assignment-2

August 28, 2024

[1]:
```
#1. Write a program to find the value of f (0.4) using Lagrange's interpolation formula for the
# table of values given below.
# x 0.3 0.5 0.6
# f(x) 0.61 0.69 0.7
#-------------------------------------------------------------------------------------------
x = [0.3, 0.5, 0.6]
f_x = [0.61, 0.69, 0.72]

x_interpolate = 0.4

def lagrange_interpolation(x, f_x, x_interpolate):
    n = len(x)
    result = 0.0

    for i in range(n):
        term = f_x[i]
        for j in range(n):
            if j != i:
                term = term * (x_interpolate - x[j]) / (x[i] - x[j])
        result += term

    return result

# Calculate f(0.4) using Lagrange's interpolation formula
f_interpolate = lagrange_interpolation(x, f_x, x_interpolate)
print(f"The value of f(0.4) is approximately: {f_interpolate:.4f}")
```

```
The value of f(0.4) is approximately: 0.6533
```

[11]:
```
# 2. Write a program to find the value of f (2) using Newton's Divided Difference interpolation
# formula for the table of values given below.
# x -1 0 3 6 7
# f(x) 3 -6 39 822 161
#-------------------------------------------------------------------------------------------
x = [-1, 0, 3, 6, 7]
```

```python
f_x = [3, -6, 39, 822, 1611]

# Function to calculate divided differences table and print them
def divided_differences(x, f_x):
    n = len(x)
    diff_table = [[0 for _ in range(n)] for _ in range(n)]

    for i in range(n):
        diff_table[i][0] = f_x[i]

    for j in range(1, n):
        for i in range(n-j):
            diff_table[i][j] = (diff_table[i+1][j-1] - diff_table[i][j-1]) /␣
 ↪(x[i+j] - x[i])

        print(f"{j} order divided differences: {[diff_table[i][j] for i in␣
 ↪range(n-j)]}")

    return [diff_table[0][i] for i in range(n)]

# Function to calculate the value at a given x using Newton's interpolation
def newtons_interpolation(x, f_x, x_interpolate):
    n = len(x)

    co= divided_differences(x, f_x)
    result = co[0]

    # Add subsequent terms
    for i in range(1, n):
        term = co[i]
        for j in range(i):
            term *= (x_interpolate - x[j])
        result += term

    return result
x_interpolate = 2

# Calculate f(2) using Newton's Divided Difference interpolation formula
f_interpolate = newtons_interpolation(x, f_x, x_interpolate)
print(f"\nThe value of f(2) is approximately: {f_interpolate:.4f}")
```

```
1 order divided differences: [-9.0, 15.0, 261.0, 789.0]
2 order divided differences: [6.0, 41.0, 132.0]
3 order divided differences: [5.0, 13.0]
4 order divided differences: [1.0]

The value of f(2) is approximately: 6.0000
```

[7]:
```
# 3. Write a program to find the value of f(0.5) by using Forward Difference␣
 ↪interpolation on
# the following table of data.
# x 0 1 2 3 4
# f(x) 1 4 16 64 256
#␣
 ↪-------------------------------------------------------------------------------------------#
x = [0, 1, 2, 3, 4]
f_x = [1, 4, 16, 64, 256]

# Function to calculate forward differences table and print them
def forward_differences(f_x, num_orders):
    n = len(f_x)
    diff_table = [f_x[:]]

    for i in range(1, num_orders + 1):
        current_diffs = []
        for j in range(n - i):
            diff = diff_table[i-1][j+1] - diff_table[i-1][j]
            current_diffs.append(diff)
        diff_table.append(current_diffs)
        if i == 1:
            print(f"First forward differences: {current_diffs}")
        elif i == 2:
            print(f"Second forward differences: {current_diffs}")
        elif i == 3:
            print(f"Third forward differences: {current_diffs}")
        else:
            print(f"{i}th forward differences: {current_diffs}")

    return diff_table
def forward_interpolation(x, f_x, x_interpolate, num_orders):
    n = len(x)
    h = x[1] - x[0]  # Assuming equal intervals
    u = (x_interpolate - x[0]) / h

    # Calculate forward differences
    diff_table = forward_differences(f_x, num_orders)
    result = diff_table[0][0]
    u_term = 1

    for i in range(1, num_orders + 1):
        u_term *= (u - i + 1) / i
        result += u_term * diff_table[i][0]

    return result
x_interpolate = 0.5
```

```
num_orders = 4

# Calculate f(0.5) using Forward Difference interpolation
f_interpolate = forward_interpolation(x, f_x, x_interpolate, num_orders)
print(f"\nThe value of f(0.5) is approximately: {f_interpolate:.4f}")
```

```
First forward differences: [3, 12, 48, 192]
Second forward differences: [9, 36, 144]
Third forward differences: [27, 108]
4th forward differences: [81]

The value of f(0.5) is approximately: -0.1016
```

[10]:
```
# 4.Write a program to find the value of f(7.5) by using Backward Difference␣
 ↪interpolation on
# the following table of data.
# x 3 4 5 6 7 8
# f(x) 28 65 126 217 344 513
#-----------------------------------------------------------------------------------------
x = [3, 4, 5, 6, 7, 8]
f_x = [28, 65, 126, 217, 344, 513]

# Function to calculate backward differences table and print them
def backward_differences(f_x, num_orders):
    n = len(f_x)
    diff_table = [f_x[:]]

    for i in range(1, num_orders + 1):
        current_diffs = []
        for j in range(n - i):
            diff = diff_table[i-1][j+1] - diff_table[i-1][j]
            current_diffs.append(diff)
        diff_table.append(current_diffs)

        if i == 1:
            print(f"First backward differences: {current_diffs[::-1]}")
        elif i == 2:
            print(f"Second backward differences: {current_diffs[::-1]}")
        elif i == 3:
            print(f"Third backward differences: {current_diffs[::-1]}")
        else:
            print(f"{i}th backward differences: {current_diffs[::-1]}")

    return diff_table
def backward_interpolation(x, f_x, x_interpolate, num_orders):
    n = len(x)
    h = x[1] - x[0]
```

```
    u = (x_interpolate - x[-1]) / h

    # Calculate backward differences
    diff_table = backward_differences(f_x, num_orders)

    # Initialize result with f(xn)
    result = diff_table[0][-1]
    u_term = 1  # Initialize with u

    for i in range(1, num_orders + 1):
        u_term *= (u + i - 1) / i
        result += u_term * diff_table[i][-1]

    return result
x_interpolate = 7.5
num_orders = 5

# Calculate f(7.5) using Backward Difference interpolation
f_interpolate = backward_interpolation(x, f_x, x_interpolate, num_orders)
print(f"\nThe value of f(7.5) is approximately: {f_interpolate:.4f}")
```

```
First backward differences: [169, 127, 91, 61, 37]
Second backward differences: [42, 36, 30, 24]
Third backward differences: [6, 6, 6]
4th backward differences: [0, 0]
5th backward differences: [0]

The value of f(7.5) is approximately: 422.8750
```

[ ]: