Computional Homework1 Report

Student: 纪浩正, jihz2023@mail.sustech.edu.cn

Combined Rounding and Truncation Errors for the Derivative of e^{5x}

0.1 Rounding Error

Machine epsilon ε represents the precision limit of floating-point arithmetic, with $\varepsilon \approx 2.2204 \times 10^{-16}$ for `float64`.

The rounding error introduced when representing x in floating-point arithmetic is:

$$|\text{round}(x) - x| \leq 0.5 \cdot \varepsilon \cdot 2^E$$

For the forward finite difference approximation of the derivative, the rounding error can be bounded by:

$$e\left(\frac{f(x+h) - f(x)}{h}\right) \leq \frac{\varepsilon|f(x+h)| + \varepsilon|f(x)|}{h} \approx \frac{\varepsilon|f(x)|}{h}$$

0.2 Truncation Error

By Taylor's theorem:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(\theta), \quad \theta \in [x, x+h]$$

The local truncation error for the forward difference formula is:

$$e\left(\frac{f(x+h) - f(x)}{h}\right) < M \cdot \frac{h}{2}, \quad M = \max_{[x,x+h]} |f''(\theta)|$$

0.3 Total Error

Thus, the total error is the sum of rounding and truncation errors:

$$\text{error} = \text{rounding error} + \text{truncation error} = \frac{\varepsilon|f(x)|}{h} + \frac{f''(x)}{2}h$$

0.4 Numerical Implementation

Calculate $f(x_0)$ at $x_0 = 1$ numerically and analytically:

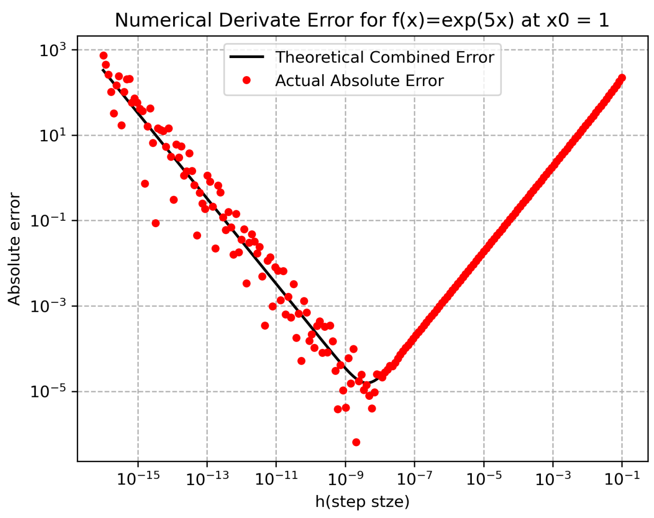
```python
import numpy as np
import matplotlib.pyplot as plt
import os
os.makedirs('images', exist_ok = True)

def f(x):
    return np.exp(5*x)
def f1(x):
    return 5 * np.exp(5*x)
def f2(x):
    return 25 * np.exp(5*x)

x0 = 1
h = np.logspace(-16, -1, 200)
f_approx = (f(x0 + h) - f(x0)) / h
f_real = f1(x0)
```

0.5 Computational Errors

- The minimum theoretical combined error eC = 1.564e-05 occurs at h = 4.103e-09
- The minimum computational absolute error abs_err = 6.499e-07 occurs at h = 2.049e-09

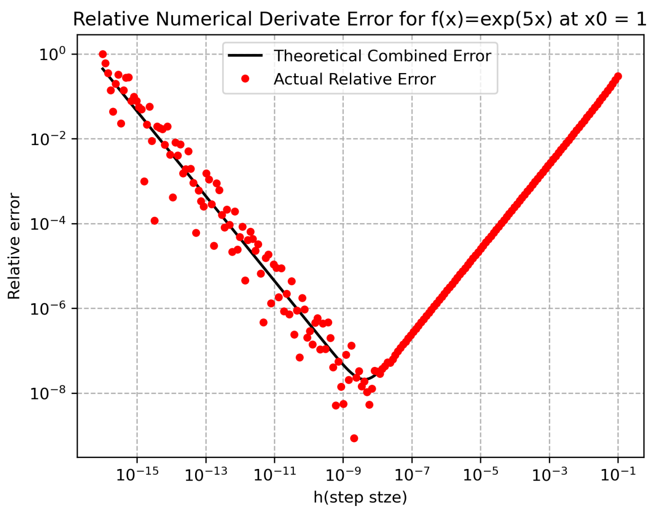
```python
E_err = np.finfo(float).eps * f(x0) / h # rounding error
T_err = 0.5 * f2(x0) * h              # truncation error
eC = E_err + T_err                    # theoretical error
rel_eC = eC / abs(f_real)             # relative theoretical error
abs_err = np.abs(f_approx - f_real)  # absolute calculation error
rel_err = abs_err / abs(f_real)      # relative calculation error

# Theoretical error minimum
idx_min = np.argmin(eC)
h_min = h[idx_min]
print(f"The minimum theoretical combined error eC = {eC[idx_min]:.3e} occurs at h =
    {h_min:.3e}")
# Actual (computed) error minimum
idx_abs = np.argmin(abs_err)
h_min_abs = h[idx_abs]
print(f"The minimum computational absolute error abs_err = {abs_err[idx_abs]:.3e} occurs at h
    = {h_min_abs:.3e}")
```

0.6 Visualization: Absolute Error



```
plt.figure()
plt.loglog(h, eC, 'k-', linewidth=1.7, label='Theoretical Combined Error')
plt.loglog(h, abs_err, 'ro', markersize=4, markerfacecolor='r', label='Actual Absolute Error')
plt.xlabel('h (step size)')
plt.ylabel('Absolute error')
plt.title('Numerical Derivative Error for $f(x)=\exp(5x)$ at $x_0=1$')
plt.legend()
plt.grid(True, which='both', ls="--")
plt.savefig('images/Numerical Derivative Error.png', dpi=300, bbox_inches='tight')
plt.show()
```

0.7 Visualization: Relative Error

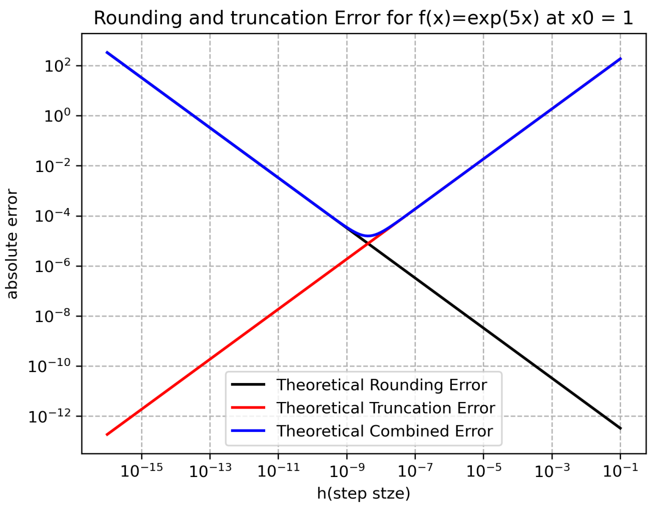


```
plt.figure()
plt.loglog(h, rel_eC, 'k-', linewidth=1.7, label='Theoretical Relative Error')
plt.loglog(h, rel_err, 'ro', markersize=4, markerfacecolor='r', label='Actual Relative Error')
plt.xlabel('h (step size)')
plt.ylabel('Relative error')
plt.title('Relative Numerical Derivative Error for $f(x)=\exp(5x)$ at $x_0=1$')
plt.legend()
plt.grid(True, which='both', ls="--")
plt.savefig('images/Relative Numerical Derivative Error.png', dpi=300, bbox_inches='tight')
plt.show()
```

0.8 Visualization: Theoretical Rounding and Truncation Errors

$$\text{Total Error} = \frac{\varepsilon|f(x)|}{h} + \frac{f''(x_0)}{2}h$$

As can be seen, for small h, the rounding error dominates; for large h, the truncation error dominates. When h is very small, the actual computed error exhibits strong oscillations due to loss of significance in floating-point computation. Beyond the minimum error point, the error curve becomes smoother because truncation error increases regularly.



```
plt.figure()
plt.loglog(h, E_err, 'k-', linewidth=1.7, label='Theoretical Rounding Error')
plt.loglog(h, T_err, 'ro', markersize=4, markerfacecolor='r', label='Theoretical Truncation
    Error')
plt.loglog(h, eC, 'b-', linewidth=1.7, label='Theoretical Combined Error')
plt.xlabel('h (step size)')
plt.ylabel('Absolute error')
plt.title('Rounding and Truncation Error for $f(x)=\exp(5x)$ at $x_0=1$')
plt.legend()
plt.grid(True, which='both', ls="--")
plt.savefig('images/Rounding and truncation Error.png', dpi=300, bbox_inches='tight')
plt.show()
```

0.9 Discussion

The results clearly show that the total error in the numerical derivative is governed by different sources depending on the step size:

- **For very small** h: Rounding error dominates and the actual (computed) error exhibits strong fluctuations, reflecting numerical instability due to loss of significance.
- **For larger** h: Truncation error dominates and both theoretical and actual error curves are smooth and increase with h.
- There exists an optimal h near the bottom of the U-shaped error curve, where total error is minimized.

This demonstrates the importance of balancing rounding and truncation errors in numerical differentiation, and confirms theoretical predictions with actual numerical experiments.