29.5957 \cdot \varepsilon_0 \leq 29.5957 \times 10^{-5}$$

(can be calculated using python code given in section 1(b) of appendix with some modification)

$$\varepsilon_{10} \leq 0.000295957$$

Error Calculation for x = 50

Initial errors:

- ε_0, ε_1 are initial errors.
- Value of J_0 (x = 50) is 5.5812327669e-02, only five significant digits are taken, the maximum value of $\varepsilon_0(1)$, $\varepsilon_1(1)$ will be $0.00009999 \times 10^{-2} \approx 1 \times 10^{-6}$.

Let's assume $\varepsilon_0(1) = \varepsilon 1(1) = \varepsilon_0 = 1 \times 10^{-4} \times 10^{-1} = 10^{-6}$.

For n = 1:

$$\varepsilon_2 = \frac{2}{50}\varepsilon_1 - \varepsilon_0$$

$$\varepsilon_3 = \frac{2 \times 2}{50}\varepsilon_2 - \varepsilon_0$$

$$\varepsilon_{10} = 0.30008 \cdot \varepsilon_0 \leq 0.3008 \times 10^{-6}$$

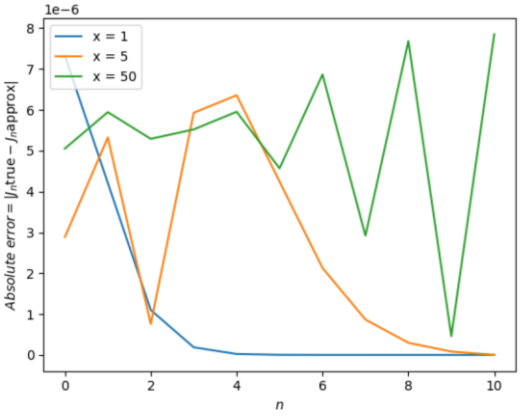
(can be calculated using python code given in section 1(b) of appendix with some modification)

$$\varepsilon_{10} \leq 3.008 \times 10^{-7}$$

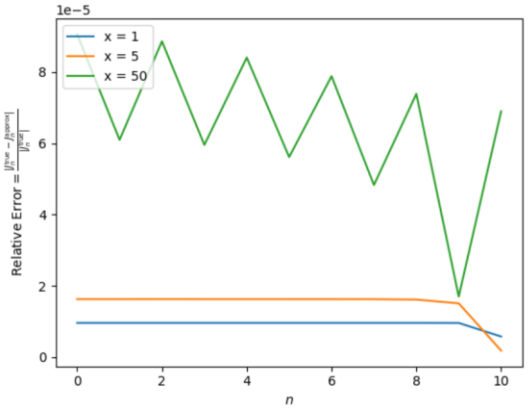
Problem 2. Backward recurrence approach: absolute error and relative error for different values of x are given in the table 1. Propagation of error can be with n can be seen in fig 2(a), (b), (c). absolute error and relative error given in table 2 are calculated using python code given in **section 2(a) of Appendix.**

Table 2

	$J_0(x=1)$	$J_0(x=5)$	$J_0(x=50)$
Absolute Error	7.323035060124994e-06	2.885719983913848e-06	5.052094149776698e-06
Relative Error	9.570121798258712e-06	1.6248718727418446e-05	9.051932361141778e-05



$2.(a)$



$2.(b)$

Error recurrence relation in backward approach is as follows:

$$\varepsilon_{n-1}(x) = \frac{2n}{x}\left(\varepsilon_n(x)\right) - \varepsilon_{n+1}(x)(x) \qquad (4)$$

Error Calculation for x = 1

Initial errors:

- ε_{10}, ε_9 are initial errors.
- Value of J_{10} (x = 1) is 2.6306151237e-10, only five significant digits are taken, the maximum value of $\varepsilon_{10}(1)$, $\varepsilon_9(1)$ will be $0.00009999 \times 10^{-10} \approx 0.0001 \times 10^{-10}$.

Let's assume $\varepsilon_{10}(1) = \varepsilon_9(1) = \varepsilon_{10} = 1\times10^{-4} \times 10^{-1} = 10^{-14}$.

$$\varepsilon_{n-1}(x) = \frac{2n}{x}\left(\varepsilon_n(x)\right) - \varepsilon_{n+1}(x)$$

n = 9: $\varepsilon_8(1) = 2 \times 9\varepsilon_9(1) - \varepsilon_{10} = 17\varepsilon_{10}$

n = 8: $\varepsilon_7(1) = 2.8\varepsilon_8(1) - \varepsilon_9(1) = 2.8 \cdot 17 \cdot \varepsilon_{10} - \varepsilon_{10} = 271\varepsilon_{10}$

$n = 1$: $\varepsilon_0(1) = 138031569.0 \cdot \varepsilon_0 \leq 138031569 \times 10^{-14} \leq 1.38031 \times 10^{-6}$

(Calculated using python code given in section 2(b) of appendix.)

Error Calculation for x = 5

Initial errors:

- ε_{10}, ε_9 are initial errors.
- Value of J_{10} (x = 5) is 1.4678026473e-03, only five significant digits are taken, the maximum value of $\varepsilon_{10}(1)$, $\varepsilon_9(1)$ will be $0.00009999 \times 10^{-3} \approx 0.0001 \times 10^{-3}$.

Let's assume $\varepsilon_{10}(5) = \varepsilon_9(5) = \varepsilon_{10} = 1\times10^{-4} \times 10^{-3} = 10^{-7}$.

n = 9: $$\varepsilon_8(5) = \frac{2\times9}{5}\varepsilon_9(5) - \varepsilon_{10}(5) = \frac{2\times9}{5}\varepsilon_{10}(5) - \varepsilon_{10}(5)$$

n = 1: $$\varepsilon_0(5) = \frac{2\times1}{5}\varepsilon_1(5) - \varepsilon_2(5) = |-24.2119 \cdot \varepsilon_{10}(5)|$$

(can be calculated using python code given in section 2(b) of appendix with some modification)

$$\varepsilon_0(5) \leq 24.2119 \times 10^{-14}$$

Error Calculation for x = 50

Initial errors:

- ε_{10}, ε_9 are initial errors.
- Value of J_{10} (x = 50) is -1.1384784915e-01, only five significant digits are taken, the maximum value of $\varepsilon_{10}(1)$, $\varepsilon_9(1)$ will be $0.00009999 \times 10^{-1} \approx 0.0001 \times 10^{-3}$.

Let's assume $\varepsilon_{10}(50) = \varepsilon_9(50) = \varepsilon_{10} = 1\times10^{-4} \times 10^{-1} = 10^{-5}$.

$n = 9$: $$\epsilon_8(50) = \frac{2\times9}{50}\epsilon_9(50) - \epsilon_{10}(50) = \frac{2\times9}{50}\epsilon_{10}(50) - \frac{1}{50}\epsilon_{10}(50)$$

n = 1: $$\epsilon_0(50) = \frac{2\times1}{50}\epsilon_1(50) - \epsilon_2(50) = 0.158086\,\epsilon_{10}(50)$$

(can be calculated using python code given in section 2(b) of appendix with some modification)

$$\epsilon_0(50) \leq 1.58086 \times 10^{-6}$$

The error calculated using the backward direction approach in a recurrence relation is less than that of the forward direction approach. The absolute error also increases exponentially, similar to the forward recurrence approach. However, both the absolute and relative errors are lower in the case of the backward recurrence approach. This is due to the fact that initial roundoff error is smaller in the backward recurrence approach compared to the forward recurrence approach.

Problem 3. Error Propagation can't be formally analyzed using the difference equation analysis (discussed in class). In order to prove it let's take $J_n = C\lambda^n$ and substitute in following recurrence relation (equation (1))

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

$$C\lambda^{n+1} + C\lambda^{n-1} = \frac{2n}{x}C\lambda^n$$

$$C\lambda^n\left(\lambda^2 - \frac{2n}{x}\lambda + 1\right) = 0$$

$$y_n\left(\lambda^2 - \frac{2n}{x}\lambda + 1\right) = 0$$

$$\lambda_{1,2} = \frac{\left(\frac{2n}{x}\right) \pm \sqrt{\left(\frac{2n}{x}\right)^2 - 4}}{2}$$

$$\lambda_{1,2} = \left(\frac{n}{x}\right) \pm \sqrt{\left(\frac{n}{x}\right)^2 - 1}$$

General solution of the equation (1)

$$J_n = C_1\left(\frac{n}{x}\left[1 + \sqrt{1 - \left(\frac{x}{n}\right)^2}\right]\right)^n + C_2\left(\frac{n}{x}\left[1 - \sqrt{1 - \left(\frac{x}{n}\right)^2}\right]\right)^n$$

Let's check whether the general solution of equation (1) satisfy equation (1) or not for $y_n(x = 1)$:

$$J_n = C_1\left(n + \sqrt{n^2 - 1}\right)^n + C_2\left(n - \sqrt{n^2 - 1}\right)^n$$

$$C_1\left(n - 1 + \sqrt{(n-1)^2 - 1}\right)^{n-1} + C_2\left(n - 1 - \sqrt{(n-1)^2 - 1}\right)^{n-1}$$

$$+C_1\left(n + 1 + \sqrt{(n+1)^2 - 1}\right)^{n+1} + C_2\left(n + 1 - \sqrt{(n+1)^2 - 1}\right)^{n+1}$$

$$= 2n \cdot \left(C_1\left(n + \sqrt{n^2 - 1}\right)^n + C_2\left(n - \sqrt{n^2 - 1}\right)^n\right)$$

Take n = 1: $$C_1(0+i)^0 + C_2(0-i)^0 + C_1\left(2 + \sqrt{3}\right)^2 + C_2\left(2 - \sqrt{3}\right)^2$$

$$= 2\left(C_1(1) + C_2(1)\right)$$

The coefficients of C_1 and C_2 on the left side are not equal to the coefficients of C_1 and C_2 on the right side. Therefore, error propagation cannot be formally analysed using the difference equation analysis, as the general solution of equation (1) does not satisfy equation (1).

So far, we have been unable to understand the error behaviour through difference equation analysis. However, the error can be understood in a loose way using difference equation analysis. If we directly substitute the value of n/x, the difference equation analysis becomes useful.

Ex.1. Solve the recurrence relation equation for n=1 and x=1 to calculate J_2(x=1). We will also try to understand the behaviour of the error when n=2 and x=1.

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

$$J_0 + J_2 = 2J_1$$

$$\lambda_{1,2} = \frac{n}{x} \pm \sqrt{\left(\frac{n}{x}\right)^2 - 1}$$

$$\lambda_{1,2} = 1,1$$

$$J_n = (C_1 + nC_2)$$

Initial values of J_0 (x=1) =0.76519768656, $J_1(x = 1) = 0.44005058574$,

$$J_0 = C_1 = 0.76519768656$$

$$J_1 = C_1 + C_2 = 0.44005058574$$

$$C_2 = -0.32514710082$$

$$J_n = 0.765197686856 - 0.325147101082n$$

If we take only first five digits,

$$\widehat{y_n} = 0.76519 - 0.32514n$$

$$\epsilon_n(x = 1) = |y_n - \widehat{y_n}| = 0.000007686856 - 0.00000571082n$$

Error behavior while calculating $J_2(x=1)$ grows linear. We can consider it stable.

Ex.2. Solve the recurrence relation equation for n=9 and x=1 to calculate $J_{10}(x=1)$. We will also try to understand the behaviour of the error when n=10 and x=1.

$$\lambda_{1,2} = 9 \pm \sqrt{9^2 - 1}$$

$$\lambda_{1,2} = 17.944, 0.0557$$

$$J_n = C_1(17.944)^n + C_2(0.0557)^n \quad (2)$$

Let's assume C_1 and C_2 are calculated putting the exact values of $J_8(1)$ = 9.4223441726e-08 and J_9 = 5.2492501799e-09 (1) in equation (2), while C_1' and C_2' are calculated putting the values of J_8 (1) = 9.4223e-08 and J_9 (1) = 5.2492e-09 taken up to the first 5 digits in equation (2)

$$\widehat{J_n} = C_1'(17.944)^n + C_2'(0.0557)^n$$

Take n = 1:

$$J_{10} = C_1(17.944)^{10} + C_2(0.0557)^{10}$$

$$\widehat{J_{10}} = C_1'(17.944)^{10} + C_2'(0.0557)^{10}$$

$$\text{error} = |y_{10} - \widehat{y_{10}}| = C_1^{"}(17.944)^{10} + C_2^{"}(0.0557)^{10}$$

as we are only taking first five digits, C_1'', $C_2'' \leq$ 0.0001e-8

$$\text{error} = |y_{10} - \widehat{y_{10}}| \leq 0.0001 \times 10^{-8} \times (17.944)^{10} + 0.0001 \times 10^{-8} \times (0.0557)^{10}$$

Error behavior while calculating $J_{10}(x=1)$ grows exponentially.

Conclusion:

Based on the above two examples, we can conclude that the error increases exponentially, with the base being $\left(\frac{n}{x} \pm \sqrt{\left(\frac{n}{x}\right)^2 - 1}\right)$ and the exponent being n. This can be practically seen in error calculated for forward recurrence approach, in Figure 1 (a), where the error increases exponentially when n/x is always greater than 1 (x=1, n=1,2, 3, . . . ,9) In Figure 1 (b), where n/x ranges from 0.2 to 2, the error does not increase exponentially; instead, it oscillates but eventually begins to increase exponentially after a certain point. In Figure 1 (c), where n/x ranges from 0.02 to 0.2, the error does not increase exponentially but instead fluctuates (due to machine epsilon). Same trend can also be seen in case of backward recurrence approach fig (2. (a)).

Appendix:

Code to import all the necessary libraries and some common data that needs to be run before executing the solution code provided in any section of this appendix."

```
import numpy as np

import sympy as sp

import matplotlib.pyplot as plt

from sympy import symbols, latex

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]
])
```

Section 1:

Section 1(a)

```
## forward Recursion
J_approx = [
    [7.6519e-01, -1.7759e-01,  5.5812e-02],
    [4.4005e-01, -3.2757e-01, -9.7511e-02],
    [0,  0, 0],
    [0,  0, 0],
    [0,  0, 0],
    [0,  0, 0],
    [0,  0, 0],
    [0,  0, 0],
    [0,  0, 0],
    [0,  0, 0],
    [0,  0, 0],
]
err_abs = np.zeros((11, 3))
err_rel = np.zeros((11, 3))

def beselfunction_forward(x_1, x_2, x_3, n):
    if n == 11:
        return (x_1[1], x_2[1], x_3[1])
    else:
        y_1 = [0, 0]
        y_2 = [0, 0]
        y_3 = [0, 0]
        y_1[0] = x_1[1]
        y_2[0] = x_2[1]
```

```
    y_3[0] = x_3[1]
    y_1[1] = 2 * (n-1) * x_1[1] - x_1[0]
    J_approx[n][0] = y_1[1]

    y_2[1] = 2 * (n-1) * x_2[1] / 5 - x_2[0]
    J_approx[n][1] = y_2[1]

    y_3[1] = 2 * (n-1) * x_3[1] / 50 - x_3[0]
    J_approx[n][2] = y_3[1]

    n = n + 1
    return beselfunction_forward(y_1, y_2, y_3, n)

a, b, c = beselfunction_forward([J_approx[0][0], J_approx[1][0]], [J_approx[0][1], J_approx[1][1]], [J_approx[0][2], J_approx[1][2]], 2)
err_abs = abs(J_approx - J_true)
err_rel = abs((J_approx - J_true)/J_true)
print("Absolute error {} {} {}".format(err_abs[10][0], err_abs[10][1], err_abs[10][2]))
print("Relative error {} {} {}".format(err_rel[10][0], err_rel[10][1], err_rel[10][2]))
# As the difference in magnitude of error for x=1, x=5, x=50. therefore error need to be plot separately
n = np.arange(11)

fig, ax = plt.subplots(1)
ax.plot(n, err_abs[:, 0], label = 'x = 1')
ax.set_ylabel(r'$Absolute\;error = |J_n{\text{true}}-J_n{\text{approx}}|$')
ax.set_xlabel(r'$n$')
ax.legend(loc='upper left')

fig, ax = plt.subplots(1)
ax.plot(n, err_abs[:, 1], label = 'x = 5')
ax.set_ylabel(r'$Absolute\;error = |J_n{\text{true}}-J_n{\text{approx}}|$')
ax.set_xlabel(r'$n$')
ax.legend(loc='upper left')

fig, ax = plt.subplots(1)
ax.plot(n, err_abs[:, 2], label = 'x = 50')
ax.set_ylabel(r'$Absolute\;error= |J_n{\text{true}}-J_n{\text{approx}}|$')
ax.set_xlabel(r'$n$')
```

```
ax.legend(loc='upper left')
```

Section 1(b)

```
# Define symbols
x = sp.symbols('x')
eps0, eps1 = sp.symbols('epsilon_0 epsilon_1')

# Initial conditions
eps = [eps0, eps0]

# Recurrence relation
for n in range(1, 10):
eps_next = (2 * n /1) * eps[-1] - eps[-2]  #2n/x in this case x = 1
eps.append(eps_next)

# Print epsilon_10
epsilon_10 = eps[-1]
sp.pprint(epsilon_10)
```

Section 2:

Section 2(a)

```
## Backward Recursion
J_approx = [
   [0, 0, 0],
   [0, 0, 0],
   [0, 0, 0],
   [0, 0, 0],
   [0, 0, 0],
   [0, 0, 0],
   [0, 0, 0],
   [0, 0, 0],
   [0, 0, 0],
   [5.2492e-09,  5.5202e-03, -2.7192e-02],
   [2.6306e-10,  1.4678e-03, -1.1384e-01],
]

err_abs = np.zeros((11, 3))
err_rel = np.zeros((11, 3))
```

```
def beselfunction_backward(x_1, x_2, x_3, n):
    if n == -1:
        return (x_1[0], x_2[0], x_3[0])
    else:
        y_1 = [0, 0]
        y_2 = [0, 0]
        y_3 = [0, 0]
        y_1[1] = x_1[0]
        y_2[1] = x_2[0]
        y_3[1] = x_3[0]
        y_1[0] = 2 * (n+1) * x_1[0] - x_1[1]
        J_approx[n][0] = y_1[0]

        y_2[0] = 2 * (n+1) * x_2[0] / 5 - x_2[1]
        J_approx[n][1] = y_2[0]

        y_3[0] = 2 * (n+1) * x_3[0] / 50 - x_3[1]
        J_approx[n][2] = y_3[0]

        n = n - 1
        return beselfunction_backward(y_1, y_2, y_3, n)

a, b, c = beselfunction_backward([J_predicted[9][0], J_predicted[10][0]], [J_predicted[9][1],
J_predicted[10][1]],[J_predicted[9][2], J_predicted[10][2]], 8)
print("2. {} {} {}".format(a, b, c))
err_abs = abs(J_approx - J_true)
err_rel = abs((J_approx - J_true)/J_true)
print("Absolute error: {} {} {}".format(err_abs[0][0], err_abs[0][1], err_abs[0][2]))
print("Relative error: {} {} {}".format(err_rel[0][0], err_rel[0][1], err_rel[0][2]))

n = np.arange(11)

fig, ax = plt.subplots(1)
ax.plot(n, err_abs[:, 0], label = 'x = 1')
ax.plot(n, err_abs[:, 1], label = 'x = 5')
ax.plot(n, err_abs[:, 2], label = 'x = 50')
ax.set_ylabel(r'$Absolute\;error = |J_n{\text{true}}-J_n{\text{approx}}|$')
```

```
ax.set_xlabel(r'$n$')
ax.legend(loc='upper left')

fig, ax = plt.subplots(1)
ax.plot(n, err_rel[:, 0], label = 'x = 1')
ax.plot(n, err_rel[:, 1], label = 'x = 5')
ax.plot(n, err_rel[:, 2], label = 'x = 50')
ax.set_ylabel(r'$\text{Relative Error} = \frac{|J_n^{\text{true}} - J_n^{\text{approx}}|}{|J_n^{\text{true}}|}$')
ax.set_xlabel(r'$n$')
ax.legend(loc='upper left')
```

Section (b):

```
# Define symbols
x = sp.symbols('x')
eps9, eps10 = sp.symbols('epsilon_9 epsilon_10')

# Initial conditions for backward recurrence
eps = [eps10, eps10]  # Start with epsilon_10 and epsilon_9

# Backward recurrence relation for error propagation
for n in range(9, 0, -1):  # Start from n=9 down to n=1
    eps_prev = (2 * n / 1) * eps[-1] - eps[-2] #2n/x in this case x = 1
    #print(n,eps[-1],eps[-2])
    eps.append(eps_prev)

# Reverse the list to get epsilon_0 at the end
eps.reverse()

# Print epsilon_0
epsilon_0 = eps[0]
sp.pprint(epsilon_0)
```