Numerical Methods
Lab 4.Numerical Differentiation

Maryna Borovyk

October 23, 2024

1 Introduction

This report analyzes the numerical and analytical derivatives of two functions: $f1(x)$ and $f2(x)$ at the point $x = 0.5$. The absolute errors are computed and plotted on a double logarithmic scale.

2 Methodology

The functions analyzed are:

$$f1(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$
$$f2(x) = e^{\sin(2x)}$$

The numerical derivative is computed using the finite difference method:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

The analytical derivatives are computed using symbolic differentiation.

3 Code

```python
import numpy as np
from sympy import Symbol, lambdify, exp, sin
import matplotlib.pyplot as plt

3 usages
def derivative(f, x, h):
    f_derivative = (f(x+h) - f(x)) / h
    return f_derivative

# function 1
def f1(x):
    return -0.1 * x**4 - 0.15 * x**3 - 0.5 * x**2 - 0.25 * x + 1.2

# function 2
1 usage
def f2(x):
    return np.exp(np.sin(2 * x))
```

```python
1 usage
def analytical_derivative_f1(x_value):
    x = Symbol('x')
    f = -0.1 * x**4 - 0.15 * x**3 - 0.5 * x**2 - 0.25 * x + 1.2
    f_derivative = f.diff(x)
    f_derivative_func = lambdify(x, f_derivative)
    return f_derivative_func(x_value)

1 usage
def analytical_derivative_f2(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)
```

```python
x = 0.5
n_values = range(1, 12)
h_values = [10**-n for n in n_values]
derivatives = [derivative(f1, x, h) for h in h_values]
```

```python
# Calculate derivatives and errors for f1
derivatives_f1 = [derivative(f1, x, h) for h in h_values]
analytical_value_f1 = analytical_derivative_f1(x)
absolute_errors_f1 = [abs(d - analytical_value_f1) for d in derivatives_f1]

# Calculate derivatives and errors for f2
derivatives_f2 = [derivative(f2, x, h) for h in h_values]
analytical_value_f2 = analytical_derivative_f2(x)
absolute_errors_f2 = [abs(d - analytical_value_f2) for d in derivatives_f2]
```

```python
print("Results for f1(x):")
print(f"Analytical value: {analytical_value_f1}")
for n, h, d, error in zip(n_values, h_values, derivatives_f1, absolute_errors_f1):
    print(f"h = 10^-{n}: derivative = {d}, absolute error = {error}")

# Print results for f2
print("\nResults for f2(x):")
print(f"Analytical value: {analytical_value_f2}")
for n, h, d, error in zip(n_values, h_values, derivatives_f2, absolute_errors_f2):
    print(f"h = 10^-{n}: derivative = {d}, absolute error = {error}")
```

```
# Plot results for f1
plt.figure(figsize=(10, 6))
plt.style.use('seaborn-v0_8-deep')
plt.loglog( *args: h_values, absolute_errors_f1, marker='o', label='f1(x)')
plt.loglog( *args: h_values, absolute_errors_f2, marker='x', label='f2(x)')
plt.xlabel('h')
plt.ylabel('Absolute Error')
plt.title('Absolute Error vs h (Double Logarithmic Plot)')
plt.legend()
plt.grid( visible: True, which="both", ls="--")
plt.show()
```

4 Results

```
Results for f1(x):
Analytical value: -0.9125
h = 10^-1: derivative = -1.003599999999999, absolute error = 0.09109999999999896
h = 10^-2: derivative = -0.9212850999999911, absolute error = 0.008785099999991108
h = 10^-3: derivative = -0.9133753500999431, absolute error = 0.0008753500999431063
h = 10^-4: derivative = -0.9125875034998732, absolute error = 8.750349987318717e-05
h = 10^-5: derivative = -0.9125087500283512, absolute error = 8.75002835121208e-06
h = 10^-6: derivative = -0.9125008749721886, absolute error = 8.749721885914497e-07
h = 10^-7: derivative = -0.9125000866028188, absolute error = 8.660281880512599e-08
h = 10^-8: derivative = -0.9125000088872071, absolute error = 8.887207081365034e-09
h = 10^-9: derivative = -0.912499964478286, absolute error = 3.552171401466353e-08
h = 10^-10: derivative = -0.9125000755005885, absolute error = 7.550058855887443e-08
h = 10^-11: derivative = -0.9125034061696624, absolute error = 3.406169662434344e-06
```

Figure 1: Results for f1(x) = -0.1 $x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$

```
Results for f2(x):
Analytical value: 2.506761534986894
h = 10^-1: derivative = 2.199057076649269, absolute error = 0.30770445833762494
h = 10^-2: derivative = 2.4807256192967753, absolute error = 0.02603591569011865
h = 10^-3: derivative = 2.5042064928371133, absolute error = 0.0025550421497806397
h = 10^-4: derivative = 2.5065065168927703, absolute error = 0.00025501809412364906
h = 10^-5: derivative = 2.506736038032642, absolute error = 2.549695425191345e-05
h = 10^-6: derivative = 2.506758985720836, absolute error = 2.549266057805255e-06
h = 10^-7: derivative = 2.5067612785534266, absolute error = 2.5643346734938177e-07
h = 10^-8: derivative = 2.5067615094798157, absolute error = 2.550707822734921e-08
h = 10^-9: derivative = 2.5067614650708947, absolute error = 6.991599921235547e-08
h = 10^-10: derivative = 2.506763685516944, absolute error = 2.1505300500379576e-06
h = 10^-11: derivative = 2.5067947717616335, absolute error = 3.323677473954234e-05
```

Figure 2: Results for f2(x) = $e^{\sin(2x)}$

The following plots show the absolute errors for different values of h on a double logarithmic scale.

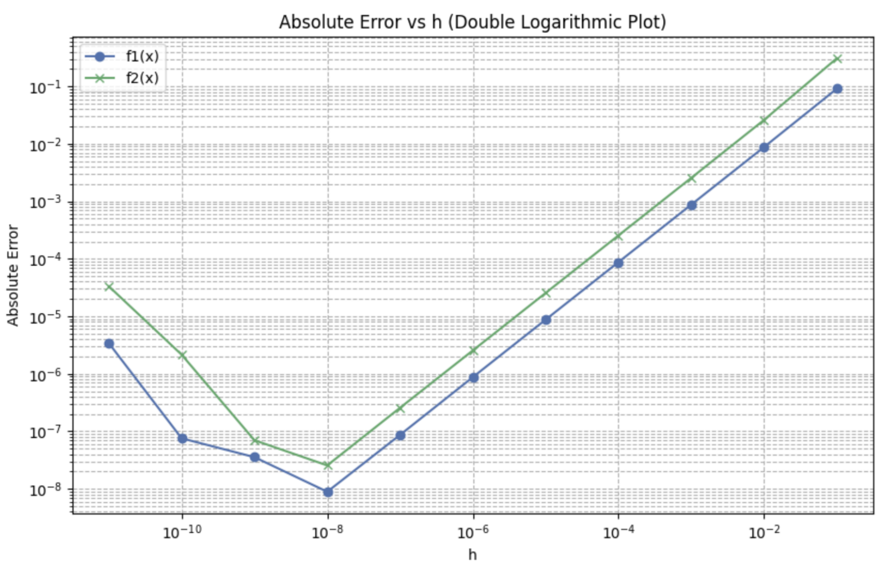

Figure 3: Absolute Error vs h (Double Logarithmic Plot)

5 Conclusion

The analysis shows the behavior of the absolute error as a function of h for both functions. The results indicate the accuracy of the numerical derivative compared to the analytical derivative.