DS288 (AUG) 3:0 Numerical Methods

Naman Pesricha
namanp@iisc.ac.in
SR - 24115

Homework-1

A Microwave engineer is interested in developing a hazardous waste treatment facility based on microwave exposure to the hazardous material. The design is based on a cylindrical microwave cavity and requires computing of various modes of the electromagnetic fields that will exist in this structure. The modes of the systems are described by the Bessel functions $J_i(x)$ for i = 1,2, ... , n. As a numerical methods expert, your job is to help the engineer to compute these Bessel functions using the recurrence relation

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

n	$J_n(1)$	$J_n(5)$	$J_n(50)$
0	$7.6519768656 \times 10^{-1}$	$-1.7759677131 \times 10^{-1}$	$5.5812327669 \times 10^{-2}$
1	$4.4005058574 \times 10^{-1}$	$-3.2757913759 \times 10^{-1}$	$-9.7511828125 \times 10^{-2}$
2	$1.1490348493 \times 10^{-1}$	$4.6565116278 \times 10^{-2}$	$-5.9712800794 \times 10^{-2}$
3	$1.9563353983 \times 10^{-2}$	$3.6483123061 \times 10^{-1}$	$9.2734804062 \times 10^{-2}$
4	$2.4766389641 \times 10^{-3}$	$3.9123236046 \times 10^{-1}$	$7.0840977282 \times 10^{-2}$
5	$2.4975773021 \times 10^{-4}$	$2.6114054612 \times 10^{-1}$	$-8.1400247697 \times 10^{-2}$
6	$2.0938338002 \times 10^{-5}$	$1.3104873178 \times 10^{-1}$	$-8.7121026821 \times 10^{-2}$
7	$1.5023258174 \times 10^{-6}$	$5.3376410156 \times 10^{-2}$	$6.0491201260 \times 10^{-2}$
8	$9.4223441726 \times 10^{-8}$	$1.8405216655 \times 10^{-2}$	$1.0405856317 \times 10^{-1}$
9	$5.2492501799 \times 10^{-9}$	$5.5202831385 \times 10^{-3}$	$-2.7192461044 \times 10^{-2}$
10	$2.6306151237 \times 10^{-10}$	$1.4678026473 \times 10^{-3}$	$-1.1384784915 \times 10^{-1}$

Table 1: Bessel functions of integer order (n = 0 - 10) for x = 1, 5, and 50.

Q1 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]

Solution: We will initialize the first two rows ($J_0(x)$ and $J_1(x)$) from values from the table using only the first 5 digits and compute forward. Iterative scheme rearranged for forward computation:

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

- Absolute error can be calculated using:

$$|(data - \widehat{data})|$$

- Relative error can be calculated using:

$$|\frac{(data - \widehat{data})}{data}|$$

Q. ...Compute both the absolute and relative errors of these values taking the tabulated values (table-1) as truth....

	$J_{10}(1)$	$J_{10}(5)$	$J_{10}(50)$
ActualValue	$2.6306151237 \times 10^{-10}$	$1.4678026473 \times 10^{-3}$	$-1.1384784915 \times 10^{-1}$
ComputedValue	$5.6055331000 \times 10^{2}$	$1.5852559616 \times 10^{-3}$	$-1.1384696301 \times 10^{-1}$
AbsoluteError	$5.6055331000 \times 10^{2}$	$1.1745331430 \times 10^{-4}$	$8.8613829735 \times 10^{-7}$
RelativeError	$2.1308830203 \times 10^{12}$	$8.0019827270 \times 10^{-2}$	$-7.7835313005 \times 10^{-6}$

Table 2: Comparison of Actual, Computed Values, and Errors for $J_{10}(x)$ in forward computation.

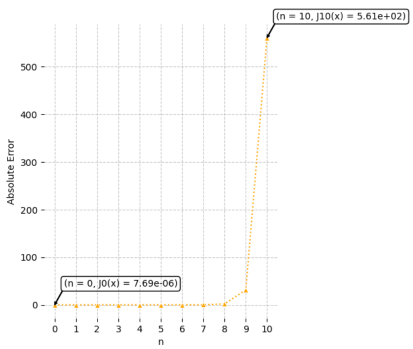

Figure 1: Forward Error x = 1

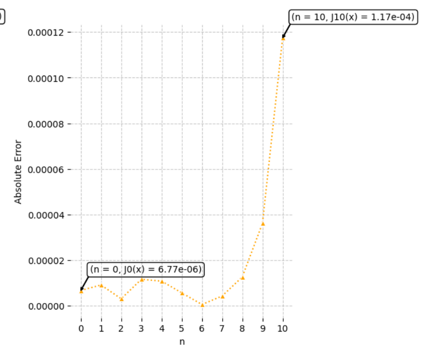

Figure 2: Forward Error x = 5

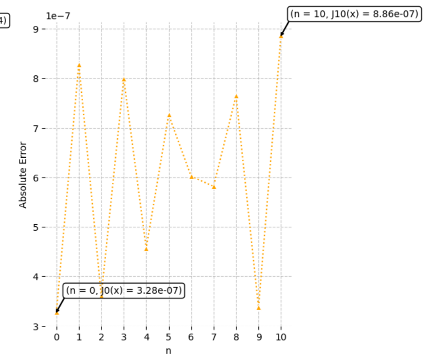

Figure 3: Forward Error x = 50

Observations:
1. From Figure 1, The error growth for x = 1 for forward pass is exponential. The error in forward pass grew from order of 10^{-6} to order of 10^{2}.

2. From Figure 2, The error growth for x = 5 for forward pass seems to be exponential at around from n = {6 to 10}. The error in forward pass grew from order of 10^{-6} to order of 10^{-4}.

3. From Figure 3, The error growth for x = 50 is juggling randomly (not exponential) for forward pass. The error in forward pass is maintained in order of 10^{-7}.

Explanations:
Observation 1: For $x = 1$, from Table 5, we can see the $\beta \geq 2$ for all iterations, hence the error is increasing exponentially across the whole forward pass in Figure 1.

Observation 2: For $x = 5$, from Table 5, we can see that $\beta \geq 2$ for all values of $n \ \epsilon \ \{6, 7, 8, 9, 10\}$. Hence the error increases exponentially from $n = 6 \ to \ 10$ and the rest is non-exponential in Figure 2.

Observation 3: For $x = 50$, from Table 5, we can see that $\beta < 2$ for all iterations. Hence the error is juggling randomly (not exponential) in Figure 3.

Q2 Compute the recursion backward, i.e. start with $J_{10}(x)$ from $J_9(x)$ compute $J_8(x)$. Again use only the first 5 digits and for x = 1,5, and 50, how accurate is $J_0(x)$ 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]

Solution: We will initialize the last two rows ($J_{10}(x)$ and $J_9(x)$) from values from the table using only the first 5 digits and compute backward. Iterative scheme rearranged for backward computation:

$$J_n(x) = \frac{2(n+1)}{x} J_{n+1}(x) - J_{n+2}(x) \tag{3}$$

Errors can be calculated similarly to Q1.

Q ...Compute both the absolute and relative errors of these values taking the tabulated values (table-1) as truth....

	$J_0(1)$	$J_0(5)$	$J_0(50)$
ActualValue	$7.6519768656 \times 10^{-1}$	$-1.7759677131 \times 10^{-1}$	$5.5812327669 \times 10^{-2}$
ComputedValue	$7.6519036352 \times 10^{-1}$	$-1.7759388559 \times 10^{-1}$	$5.5807275575 \times 10^{-2}$
AbsoluteError	$7.3230350601 \times 10^{-6}$	$2.8857199839 \times 10^{-6}$	$5.0520941498 \times 10^{-6}$
RelativeError	$9.5701217983 \times 10^{-6}$	$-1.6248718727 \times 10^{-5}$	$9.0519323611 \times 10^{-5}$

Table 3: Comparison of Actual, Computed Values, and Errors for $J_0(x)$ in forward computation.

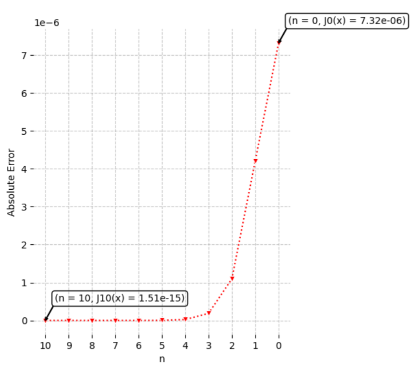

Figure 4: Backward Error x = 1

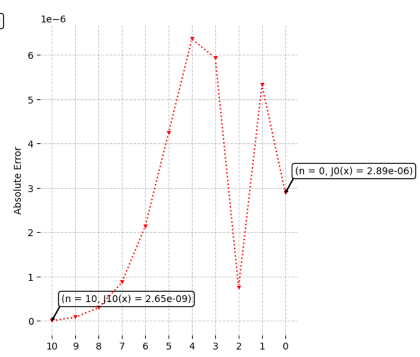

Figure 5: Backward Error x = 5

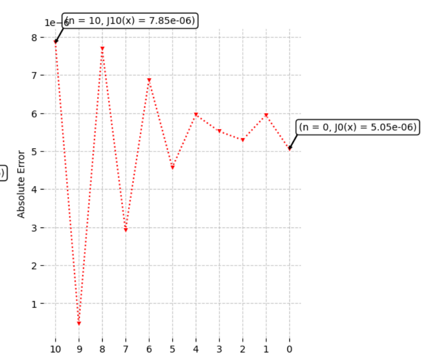

Figure 6: Backward Error x = 50

Observations:

4. From Figure 4, The error growth for x = 1 for backward pass is exponential. The error in backward pass grew from order of 10^{-15} to 10^{-6}.

5. From Figure 5, For backward pass, the error seems to be exponential till n = 10 to 4 and then juggling randomly (not exponential). The error in backward pass grew from order of 10^{-9} to 10^{-6}.

6. From Figure 6, The error growth for x = 50 is juggling randomly (not exponential) The error in backward pass is maintained in order of 10^{-6}.

Explanations:
Observation 4: For $x = 1$, from Table 6, we can see the $\beta \geq 2$ for all iterations, hence the error is increasing exponentially across the whole forward pass in Figure 4.

Observation 5: For $x = 5$, from Table 6, we can see that $\beta \geq 2$ for all values of $n \ \epsilon \ \{8, 7, 6, 5, 4\}$. Hence the error increases exponentially from $n = 10 \ to \ 4$ in and the rest is non-exponential in Figure 5.

Observation 6: For $x = 50$, from Table 6, we can see that $\beta < 2$ for all iterations. Hence the error is juggling randomly (not exponential) in Figure 6.

Q: ...Is the last value computed by the recurrence relation having less or more error compared to the forward approach?

	Direction	$J_0(1)$	$J_0(5)$	$J_0(50)$
AbsoluteError	Backward	$7.3230350601 \times 10^{-6}$	$2.8857199839 \times 10^{-6}$	$5.0520941498 \times 10^{-6}{}_M$
RelativeError	Backward	$9.5701217983 \times 10^{-6}$	$-1.6248718727 \times 10^{-5}$	$9.0519323611 \times 10^{-5}{}_M$
	Direction	$J_{10}(1)$	$J_{10}(5)$	$J_{10}(50)$
AbsoluteError	Forward	$5.6055331000 \times 10^{2}{}_M$	$1.1745331430 \times 10^{-4}{}_M$	$8.8613829735 \times 10^{-7}$
RelativeError	Forward	$2.1308830203 \times 10^{12}{}_M$	$8.0019827270 \times 10^{-2}{}_M$	$-7.7835313005 \times 10^{-6}$

Table 4: Comparison of errors in last iteration for Forward and Backward computation. Max values are marked with subscript $_M$.

From Table 4, it is clear that the forward approach has more absolute and relative error for $x = 1$ and $x = 5$ whereas the backward approach has more relative and absolute error for $x = 50$.

Q3 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]

Solution: The error behavior can be analyzed using difference equation analysis.

We know that to find the class of error growth, we have to write the iterative scheme.

The forward scheme is given by

$$J_i(x) = \frac{2(i-1)}{x} J_{i-1}(x) - J_{i-2}(x)$$

Now let us assume that the $\frac{2(i-1)}{x}$ is a constant represented by β. Our scheme then becomes:

$$J_i(x) = \beta J_{i-1}(x) - J_{i-2}(x) \tag{4}$$

Where β varies with the input x. We can calculate the maximum and minimum values of β to get some intuition about the errors.

As discussed in class, we cannot represent the numbers to their exact precision in computer. Let $\hat{J}_i(x)$ represent the representation we can have in our computers.

$$\hat{J}_i(x) = \beta \hat{J}_{i-1}(x) - \hat{J}_{i-2}(x) \tag{5}$$

Subtracting (5) from (4) we get:

$$e_i(x) = \beta e_{i-1}(x) - e_{i-2}(x)$$

Now, let's assume the error class is exponential

$$e_n \propto k^n \implies k^i = \beta \cdot k^{i-1} - k^{i-2} \implies k^i - \beta \cdot k^{i-1} + k^{i-2} = 0 \implies k^{i-2}\left(k^2 - \beta k + 1\right) = 0$$

$$\implies k = \frac{\beta \pm \sqrt{\beta^2 - 4}}{2}$$

Since $\beta > 0$, the roots of the equation will be real only if $\beta \geq 2$, which is also the condition for which the error growth will be exponential as:

$$k_1 = \frac{\beta + \sqrt{\beta^2 - 4}}{2} \; and \; k_2 = \frac{\beta - \sqrt{\beta^2 - 4}}{2}$$
$$\epsilon_n = C_1 \cdot k_1^n + C_2 \cdot k_2^n$$
$$since \; k_2 \leq 1 \; and \; k_1 \geq 1 \; \forall \; \beta \geq 2$$

For backward computation, we can use the same scheme except for the following changes:

$$\epsilon_n \propto k^{10-n}$$

$$e_i(x) = \beta e_{i+1}(x) - e_{i+2}(x)$$

$$\implies k^{10-i} = \beta \cdot k^{10-i-1} - k^{10-i-2} \implies k^{10-i-2}\left(k^2 - \beta k + 1\right) = 0$$

$$\implies k = \frac{\beta \pm \sqrt{\beta^2 - 4}}{2}$$

Which gives us the same condition $\beta \geq 2$.

Now, we will calculate the values of β that we will get for both forward and backward pass in an attempt to justify our claims.

n	$J_n(1)$	$J_n(5)$	$J_n(50)$
2	$\mathbf{2.00}_e$	0.40	0.04
3	$\mathbf{4.00}_e$	0.80	0.08
4	$\mathbf{6.00}_e$	1.20	0.12
5	$\mathbf{8.00}_e$	1.60	0.16
6	$\mathbf{10.00}_e$	$\mathbf{2.00}_e$	0.20
7	$\mathbf{12.00}_e$	$\mathbf{2.40}_e$	0.24
8	$\mathbf{14.00}_e$	$\mathbf{2.80}_e$	0.28
9	$\mathbf{16.00}_e$	$\mathbf{3.20}_e$	0.32
10	$\mathbf{18.00}_e$	$\mathbf{3.60}_e$	0.36

Table 5: Forward values

n	$J_n(1)$	$J_n(5)$	$J_n(50)$
8	$\mathbf{18.00}_e$	$\mathbf{3.60}_e$	0.36
7	$\mathbf{16.00}_e$	$\mathbf{3.20}_e$	0.32
6	$\mathbf{14.00}_e$	$\mathbf{2.80}_e$	0.28
5	$\mathbf{12.00}_e$	$\mathbf{2.40}_e$	0.24
4	$\mathbf{10.00}_e$	$\mathbf{2.00}_e$	0.20
3	$\mathbf{8.00}_e$	1.60	0.16
2	$\mathbf{6.00}_e$	1.20	0.12
1	$\mathbf{4.00}_e$	0.80	0.08
0	$\mathbf{2.00}_e$	0.40	0.04

Table 6: Backward values

In Table 5 and Table 6, the values at which the growth will be exponential $\beta \geq 2$ are boldened and marked with subscript $_e$.

These β values are used to explain *Observations 1-6* in **Q1** and **Q2** in the corresponding *Explanations* section.

CODE (Python)

```
# %% [markdown]
# # DS288-2024 Numerical Methods
# ## Homework-1
#
# **Naman Pesricha** Mtech CDS **SR-24115**
#
# -------------------

# %%
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.core.display import display, HTML
import warnings

warnings.filterwarnings("ignore")

pd.set_option('display.float_format', lambda x: f'{x:.10e}')

# This function is taken from https://stackoverflow.com/questions/38783027/
def display_side_by_side(dfs:list, captions:list, tablespacing=5):
    """Display tables side by side to save vertical space
    Input:
        dfs: list of pandas.DataFrame
        captions: list of table captions
    """
    output = ""
    for (caption, df) in zip(captions, dfs):
        output += df.style.set_table_attributes("style='display:inline'").set_caption(
            caption)._repr_html_()
        output += tablespacing * "\xa0"
    display(HTML(output))

# %% [markdown]
# ### Loading the dataset

# %%
data = pd.read_csv('./data.csv')
data = data.set_index('n')
computed_forward = data.copy(deep=True)
computed_forward.loc[:,:] = 0
computed_backward = computed_forward.copy(deep=True)
data

# %% [markdown]
# ## Q1

# %% [markdown]
# ### Forward Computation
# - We will initialize the first two rows ( $J_0(x)$ and $J_1(x)$ ) from values from the
    table using only the first 5 digits and compute forward.
# - Iterative scheme rearranged for forward computation:
#     $$J_n(x) = \frac{2(n-1)}{x}J_{n-1}(x) - J_{n-2}(x)$$
# - Absolute error can be calculated using: $$|(data - \hat{data})|$$
# - Relative error can be calculated using: $$|\frac{(data - \hat{data})}{data}|$$

# %%
computed_forward.iloc[:2, :] = [[7.6519e-01,-1.7759e-01,5.5812e-02],
                                 [4.4005e-01,-3.2757e-01,-9.7511e-02]]

```

```python
for i in range(2,11):
    computed_forward.loc[i,'Jn(1)'] = computed_forward.loc[i-1,'Jn(1)']*2*(i-1)/1 - computed_forward.loc[i-2,'Jn(1)']
    computed_forward.loc[i,'Jn(5)'] = computed_forward.loc[i-1,'Jn(5)']*2*(i-1)/5 - computed_forward.loc[i-2,'Jn(5)']
    computed_forward.loc[i,'Jn(50)'] = computed_forward.loc[i-1,'Jn(50)']*2*(i-1)/50 - computed_forward.loc[i-2,'Jn(50)']

computed_forward
absolute_error_computed_forward = abs(computed_forward-data)
relative_error_computed_forward = absolute_error_computed_forward/data
forward_result = pd.DataFrame(
    index=['Actual_Value', 'Computed_Value','Absolute_Error','Relative_Error'], columns= ['Jn(1)','Jn(5)','Jn(50)']
)
forward_result.loc['Actual_Value'] = data.loc[10]
forward_result.loc['Computed_Value'] = computed_forward.loc[10]
forward_result.loc['Absolute_Error'] = absolute_error_computed_forward.loc[10]
forward_result.loc['Relative_Error'] = relative_error_computed_forward.loc[10]
forward_result.columns=['J10(1)','J10(5)','J10(50)']
print("Table 1.1 : Result of Forward Computation.")
forward_result

# %% [markdown]
# -------

# %% [markdown]
# ## Q2

# %% [markdown]
# ### Backward Computation
# - We will initialize the last two rows ( $J_{10}(x)$ and $J_9(x)$ ) from values from the table using only the first 5 digits and compute backward.
# - Iterative scheme rearranged for backeard computation:
#     $J_{n}(x) = \frac{2(n+1)}{x}J_{n+1}(x) - J_{n+2}(x)$
# - Errors can be calculated similarly to Q1.

# %%
computed_backward.iloc[-2:, :] = [[5.2492e-09,5.5202e-03,-2.7192e-02],
                                  [2.6306e-10,1.4678e-03,-1.1384e-01]]

for i in range(8,-1,-1):
    computed_backward.loc[i,'Jn(1)'] = computed_backward.loc[i+1,'Jn(1)']*2*(i+1)/1 - computed_backward.loc[i+2,'Jn(1)']
    computed_backward.loc[i,'Jn(5)'] = computed_backward.loc[i+1,'Jn(5)']*2*(i+1)/5 - computed_backward.loc[i+2,'Jn(5)']
    computed_backward.loc[i,'Jn(50)'] = computed_backward.loc[i+1,'Jn(50)']*2*(i+1)/50 - computed_backward.loc[i+2,'Jn(50)']

computed_backward
absolute_error_computed_backward = abs(computed_backward-data)
relative_error_computed_backward = absolute_error_computed_backward/data

backward_result = pd.DataFrame(
    index=['Actual_Value', 'Computed_Value','Absolute_Error','Relative_Error'],
    columns= ['Jn(1)','Jn(5)','Jn(50)']
    )
backward_result.loc['Actual_Value'] = data.loc[0]
backward_result.loc['Computed_Value'] = computed_backward.loc[0]
backward_result.loc['Absolute_Error'] = absolute_error_computed_backward.loc[0]
backward_result.loc['Relative_Error'] = relative_error_computed_backward.loc[0]
backward_result.columns=['J0(1)','J0(5)','J0(50)']
print('Table 1.2: Result of Backward Computation.')
backward_result
```

```python
# %% [markdown]
# #### From the computed tables *(Table 1.1 and Table 1.2)* it is evident that :
# - Absolute Error for x = 1 is more in Forward Computation.
# - Absolute Error for x = 5 is more in Forward Computation.
# - Absolute Error for x = 50 is more in Backward Computation.
# - Relative Error for x = 1 is more in Forward Computation.
# - Relative Error for x = 5 is more in Forward Computation.
# - Relative Error for x = 50 is more in Backward Computation.

# %% [markdown]
# -----

# %% [markdown]
# ## Q3

# %% [markdown]
# ### Plotting absolute error for forward compuation and backward computation

# %%
fig =plt.figure(figsize=(20, 15))
custom_xticks = range(0, 11, 1)

def annotate_points(ax, data, y_col):
    for i in [0, 10]:
        ax.annotate(
            f"(n = {data.index[i]}, J{data.index[i]}(x) = {data[y_col].iloc[i]:.2e})",
            (data.index[i], data[y_col].iloc[i]),
            textcoords="offset points",
            xytext=(10,20),
            fontsize=10,
            bbox=dict(facecolor="none", edgecolor='black', boxstyle='round,pad=0.3'),
            arrowprops=dict(facecolor='black', shrinkA=5, shrinkB=5, width=0.5, headwidth=3, headlength=3)
        )

def subplot_borders_off(ax):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.spines['left'].set_visible(False)
    ax.spines['bottom'].set_visible(False)


# Plot for x = 1
ax1 = plt.subplot(3, 2, 2)
sns.lineplot(data=absolute_error_computed_backward, x='n', y='Jn(1)', marker='v',  linestyle=':', color= 'red', ax=ax1)
plt.title('Fig 1.2 Error for x = 1 [Backward]')
plt.xlabel('n')
plt.ylabel('Absolute Error')
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(ticks=custom_xticks)
plt.gca().invert_xaxis()
annotate_points(ax1, absolute_error_computed_backward, 'Jn(1)')
subplot_borders_off(ax1)

# Plot for x = 5
ax2 = plt.subplot(3, 2, 4)
sns.lineplot(data=absolute_error_computed_backward, x='n', y='Jn(5)', marker='v',  linestyle=':', color= 'red', ax= ax2)
plt.title('Fig 1.4 Error for x = 5 [Backward]')
plt.xlabel('n')
plt.ylabel('Absolute Error')
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(ticks=custom_xticks)
plt.gca().invert_xaxis()
```

```python
annotate_points(ax2, absolute_error_computed_backward, 'Jn(5)')
subplot_borders_off(ax2)

# Plot for x = 50
ax3 = plt.subplot(3, 2, 6)
sns.lineplot(data=absolute_error_computed_backward, x='n', y='Jn(50)', marker='v', linestyle=':', color= 'red')
plt.title('          Fig 1.6 Error for x = 50 [Backward]')
plt.xlabel('n')
plt.ylabel('Absolute Error')
plt.xticks(ticks=custom_xticks)
plt.grid(True, linestyle='--', alpha=0.7)
plt.gca().invert_xaxis()
annotate_points(ax3, absolute_error_computed_backward, 'Jn(50)')
subplot_borders_off(ax3)

# Plot for x = 1
ax4 = plt.subplot(3, 2, 1)
sns.lineplot(data=absolute_error_computed_forward, x='n', y='Jn(1)', marker='^',  linestyle=':', color= 'orange')
plt.title('Fig 1.1 Error for x = 1 [Forward]')
plt.xlabel('n')
plt.ylabel('Absolute Error')
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(ticks=custom_xticks)
annotate_points(ax4, absolute_error_computed_forward, 'Jn(1)')
subplot_borders_off(ax4)

# Plot for x = 5
ax5 = plt.subplot(3, 2, 3)
sns.lineplot(data=absolute_error_computed_forward, x='n', y='Jn(5)', marker='^',  linestyle=':', color= 'orange')
plt.title('Fig 1.3 Error for x = 5 [Forward]')
plt.xlabel('n')
plt.ylabel('Absolute Error')
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(ticks=custom_xticks)
annotate_points(ax5, absolute_error_computed_forward, 'Jn(5)')
subplot_borders_off(ax5)

# Plot for x = 50
ax6 = plt.subplot(3, 2, 5)
sns.lineplot(data=absolute_error_computed_forward, x='n', y='Jn(50)', marker='^',  linestyle=':', color= 'orange')
plt.title('Fig 1.5 Error for x = 50 [Forward]')
plt.xlabel('n')
plt.ylabel('Absolute Error')
plt.xticks(ticks=custom_xticks)
plt.grid(True, linestyle='--', alpha=0.7)
annotate_points(ax6, absolute_error_computed_forward, 'Jn(50)')
subplot_borders_off(ax6)

print("Forward and Backward absolute errors against n")

plt.tight_layout()
plt.show()

# %% [markdown]
# ------
# ### Observations from
# **From the above graphs [Fig 1.1 - 1.6], we can make the following observations:**
# 1. The error growth for x = 1 for both forward and backward pass is exponential *(Fig 1.1, 1.2)*.
#     1. The error in forward pass grew from order of $10^{-6}$ to order of $10^{2}$ *(Fig 1.1)*.
```

```
#     2. The error in backward pass grew from order of $10^{-15}$ to $10^{-6}$ *(Fig 1.2)*.
# 2. The error growth for x = 5 for forward pass seems to be exponential at around from n = {6 to 10}. For backward pass, the error seems to be exponential till n = {10 to 4} and then juggling randomly (not exponential) *(Fig 1.3 , 1.4)*.
#     1. The error in forward pass grew from order of $10^{-6}$ to order of $10^{-4}$ *(Fig 1.3)*.
#     2. The error in backward pass grew from order of $10^{-9}$ to $10^{-6}$ *(Fig 1.4)*.
# 3. The error growth for x = 50 is juggling randomly (not exponential) for both forward pass and backward pass *(Fig 1.5, 1.6)*.
#     1. The error in forward pass maintained in order of $10^{-7}$ *(Fig 1.5)*.
#     2. The error in backward pass maintained in order of $10^{-6}$ *(Fig 1.6)*.
#
# -----

# %% [markdown]
# ### Difference Equation Analysis
#
# The error behaviour can be analyzed using difference equation analysis.
#
# We know that to find the class of error growth, we have to write the iterative scheme.
#
# The forward scheme is given by for iteration i $$J_{i}(x) = \frac{2(i-1)}{x}J_{i-1}(x) - J_{i-2}(x)$$
#
# Now let us assume that the $\frac{2(i-1)}{x}$ is a constant represented by $\beta$. Our scheme then becomes (1) : $$J_{i}(x) = \beta J_{i-1}(x) - J_{i-2}(x)$$
#
# Where $\beta$ varies with the input x. We can calculate the maximum and minimum values of $\beta$ to get some intuition about the errors.
#
# As discussed in class, we cannot represent the numbers to their exact precision in computer. Let $\hat{J}_{i}(x)$ represent the representation we can have in our computers (2). $$\hat{J}_{i}(x) = \beta \hat{J}_{i-1}(x) - \hat{J}_{i-2}(x)$$
#
# Subtracting equation (2) from (1) we get: $$e_{i}(x) = \beta e_{i-1}(x) - e_{i-2}(x)$$
#
# Now, let's assume the error class is exponential
#
# $$e_n \propto k^n \\ \implies k^i = \beta \cdot k^{i-1} - k^{i-2}   \\   \implies k^i - \beta \cdot k^{i-1} + k^{i-2} = 0   \\   \implies k^{i - 2} \left( k^2 - \beta k + 1 \right) = 0  \\  k = \frac{\beta \pm \sqrt{\beta^2 - 4}}{2}$$
#
# Since $\beta > 0$, the roots of the equation will be real only if $ \beta \geq 2$, which is also the condition for which the error growth will be exponential as:
#
# $$k_1 = \frac{\beta + \sqrt{\beta^2 - 4}}{2} \ and \ k_2 = \frac{\beta - \sqrt{\beta^2 - 4}}{2}$$
# $$\epsilon_n =  C_1 \cdot k_1^n + C_2 \cdot k_2^n$$
# $$since \ k_2 \leq 1 \ and \ k_1 \geq 1 \ \forall \  \beta \geq 2$$
#
#
#
# For backward computation, we can use the same scheme except the following changes: $$\epsilon_{n} \propto k^{10-n}$$
#
# $$ e_{i}(x) = \beta e_{i+1}(x) - e_{i+2}(x)$$
# $$\implies k^{10-i} = \beta \cdot k^{10-i-1} - k^{10-i-2}   \\   \implies k^{10- i - 2} \left( k^2 - \beta k + 1 \right) = 0  \\  k = \frac{\beta \pm \sqrt{\beta^2 - 4}}{2}$$
#
# Which gives us the same condition $\beta \geq 2$.
#
# Now, we will calculate the values of $\beta$ that we will get for both forward and backward pass in an attempt to justify our claims.
#
#
```

```python
# -----

# %%
beta_values_forward = pd.DataFrame(index=[i for i in range(2, 11)], columns=['Jn(1)','Jn(5)','Jn(50)'])
beta_values_backward = pd.DataFrame(index=[i for i in range(8, -1, -1)], columns=['Jn(1)','Jn(5)','Jn(50)'], )
for i in range(2, 11):
    beta_values_forward.loc[i, 'Jn(1)'] = 2*(i-1)/1
    beta_values_forward.loc[i, 'Jn(5)'] = 2*(i-1)/5
    beta_values_forward.loc[i, 'Jn(50)'] = 2*(i-1)/50

for i in range(8, -1, -1):
    beta_values_backward.loc[i, 'Jn(1)'] = 2*(i+1)/1
    beta_values_backward.loc[i, 'Jn(5)'] = 2*(i+1)/5
    beta_values_backward.loc[i, 'Jn(50)'] = 2*(i+1)/50

beta_values_forward
display_side_by_side(
    [beta_values_forward,beta_values_backward],
    ['Table 1.3: Forward Beta Values for computing ith element'
     ,'Table 1.4: Backward Beta Values for computing ith element']
    )

# %% [markdown]
# ------
# ### Coming back to our observations
#
# **Now we will justify all the observations we made from the Fig 1.1 - 1.6 by referring the $\beta$ values from table 1.3, 1.4:**
#
# 1. *'The error growth for x = 1 for both forward and backward pass is exponential (Fig 1.1, 1.2).'*.
#     - This can be explained by the fact that in both forward and backward computation, the value of $\beta \geq 2$ , hence the error grows exponentially in both forward and backward computation.
#
# 2. *'The error growth for x = 5 for forward pass seems to be exponential at around from n = {6 to 10}. For backward pass, the error seems to be exponential till n = {10 to 4} and then juggling randomly (not exponential) (Fig 1.3 , 1.4).'*
#     - From the tables, we can see that in forward computation, $\beta \geq 2$ only for $n \ \epsilon \ \{6,7,8,9,10\}$ explaining exponential error growth in that range and not exponential growth for $n \ \epsilon \ \{0,1,2,3,4\}$.
#     - In the backward computation the condition holds true for values $n \ \epsilon \ \{8,7,6,5,4\}$ explaining the exponential error growth in the corresponding range and not exponential for $n \ \epsilon \ \{3,2,1,0\}$.
#
# 3. *The error growth for x = 50 is juggling randomly (not exponential) for both forward pass and backward pass (Fig 1.5, 1.6).*
#     - From the tables, it is evidnet that all values of $\beta \lt 2$ for both forward pass and backward pass explaining the non exponential error propagation in both.
#
# --------

# %% [markdown]
#
```