Bhikshapathi kaniki SR No: 06-18-01-10-51-24-1-24489

DS 288 (AUG) 3:0 Numerical Methods
Homework–1[1]

Question-1

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

1. Compute the recursion in the forward direction, i.e., compute $J_2(x)$ from $J_1(x)$ and $J_0(x)$ with starting values taken from table-1. Use only the first 5 digits given in the table for each quantity when supplying the starting values to your program. For $x = 1, 5,$ and 50, how accurate is $J_{10}(x)$? Compute both the absolute and relative errors of these values taking the tabulated values (table-1) as truth. [3 points]

Output

Results for $x = 1$

n	Computed $J_n(x)$	Original $J_n(x)$	Abs Error	Rel Error
0	7.6520000000e-01	7.6519768656e-01	2.3134400000e-06	3.0233233067e-06
1	4.4005000000e-01	4.4005058574e-01	5.8574000000e-07	1.3310742423e-06
2	1.1490000000e-01	1.1490348493e-01	3.4849300000e-06	3.0329193254e-05
3	1.9550000000e-02	1.9563353983e-02	1.3353983000e-05	6.8260192049e-04
4	2.4000000000e-03	2.4766389641e-03	7.6638964100e-05	3.0944746170e-02
5	-3.5000000000e-04	2.4975773021e-04	5.9975773021e-04	2.4013580269e+00
6	-5.9000000000e-03	2.0938338002e-05	5.9209383380e-03	2.8277976683e+02
7	-7.0450000000e-02	1.5023258174e-06	7.0451502326e-02	4.6894955482e+04
8	-9.8040000000e-01	9.4223441726e-08	9.8040009422e-01	1.0405055008e+07
9	-1.5615950000e+01	5.2492501799e-09	1.5615950005e+01	2.9748915502e+09
10	-2.8010670000e+02	2.6306151237e-10	2.8010670000e+02	1.0647954445e+12

Results for $x = 5$

n	Computed $J_n(x)$	Original $J_n(x)$	Abs Error	Rel Error
0	-1.7760000000e-01	-1.7759677131e-01	3.2286900000e-06	1.8179891313e-05
1	-3.2758000000e-01	-3.2757913759e-01	8.6240999997e-07	2.6326768130e-06
2	4.6568000000e-02	4.6565116278e-02	2.8837220000e-06	6.1928804876e-05
3	3.6483440000e-01	3.6483123061e-01	3.1693900000e-06	8.6872771136e-06
4	3.9123328000e-01	3.9123236046e-01	9.1954000003e-07	2.3503679475e-06
5	2.6113884800e-01	2.6114054612e-01	1.6981200000e-06	6.5027052491e-06
6	1.3104441600e-01	1.3104873178e-01	4.3157800001e-06	3.2932634612e-05
7	5.3367750400e-02	5.3376410156e-02	8.6597560001e-06	1.6223938580e-04
8	1.8385285120e-02	1.8405216655e-02	1.9931535000e-05	1.0829285726e-03
9	5.4651619840e-03	5.5202831385e-03	5.5121154501e-05	9.9852042220e-03
10	1.2892980224e-03	1.4678026473e-03	1.7850462490e-04	1.2161350522e-01

Results for $x = 50$

n	Computed $J_n(x)$	Original $J_n(x)$	Abs Error	Rel Error
0	5.5812000000e-02	5.5812327669e-02	3.2766900000e-07	5.8709072653e-06
1	-9.7512000000e-02	-9.7511828125e-02	1.7187500000e-07	1.7626066837e-06
2	-5.9712480000e-02	-5.9712800794e-02	3.2079400000e-07	5.3722819184e-06
3	9.2735001600e-02	9.2734804062e-02	1.9753800000e-07	2.1301387542e-06
4	7.0840680192e-02	7.0840977282e-02	2.9709000000e-07	4.1937591970e-06
5	-8.1400492769e-02	-8.1400247697e-02	2.4507228001e-07	3.0107068092e-06
6	-8.7120778746e-02	-8.7121026821e-02	2.4807514400e-07	2.8474772744e-06
7	6.0491505870e-02	6.0491201260e-02	3.0461027456e-07	5.0356129191e-06
8	1.0405840039e-01	1.0405856317e-01	1.6278046713e-07	1.5643159215e-06
9	-2.7192817746e-02	-2.7192461044e-02	3.5670162404e-07	1.3117666086e-05
10	-1.1384781478e-01	-1.1384784915e-01	3.4372042476e-08	3.0191209349e-07

Question -2

2. Compute the recursion backward, i.e. start with J10(x) from J9(x) compute J8(x). Again use only first 5 digits and for x = 1,5, and 50, how accurate is J0(x) in this backward approach?. Compute both the absolute and relative errors of these values taking the tabulated values (table-1) as truth. Is the last value computed by the recurrence relation is having less or more error compared to the forward approach?. [3 points]

Results for $x = 1$

n	Computed $J_n(x)$	Original $J_n(x)$	Absolute Error	Relative Error
0	7.6520498164e-01	7.6519768656e-01	7.2950819401e-06	9.5335912121e-06
1	4.4005478101e-01	4.4005058574e-01	4.1952685000e-06	9.5336050808e-06
2	1.1490458038e-01	1.1490348493e-01	1.0954450600e-06	9.5336104094e-06
3	1.9563540492e-02	1.9563353983e-02	1.8650874000e-07	9.5335769195e-06
4	2.4766625754e-03	2.4766389641e-03	2.3611280000e-08	9.5335978889e-06
5	2.4976011130e-04	2.4975773021e-04	2.3810900000e-09	9.5335988118e-06
6	2.0938537620e-05	2.0938338002e-05	1.9961800000e-10	9.5336124568e-06
7	1.5023401400e-06	1.5023258174e-06	1.4322600000e-11	9.5336176973e-06
8	9.4224340000e-08	9.4223441726e-08	8.9827400001e-13	9.5334450064e-06
9	5.2493000000e-09	5.2492501799e-09	4.9820100000e-14	9.4908983746e-06
10	2.6306000000e-10	2.6306151237e-10	1.5123700000e-15	5.7491116294e-06

Results for $x = 5$

n	Computed $J_n(x)$	Original $J_n(x)$	Absolute Error	Relative Error
0	-1.7759739106e-01	-1.7759677131e-01	6.1974894394e-07	3.4896408272e-06
1	-3.2758028050e-01	-3.2757913759e-01	1.1429053599e-06	3.4889442848e-06
2	4.6565278861e-02	4.6565116278e-02	1.6258279997e-07	3.4915149573e-06
3	3.6483250358e-01	3.6483123061e-01	1.2729739999e-06	3.4892133487e-06
4	3.9123372544e-01	3.9123236046e-01	1.3649799999e-06	3.4889240714e-06
5	2.6114145712e-01	2.6114054612e-01	9.1099999994e-07	3.4885429072e-06
6	1.3104918880e-01	1.3104873178e-01	4.5701999996e-07	3.4874049809e-06
7	5.3376596000e-02	5.3376410156e-02	1.8584399999e-07	3.4817628134e-06
8	1.8405280000e-02	1.8405216655e-02	6.3344999995e-08	3.4416872772e-06
9	5.5203000000e-03	5.5202831385e-03	1.6861500000e-08	3.0544628920e-06
10	1.4678000000e-03	1.4678026473e-03	2.6473000001e-09	1.8035803416e-06

Results for $x = 50$

n	Computed $J_n(x)$	Original $J_n(x)$	Absolute Error	Relative Error
0	5.5814212372e-02	5.5812327669e-02	1.8847034238e-06	3.3768586665e-05
1	-9.7513132587e-02	-9.7511828125e-02	1.3044623059e-06	1.3377477697e-05
2	-5.9714737676e-02	-5.9712800794e-02	1.9368819161e-06	3.2436628166e-05
3	9.2735953573e-02	9.2734804062e-02	1.1495112326e-06	1.2395682983e-05
4	7.0843052105e-02	7.0840977282e-02	2.0748227040e-06	2.9288453994e-05
5	-8.1401065236e-02	-8.1400247697e-02	8.1753948000e-07	1.0043451993e-05
6	-8.7123265152e-02	-8.7121026821e-02	2.2383310000e-06	2.5692201776e-05
7	6.0491481600e-02	6.0491201260e-02	2.8034000000e-07	4.6343930053e-06
8	1.0406088000e-01	1.0405856317e-01	2.3168300000e-06	2.2264674136e-05
9	-2.7192000000e-02	-2.7192461044e-02	4.6104400000e-07	1.6954846391e-05
10	-1.1385000000e-01	-1.1384784915e-01	2.1508500000e-06	1.8892320022e-05

Analysis of Relative Errors for Forward and Backward Recursion

For $x = 1$:

- **Forward Recursion:** The relative error is extremely large, at $1.064795445 \times 10^{12}$.
- **Backward Recursion:** The relative error is very small, at $9.5335912121 \times 10^{-6}$.
- **Observation:** For $x = 1$, the forward recursion method is highly inaccurate, while the backward recursion method is significantly more precise.

For $x = 5$:

- **Forward Recursion:** The relative error is $1.2161350522 \times 10^{-1}$, which is still relatively high but much lower than the error for $x = 1$.
- **Backward Recursion:** The relative error is again very small, at $3.4896408272 \times 10^{-6}$.
- **Observation:** As x increases, the forward recursion method becomes more accurate, but it still lags significantly behind the backward recursion method in terms of precision.

For $x = 50$:

- **Forward Recursion:** The relative error drops significantly to $3.0191209349 \times 10^{-7}$.
- **Backward Recursion:** The relative error increases slightly to $3.3768586665 \times 10^{-5}$.
- **Observation:** As x becomes much larger, the forward recursion method becomes more accurate than in the previous cases, but the backward recursion method sees a small increase in error.

Question -3

3. Explain your finding. Can the error propagation be formally analyzed using the difference equation analysis we performed in class?. Can the error behavior be understood by this analysis?. Defend your answer to these questions. In case your answers are yes, do the analysis. If the answer is no, how will you explain the error propagation?. [4 points]

Introducting small errors into Computation

Assume computing bessel function with small error for $J_{n-1}(x)$, $J_n(x)$ and $J_{n+1}(x)$

$$J_{n-1}(x) \rightarrow J_{n-1}(x) + \delta J_{n-1}(x)$$

$$J_n(x) \rightarrow J_n(x) + \delta J_n(x)$$

$$J_{n+1}(x) \rightarrow J_{n+1}(x) + \delta J_{n+1}(x)$$

Here $\delta J_{n-1}(x)$, $\delta J_n(x)$, $\delta J_{n+1}(x)$ are the small errors for computing bessel function values

Errors introduced due to Recurrence Relation

Substitute the errors in bessel recurrsive relation

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

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

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

Forward Recursion :-

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

Backward Recursion :-

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

→ For forward $J_{n+1}(x)$ is influenced by $J_{n-1}(x)$ and $J_n(x)$

→ For Backward $J_{n-1}(x)$ is influenced by $J_{n+1}(x)$ and $J_n(x)$

ERROR Growth

→ In the recurrence relation error get add by $J_n(x)$ addition and also added because of multiplication of $\frac{2n}{x}$

→ If we look into the recurrence relation

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

→ The factor $\frac{2n}{x}$ can either amplify or attenuate the error depending upon the values of x and n.

→ if $\frac{2n}{x} < 0$ then the error growth decrease in every next iteration, leading to convergence to zero.

→ if $\frac{2n}{x} > 0$, then error growth increases in every next iteration, rapid error growth.

Solving the differential equation:-

$$\varepsilon_{n+1} = \frac{2n}{x}\varepsilon_n - \varepsilon_{n-1}$$

Let suppose $\frac{2n}{x} = p$ and $\varepsilon_{n-1} = a$

$$a^2 = pa - 1 \Rightarrow a^2 - pa + 1 = 0$$

$$a = \frac{+p \pm \sqrt{p^2 - 4}}{2} \Rightarrow a = \frac{p + \sqrt{p^2-4}}{2} \text{ or } a = \frac{p - \sqrt{p^2-4}}{2}$$

$$\varepsilon_n = c_1 \times \left(\frac{p + \sqrt{p^2-4}}{2}\right)^n + c_2\left(\frac{p - \sqrt{p^2-4}}{2}\right)$$

→ max and min of p for $x = 1 \Rightarrow 2$ and 20

substitute $p = 1$ and $p = 20$

$$\varepsilon_n(1) = c_1 + c_2 n \qquad \varepsilon_n(20) = (10 + 3\sqrt{11})^n c_1 + (10 - 3\sqrt{11})^n c_2$$

So

$$\boxed{c_1 + c_2(n) \leq \varepsilon_n(1) \leq c_3(10)^n}$$

$\Rightarrow x = 50$, $p_{min} = \frac{2 \times 1}{50} = 0.04$ $\qquad p_{max} = \frac{2 \times 10}{50} = 0.4$

As x goes up, the p values decrease and also the bounded error also decreases.

→ From both the Recurrence Relation it is clear that the factor $2n/x$ is impacting highly on the error propagation.

	$n=1$	$n=5$	$n=10$
$x=1$	2	10	20
$x=5$	0.4	2	4
$x=10$	0.2	1	2

$\frac{2n}{x}$

Forward Recurrence Relation Error Growth :-

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

→ when x is small then $\frac{2n}{x}$ will be large. This means any small error in J_{n-1} or J_n is significantly amplified for computing J_{n+1} term

→ So error get amplified for every next iteration [or increasing the value of n]

→ when x is large $\frac{2n}{x}$ will be small. This means error propagation will decrease.

→ The forward Recursion is stabilize for large value of n.

So, that is the reason we see error get triggered or increased for smaller x and tends to decrease as the x increase ($x=50$)

ERROR Growth in backward Recursion :-

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

→ when x is small, $\frac{2n}{x}$ is will be large. But here as n is decreasing the error amplification for J_{n+1} to J_{n-1} will decrease (as n decreases, $\frac{2n}{x}\downarrow$)

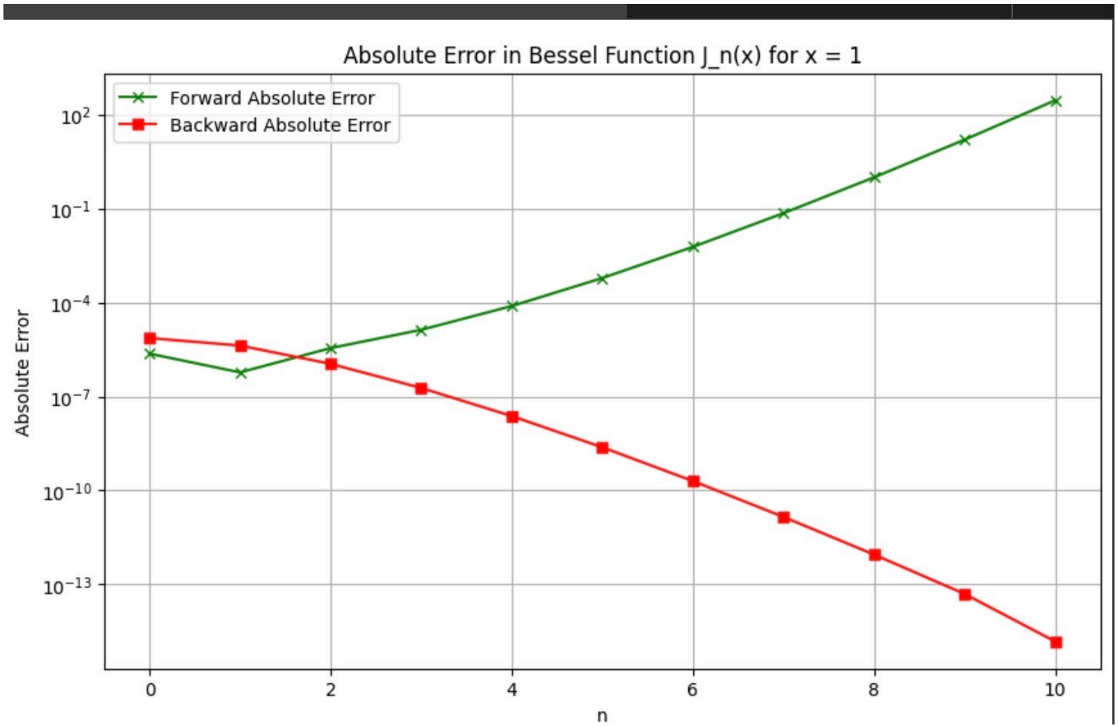
→ So Backward recursion will decreasing error. thus will give more accurate results

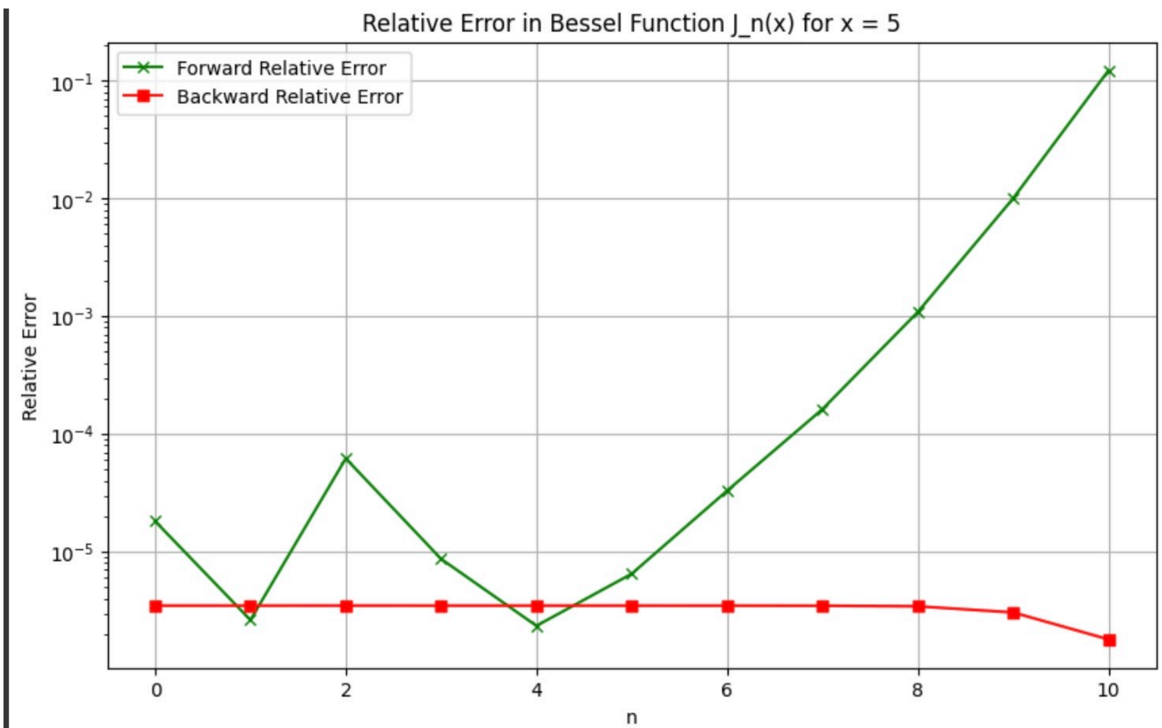
→ But as n increases, $\frac{2n}{x}$ will decrease, the factor for which amplification for higher n $\left(\frac{2\times 10}{50}\right)$ is more than forward recursion $\left(\frac{2\times 1}{50}\right)$

∴ More large value of n, forward Recursion is more stable than backward Recursion.

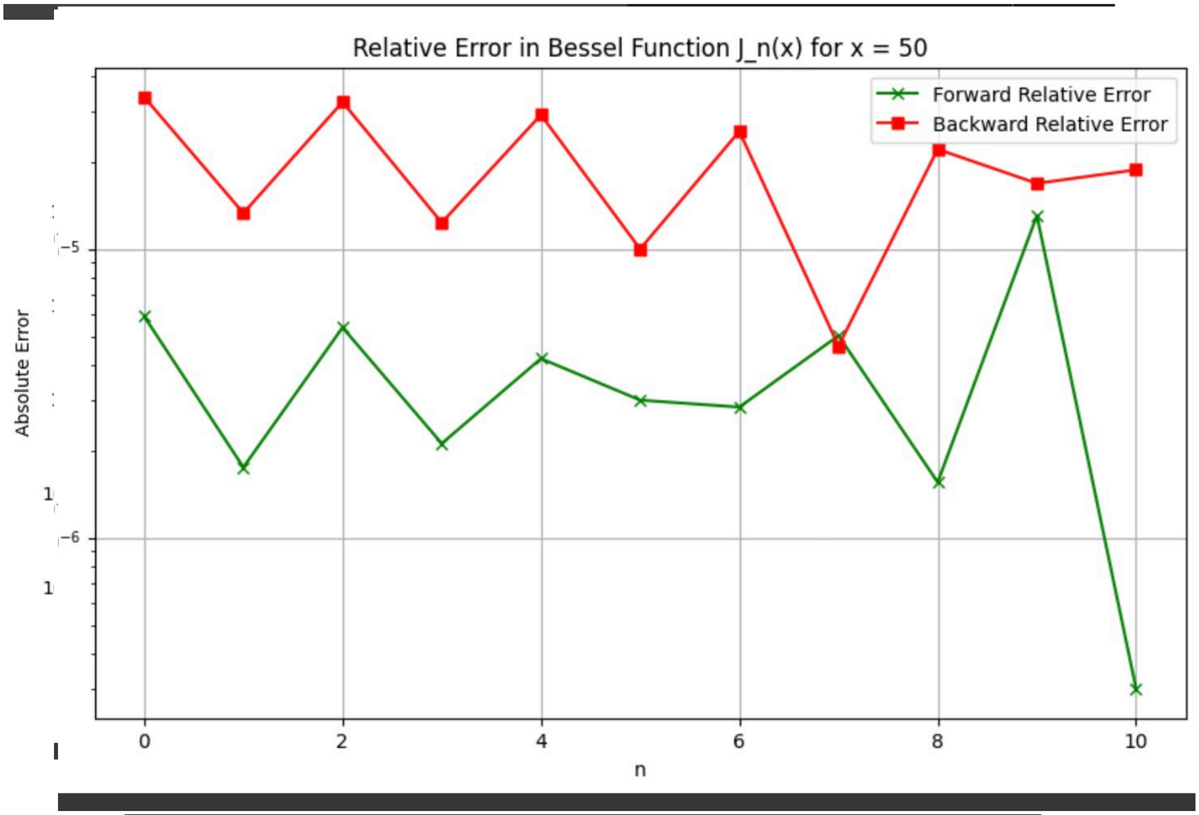
Conclusion

- Shrinking or growing of errors in bessel Recursive relation depends on the values of n, x.
- For small values of x Backward recursion generates less error
- For large values of x Forward recursion generates less error and accurate results.






```
Results for x = 50:
```



CODE

```
import numpy as np

def truncate_to_significant_digits(value, digits):
    if value == 0:
        return 0
    else:
        return round(value, digits - int(np.floor(np.log10(abs(value)))) - 1)

def bessel_forward_recursion(x, n_max, tab_0, tab_1, table_full):
    computed_values = np.zeros(n_max + 1)
    computed_values[0] = tab_0
    computed_values[1] = tab_1

    for n in range(1, n_max):
        computed_values[n + 1] = ((2 * n) * computed_values[n]) / x -
        computed_values[n - 1]
    absolute_errors = np.abs(computed_values - table_full)
    relative_errors = absolute_errors / np.abs(table_full)

    return computed_values, absolute_errors, relative_errors

def bessel_backward_recursion(x, n_max, tab_10, tab_9, table_full):
```

```
    computed_values = np.zeros(n_max + 1)
    computed_values[n_max] = tab_10
    computed_values[n_max - 1] = tab_9

    for n in range(n_max - 1, 0, -1):
        computed_values[n - 1] = ((2 * n) * computed_values[n]) / x -
        computed_values[n + 1]

    absolute_errors = np.abs(computed_values - table_full)
    relative_errors = absolute_errors / np.abs(table_full)

    return computed_values, absolute_errors, relative_errors

def compute_bessel(x_values, n_max, table_0, table_1, table_10, table_9, table_full):
    for x in x_values:
        print(f"Results for x = {x}:\n")

        # Forward Recursion
        tab_0 = truncate_to_significant_digits(table_0[x], 5)
        tab_1 = truncate_to_significant_digits(table_1[x], 5)
        calculated_forward, absolute_error_forward, relative_error_forward =
        bessel_forward_recursion(x, n_max, tab_0, tab_1, table_full[x])

        print("Forward Recursion:")
        print("n  | Computed J_n(x)  | Original J_n(x)  | Abs Error |RelError")
        print("---|------------------|--------------|-----------|------------")
        for n in range(n_max + 1):
            print(f"{n:<2} | {calculated_forward[n]:.10e} | {table_full[x]
            [n]:.10e}  | {absolute_error_forward[n]:.10e}  |
            {relative_error_forward[n]:.10e}")
        print("\n")

        # Backward Recursion
        tab_10 = truncate_to_significant_digits(table_10[x], 5)
        tab_9 = truncate_to_significant_digits(table_9[x], 5)
        calculated_backward, absolute_error_backward, relative_error_backward =
        bessel_backward_recursion(x, n_max, tab_10, tab_9, table_full[x])

        print("Backward Recursion:")
        print("n  | Computed J_n(x)  | Original J_n(x)  | Abs Error  | RelError")
        print("---|------------------|------------------|-----------|----------")
        for n in range(n_max + 1):
            print(f"{n:<2} | {calculated_backward[n]:.10e} | {table_full[x]
            [n]:.10e}  | {absolute_error_backward[n]:.10e}  |
            {relative_error_backward[n]:.10e}")
        print("\n")
```

```
n_max = 10
x_values = [1, 5, 50]

table_0 = {1: 7.6519768656e-01, 5: -1.7759677131e-01, 50: 5.5812327669e-02}
table_1 = {1: 4.4005058574e-01, 5: -3.2757913759e-01, 50: -9.7511828125e-02}
table_10 = {1: 2.6306151237e-10, 5: 1.4678026473e-03, 50: -1.1384784915e-01}
table_9 = {1: 5.2492501799e-09, 5: 5.5202831385e-03, 50: -2.7192461044e-02}

table_full = {
    1: [7.6519768656e-01, 4.4005058574e-01, 1.1490348493e-01, 1.9563353983e-02,
        2.4766389641e-03, 2.4975773021e-04, 2.0938338002e-05, 1.5023258174e-06,
        9.4223441726e-08, 5.2492501799e-09, 2.6306151237e-10],
    5: [-1.7759677131e-01, -3.2757913759e-01, 4.6565116278e-02, 3.6483123061e-01,
        3.9123236046e-01, 2.6114054612e-01, 1.3104873178e-01, 5.3376410156e-02,
        1.8405216655e-02, 5.5202831385e-03, 1.4678026473e-03],
    50: [5.5812327669e-02, -9.7511828125e-02, -5.9712800794e-02, 9.2734804062e-02,
         7.0840977282e-02, -8.1400247697e-02, -8.7121026821e-02, 6.0491201260e-02,
         1.0405856317e-01, -2.7192461044e-02, -1.1384784915e-01]
}

compute_bessel(x_values, n_max, table_0, table_1, table_10, table_9, table_full)
```