Lab-2 Nm

September 12, 2023

[ ]:
```
# ID: C201050
# Name: AYAT ULLAH
# Sec: 7BM
# Course Code: CSE-4746
# Course Title: Numerical Methods Lab
#
```

[1]:
```
# The following values of f (x) are given.
# x 1 2 3 4 5
# y = f(x) 1 8 27 64 125
# Write a program to find difference table for the above values.
```

```
  Cell In[1], line 1
    The following values of f (x) are given.
        ^
SyntaxError: invalid syntax
```

[2]:
```
def create_difference_table(x_values, y_values):
    n = len(x_values)
    difference_table = []

    # Initialize the first column of the difference table with y_values
    difference_table.append(y_values)

    for i in range(1, n):
        next_row = []
        for j in range(n - i):
            # Calculate the difference between adjacent values
            difference = difference_table[i - 1][j + 1] - difference_table[i -
 ↪1][j]
            next_row.append(difference)
        difference_table.append(next_row)

    return difference_table
```

```
# Given values
x_values = [1, 2, 3, 4, 5]
y_values = [1, 8, 27, 64, 125]

# Create the difference table
difference_table = create_difference_table(x_values, y_values)

# Display the difference table
for row in difference_table:
    print(row)
```

```
[1, 8, 27, 64, 125]
[7, 19, 37, 61]
[12, 18, 24]
[6, 6]
[0]
```

[3]:
```
# Write a program to find the values of y when x = 1.7 by using Newton&#39;s␣
↪forward interpolation
# formula.
```

[22]:
```
# Define the factorial function
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

# Given values
data_points = [1, 8, 27, 64, 125]
initial_value = 1
target_value = 1.7
initial_x = 1
step_size = 1
u = (target_value - initial_x) / step_size

# Initialize an empty list for forward differences
forward_diff = []

# Determine the number of iterations based on the number of data points
num_iterations = len(data_points) - 1

while num_iterations:
    # Initialize an empty list for the differences in each iteration
    iteration_diff = []
```

```
    for i in range(1, num_iterations + 1):
        # Calculate the forward differences
        iteration_diff.append(data_points[i] - data_points[i - 1])

    # Append the first difference to the list
    forward_diff.append(iteration_diff[0])

    # Update the data points for the next iteration
    data_points = iteration_diff
    num_iterations -= 1

# Round the forward differences to four decimal places
forward_diff = list(map(lambda x: round(x, 4), forward_diff))

# Initialize variables for the result, subtract, and product
result = initial_value
subtract_value = 0
product = 1

# Calculate the Newton interpolation polynomial
for delta in forward_diff:
    product *= u - subtract_value
    factorial_value = factorial(subtract_value + 1)
    quotient = delta / factorial_value
    result += product * quotient
    subtract_value += 1

# Print the result with four decimal places
print(f"{result:.4f}")
```

```
4.9130
```

[24]:

```
# The following values of f (x) are given.
# x 1 2 3 4 5
# y = f(x) 1 8 27 64 125
# Write a program to find the values of y when x = 4.7 by using Newton&#39;s␣
↪backward interpolation
# formula.
```

[29]:

```
# Define the factorial function
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

# Given values
```

```python
data_points = [1, 8, 27, 64, 125]
final_value = 125
initial_xn = 5
target_x = 4.7
step_size = 1
u = (target_x - initial_xn) / step_size

# Initialize an empty list for backward differences
backward_diff = []

# Determine the number of iterations based on the number of data points
num_iterations = len(data_points) - 1

while num_iterations:
    # Initialize an empty list for the differences in each iteration
    iteration_diff = []

    for i in range(1, num_iterations + 1):
        # Calculate the backward differences
        iteration_diff.append(data_points[i] - data_points[i - 1])

    # Append the last difference to the list
    backward_diff.append(iteration_diff[-1])

    # Update the data points for the next iteration
    data_points = iteration_diff
    num_iterations -= 1

# Round the backward differences to four decimal places
backward_diff = list(map(lambda x: round(x, 4), backward_diff))

# Initialize variables for the result, add, and product
result = final_value
add_value = 0
product = 1

# Calculate the Newton interpolation polynomial
for delta in backward_diff:
    product *= u + add_value
    factorial_value = factorial(add_value + 1)
    quotient = delta / factorial_value
    result += product * quotient
    add_value += 1

# Print the result with four decimal places
print(f"{result:.4f}")
```

```
103.8230
```

[ ]:
```
# The following values of f (x) are given.
# x 1 2 3 4 5
# y = f(x) 1 8 27 64 125
# Write a program to find the values of x for which f (x) = 85 by using␣
↪Lagrange's inverse
# interpolation formula.
```

[33]:
```
def lagrange_interpolation(x, y, Y):
    ans = 0
    for i in range(len(x)):
        denum = 1
        num = 1
        for j in range(len(x)):
            if i==j:
                continue
            denum *= (Y - y[j])
            num *= (y[i] - y[j])
        ans += x[i]*(denum/num)
    return ans

# Input data
x = [1, 2, 3, 4, 5]
y = [1, 8, 27, 64, 125]
Y = 85

# Calculate the interpolated value
result = lagrange_interpolation(x, y, Y)
print(f"The interpolated value at Y={Y} is approximately {result:.4f}")
```

```
The interpolated value at Y=85 is approximately 5.6469
```

[34]:
```
# The following values of f (x) are given. Prepare the divided difference table␣
↪for the following data
# x 1 3 4 6 10
# y = f(x) 0 18 58 190 920
# Write a program to find the values of y when x = 2.7 by using Newton&#39;s␣
↪divided difference formula.
```

[36]:
```
def interpolate(x_values, y_values, target_x):
    n = len(y_values) - 1
    inc = 1
    diff = []

    # Calculate divided differences
    while n:
        j = 0
        fx = []
```

```python
        for i in range(1, n + 1):
            denum = y_values[i - 1] - y_values[i]
            num = x_values[j] - x_values[j + inc]
            fx.append(denum / num)
            j += 1
        diff.append(fx[0])
        inc += 1
        y_values = fx
        n -= 1

    # Round the divided differences
    diff = [round(x, 4) for x in diff]

    # Initialize the answer
    interpolated_value = 0
    prod = 1

    # Calculate the interpolated value
    for i in range(len(diff)):
        prod *= (target_x - x_values[i])
        interpolated_value += prod * diff[i]

    return round(interpolated_value, 4)

# Given data
x_values = [1, 3, 4, 6, 10]
y_values = [0, 18, 58, 190, 920]
target_x = 2.7

# Calculate and print the interpolated value
result = interpolate(x_values, y_values, target_x)
print(f"Interpolated value at x = {target_x}: {result}")
```

```
Interpolated value at x = 2.7: 9.3546
```