Numerical Methods
Lab 5.Numerical Differentiation 2

Maryna Borovyk

November 3, 2024

Task 1

[37]:
```
import numpy as np
from sympy import Symbol, lambdify, exp, sin
import matplotlib.pyplot as plt
```

Defining the function f(x)

[38]:
```
def f(x):
    return np.exp(np.sin(2 * x))
```

Defining the numerical derivative function using the given approximation

[39]:
```
def numerical_derivative(x, h):
    f_derivative = (f(x + h / 2) - f(x - h / 2)) / h
    return f_derivative
```

Defining the analytical derivative function

[40]:
```
def analytical_derivative_f(x_value):
    x = Symbol('x')
    f = exp(sin(2 * x))
    f_derivative = f.diff(x)
    f_derivative_func = lambdify(x, f_derivative)
    return f_derivative_func(x_value)
```

Point at which the derivative is to be calculated

[41]:
```
x = 0.5
```

Calculating the derivative and absolute error for different values of h

[42]:
```
n_values = range(1, 12)
h_values = [10**-n for n in n_values]
errors = []

for h in h_values:
    num_derivative = numerical_derivative(x, h)
    analytical_valuee = analytical_derivative_f(x)
```

```
    absolute_error = np.abs(num_derivative - analytical_valuee)
    errors.append(absolute_error)
```

[43]:

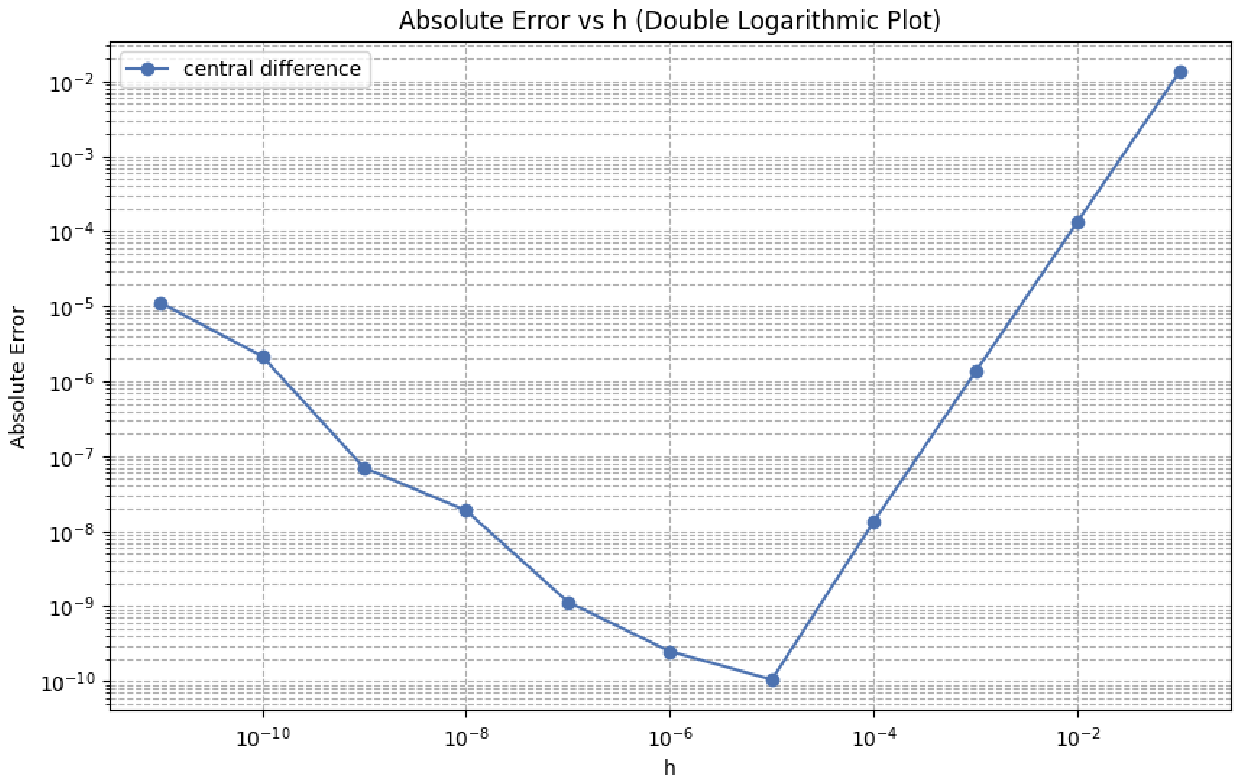
```
for h, error in zip(h_values, errors):
    print(f"h = {h:.1e}, Absolute Error = {error:.5e}")
```

```
h = 1.0e-01, Absolute Error = 1.34656e-02
h = 1.0e-02, Absolute Error = 1.35047e-04
h = 1.0e-03, Absolute Error = 1.35051e-06
h = 1.0e-04, Absolute Error = 1.35078e-08
h = 1.0e-05, Absolute Error = 1.05175e-10
h = 1.0e-06, Absolute Error = 2.50096e-10
h = 1.0e-07, Absolute Error = 1.13827e-09
h = 1.0e-08, Absolute Error = 1.89018e-08
h = 1.0e-09, Absolute Error = 6.99160e-08
h = 1.0e-10, Absolute Error = 2.15053e-06
h = 1.0e-11, Absolute Error = 1.11721e-05
```

Task 2

[44]:

```
plt.figure(figsize=(10, 6))
plt.style.use('seaborn-v0_8-deep')
plt.loglog(h_values, errors, marker='o', label='central difference')
plt.xlabel('h')
plt.ylabel('Absolute Error')
plt.title('Absolute Error vs h (Double Logarithmic Plot)')
plt.legend()
plt.grid(True, which="both", ls="--")
plt.show()
```



Deriving the estimated value of h for minimum error

[45]:
```
estimated_error =np.argmin(errors)
estimated_h = h_values[estimated_error]
print(f"Estimated h for minimum error: {estimated_h:.1e}")
```

```
Estimated h for minimum error: 1.0e-05
```

Task 3

Comparing the results with two-point forward difference method

[46]:
```
from lab4 import derivative

errors_2 = []

for h in h_values:
    num_derivative = derivative(f, x, h)
    analytical_valuee = analytical_derivative_f(x)
    absolute_error = np.abs(num_derivative - analytical_valuee)
    errors_2.append(absolute_error)
```

[47]:

```
errors_2
```

[47]:

```
[0.30770445833762494,
 0.02603591569011865,
 0.0025550421497806397,
 0.00025501809412364906,
 2.549695425191345e-05,
 2.549266057805255e-06,
 2.5643346734938177e-07,
 2.550707822734921e-08,
 6.991599921235547e-08,
 2.1505300500379576e-06,
 3.323677473954234e-05]
```

[48]:

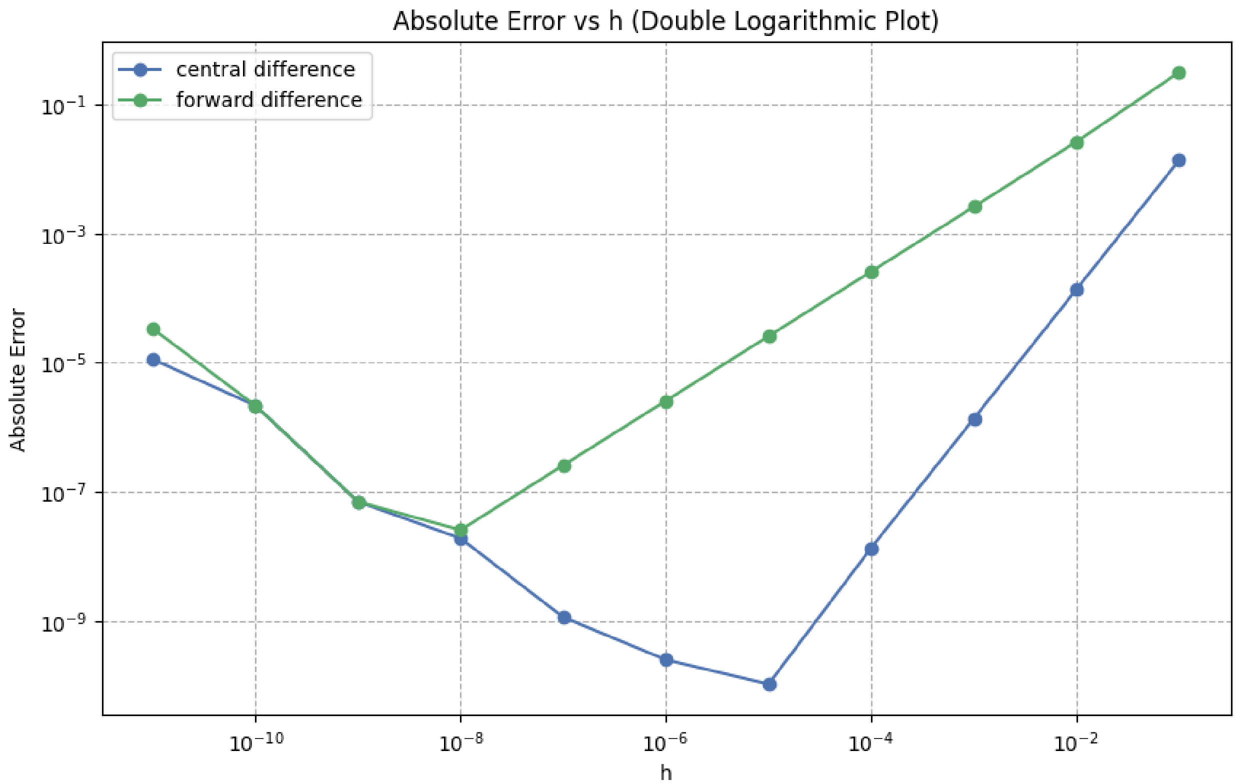
```
errors
```

[48]:

```
[0.013465609469768935,
 0.00013504724926516332,
 1.3505116287504393e-06,
 1.350778777720052e-08,
 1.0517542392562973e-10,
 2.5009594395442036e-10,
 1.1382743636545456e-09,
 1.890184275765705e-08,
 6.991599921235547e-08,
 2.1505300500379576e-06,
 1.1172146245463921e-05]
```

[49]:

```
plt.figure(figsize=(10, 6))
plt.style.use('seaborn-v0_8-deep')
plt.loglog(h_values, errors, marker='o', label='central difference')
plt.loglog(h_values, errors_2, marker='o', label='forward difference')
plt.xlabel('h')
plt.ylabel('Absolute Error')
plt.title('Absolute Error vs h (Double Logarithmic Plot)')
plt.legend()
plt.grid(True, which="both", ls="--")
plt.show()
```



Task 4

Deriving the formula for an approximate value of the second derivative using the given equations Start with the given equations:

$$f\left(x+\frac{h}{2}\right)=f(x)+\frac{h}{2}f'(x)+\frac{h^2}{8}f''(x)+\frac{h^3}{48}f'''(x)+\dots$$

$$f\left(x-\frac{h}{2}\right)=f(x)-\frac{h}{2}f'(x)+\frac{h^2}{8}f''(x)-\frac{h^3}{48}f'''(x)+\dots$$

Add the two equations to eliminate the odd derivatives:

$$f\left(x+\frac{h}{2}\right)+f\left(x-\frac{h}{2}\right)=2f(x)+\frac{h^2}{4}f''(x)+\dots$$

Rearrange to solve for $f''(x)$:

$$f''(x)\approx\frac{8}{h^2}\left(\frac{f\left(x+\frac{h}{2}\right)+f\left(x-\frac{h}{2}\right)-2f(x)}{2}\right)$$

Thus, the formula for the approximate value of the second derivative is:

[50]:
```python
def second_derivative(f, x, h):
    return (8 / h**2) * ((f(x + h/2) + f(x - h/2) - 2 * f(x)) / 2)
```

Analyzing the relationship between the error and h for same function f(x)

[51]:
```python
errors_3 = []
```

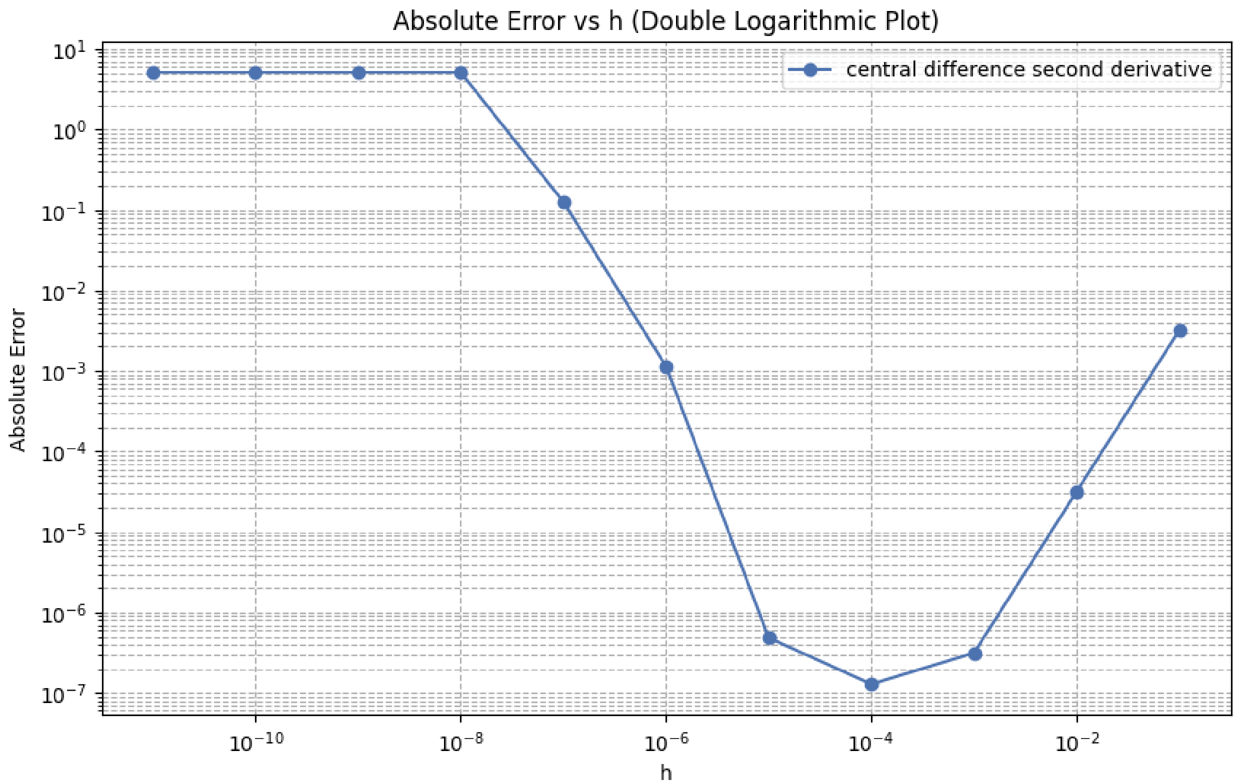
[52]:
```python
def analytical_second_derivative_f(x_value):
    x = Symbol('x')
    f = exp(sin(2 * x))
    f_derivative = f.diff(x)
    f_second_derivative = f_derivative.diff(x)
    f_derivative_func = lambdify(x, f_second_derivative)
    return f_derivative_func(x_value)
```

[53]:
```python
for h in h_values:
    num_derivative = second_derivative(f, x, h)
    analytical_value2 = analytical_second_derivative_f(x)
    absolute_error = np.abs(num_derivative - analytical_value2)
    errors_3.append(absolute_error)
```

[54]:
```python
errors_3
```

[54]:
```
[0.003188813375989419,
 3.1653391875607895e-05,
 3.136965922578838e-07,
 1.2895548096025777e-07,
 4.842268497284863e-07,
 0.0011373526040650006,
 0.12548233136208164,
 5.099281481682784,
 5.099281481682784,
 5.099281481682784,
 5.099281481682784]
```

[55]:
```python
plt.figure(figsize=(10, 6))
plt.style.use('seaborn-v0_8-deep')
plt.loglog(h_values, errors_3, marker='o', label='central difference second␣
 ↪derivative')
plt.xlabel('h')
plt.ylabel('Absolute Error')
plt.title('Absolute Error vs h (Double Logarithmic Plot)')
plt.legend()
plt.grid(True, which="both", ls="--")
plt.show()
```



Conclusion

In this analysis, we computed the first and second derivatives of the function $f(x) = e^{\sin(2x)}$ at $x = 0.5$ using the two-point central difference method and compared the results with the two-point forward difference method. The absolute errors were plotted as a function of h on a double logarithmic plot.

For the first derivative, the central difference method showed a lower absolute error compared to the forward difference method, especially for smaller values of h.

Then we can compare it with the central difference second derivative method, and the results shows that it has a higher absolute errors then first derivatives.