Homework-1

August 9, 2024

1 DS288-2024 Numerical Methods

1.1 Homework-1

Naman Pesricha Mtech CDS 06-18-01-10-51-24-1-24115

For ease of indexing, I will right J__n(x) as J__x[n].
Creating list J__1 with all values set to 0 and then initializing J__1[0] and J__1[1]

1.1.1 Q1

Solution Approach

- We will make three empty lists of size lenght 11 for J__1, J__5, J__50.
- We will initialize the first two elements (0th and 1th) with the corresponding values upto five digits (not decimal places?)
- We will rearrange the iterative scheme in a way so that the J[n] = f(J[n-1], J[n-2])
- Use a for loop to update the corresponding values (although this can be done inplace by maintaining only two variables per x value instead of a list, having the whole list has it's own benefit in terms of checking the corectness of the code).

[ ]:
```
J_1_10_truth = 2.6306151237e-10
J_5_10_truth =  1.4678026473e-03
J_50_10_truth = -1.1384784915e-01

J_1 = [0]*11
J_1[0] = 7.6519e-01
J_1[1] = 4.4005e-01

J_5 = [0]*11
J_5[0] = -1.7759e-01
J_5[1] = -3.2757e-01

J_50 = [0]*11
J_50[0] = 5.5812e-02
J_50[1] = -9.7511e-02
```

```python
# Iterating over a for loop from i = 2 to 10 (inclusive) to compute the
↪subsequent values using the iterative scheme
# J_x[n] = J_x[n-1]* 2*(n-1)/x - J_x[n-2] (Rearranging the scheme and then
↪replacing n+1 with n)

for i in range(2,11):
    J_1[i] = J_1[i-1]*2*(i-1)/1 - J_1[i-2]
    J_5[i] = J_5[i-1]*2*(i-1)/5 - J_5[i-2]
    J_50[i] = J_50[i-1]*2*(i-1)/50 - J_50[i-2]

# Computing absolute values
abs_J_1_10 = abs(J_1_10_truth - J_1[10])
abs_J_5_10 = abs(J_5_10_truth - J_5[10])
abs_J_50_10 = abs(J_50_10_truth - J_50[10])

# Computing relative values

rel_J_1_10 = abs_J_1_10/J_1_10_truth
rel_J_5_10 = abs_J_5_10/J_5_10_truth
rel_J_50_10 = abs_J_50_10/J_50_10_truth

print("Computed Values of J_x[10] for Q1")
print(f"J_1[10]  : {J_1[10]}, Actual Value : {J_1_10_truth}")
print(f"J_5[10]  : {J_5[10]}, Actual Value : {J_5_10_truth}")
print(f"J_50[10] : {J_50[10]}, Actual Value : {J_50_10_truth}")

print("\nAbsolute Errors for Q1:")
print(f"abs_J_1_10 : {abs_J_1_10}")
print(f"abs_J_5_10 : {abs_J_5_10}")
print(f"abs_J_50_10 : {abs_J_50_10}")

print("\nRelative Errors for Q1:")
print(f"rel_J_1_10 : {rel_J_1_10}")
print(f"rel_J_5_10 : {rel_J_5_10}")
print(f"rel_J_50_10 : {rel_J_50_10}")
```

```
Computed Values of J_x[10] for Q1
J_1[10]  : 560.5533099964255, Actual Value : 2.6306151237e-10
J_5[10]  : 0.0015852559616038944, Actual Value : 0.0014678026473
J_50[10] : -0.11384696301170265, Actual Value : -0.11384784915

Absolute Errors for Q1:
abs_J_1_10 : 560.5533099961624
abs_J_5_10 : 0.00011745331430389433
abs_J_50_10 : 8.86138297351291e-07

Relative Errors for Q1:
```

```
rel_J_1_10 : 2130883020271.455
rel_J_5_10 : 0.08001982727034036
rel_J_50_10 : -7.783531300479479e-06
```

1.2 ——-

1.2.1 Q2

Solution Approach

- Similar to Q1, we will make three empty lists of size lenght 11 for J__1, J__5, J__50.
- We will initialize the last two elements (10th and 9th) with the corresponding values upto five digits (not decimal places?)
- We will rearrange the iterative scheme in a way so that the J[n] = f(J[n+1], J[n+2])
- Use a for loop to update the corresponding values.

[ ]:
```
J_1_0_truth = 7.6519768656e-01
J_5_0_truth = -1.7759677131e-01
J_50_0_truth =  5.5812327669e-02

J_1 = [0]*11
J_1[9] = 5.2492e-09
J_1[10] = 2.6306e-10

J_5 = [0]*11
J_5[9] = 5.5202e-03
J_5[10] = 1.4678e-03

J_50 = [0]*11
J_50[9] = -2.7192e-02
J_50[10] = -1.1384e-01

# Iterating over a for loop from i = 8 to 0 (inclusive) to compute the␣
↪subsequent values using the iterative scheme
# J_x[n] = J_x[n-1]* 2*(n+1)/x - J_x[n+2] (Rearranging the scheme and then␣
↪replacing n-1 with n)

for i in range(8,-1,-1):
    J_1[i] = J_1[i+1]*2*(i+1)/1 - J_1[i+2]
    J_5[i] = J_5[i+1]*2*(i+1)/5 - J_5[i+2]
    J_50[i] = J_50[i+1]*2*(i+1)/50 - J_50[i+2]

# Computing absolute values
abs_J_1_0 = abs(J_1_0_truth - J_1[0])
abs_J_5_0 = abs(J_5_0_truth - J_5[0])
abs_J_50_0 = abs(J_50_0_truth - J_50[0])

# Computing relative values
```

```python
rel_J_1_0 = abs_J_1_0/J_1_0_truth
rel_J_5_0 = abs_J_5_0/J_5_0_truth
rel_J_50_0 = abs_J_50_0/J_50_0_truth

print("Computed Values of J_x[0] for Q2")
print(f"J_1[0]  : {J_1[0]}, Actual Value : {J_1_0_truth}")
print(f"J_5[0]  : {J_5[0]}, Actual Value : {J_5_0_truth}")
print(f"J_50[0] : {J_50[0]}, Actual Value : {J_50_0_truth}")

print("\nAbsolute Errors for Q2:")
print(f"abs_J_1_0 : {abs_J_1_0}")
print(f"abs_J_5_0 : {abs_J_5_0}")
print(f"abs_J_50_0 : {abs_J_50_0}")

print("\nRelative Errors for Q2:")
print(f"rel_J_1_0 : {rel_J_1_0}")
print(f"rel_J_5_0 : {rel_J_5_0}")
print(f"rel_J_50_0 : {rel_J_50_0}")
```

```
Computed Values of J_x[0] for Q2
J_1[0]  : 0.7651903635249399, Actual Value : 0.76519768656
J_5[0]  : -0.1775938855900161, Actual Value : -0.17759677131
J_50[0] : 0.055807275574850226, Actual Value : 0.055812327669

Absolute Errors for Q2:
abs_J_1_0 : 7.323035060124994e-06
abs_J_5_0 : 2.885719983913848e-06
abs_J_50_0 : 5.052094149776698e-06

Relative Errors for Q2:
rel_J_1_0 : 9.570121798258712e-06
rel_J_5_0 : -1.6248718727418446e-05
rel_J_50_0 : 9.051932361141778e-05
```

The last value computed by the backward approach is having less relative error when compared to the forward approach as observed above.

1.3 ———-

1.3.1 Q3