Chapter 4

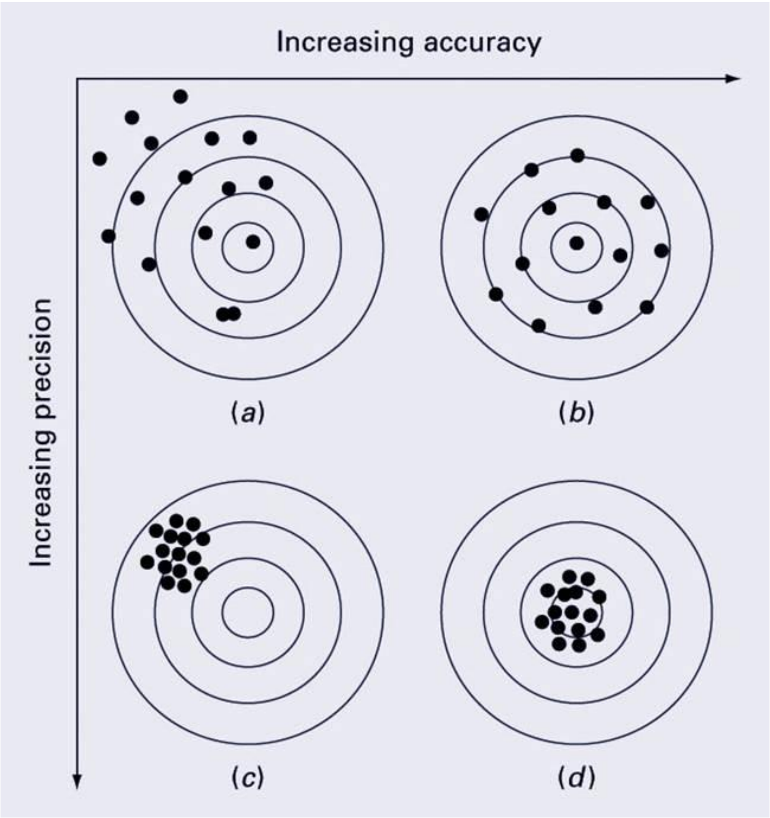
Roundoff and Truncation Errors

Numerical Methods
Fall 2019

Accuracy and Precision

- *Accuracy* refers to how closely a computed or measured value agrees with the true value, while *precision* refers to how closely individual computed or measured values agree with each other.

a) inaccurate and imprecise
b) accurate and imprecise
c) inaccurate and precise
d) accurate and precise



Error Definitions (1)

- True error (E_t): the difference between the true value and the approximation.
  - Absolute error ($|E_t|$): the absolute difference between the true value and the approximation.

$$E_t = \text{true value} - \text{approximation}$$

- True fractional relative error: the true error divided by the true value.
- Relative error (ε_t): the true fractional relative error expressed as a percentage.

$$\varepsilon_t = \frac{\text{true value} - \text{approximation}}{\text{true value}} 100\%$$

Error Definitions (2)

- The approximate percent relative error can be given as the approximate error divided by the approximation, expressed as a percentage

$$\varepsilon_a = \frac{\text{approximate error}}{\text{approximation}} 100\%$$

- For iterative processes, the error can be approximated as the difference in values between successive iterations.

$$\varepsilon_a = \frac{\text{present approximation} - \text{previous approximation}}{\text{present approximation}} 100\%$$

Using Error Estimates

- Often, when performing calculations, we may not be concerned with the sign of the error but are interested in whether the absolute value of the percent relative error is lower than a pre-specified tolerance ε_s. For such cases, the computation is repeated until

$$|\varepsilon_a| < \varepsilon_s$$

- This relationship is referred to as a *stopping criterion*.

Using Error Estimates

- Example 4.1

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

  - Let's estimate $e^{0.5}$

$$e^{0.5} = 1.648721\ldots.$$

True value

$$e^x = 1 + x$$

or for $x = 0.5$

$$e^{0.5} = 1 + 0.5 = 1.5$$

$$\varepsilon_t = \left|\frac{1.648721 - 1.5}{1.648721}\right| \times 100\% = 9.02\%$$

$$\varepsilon_a = \left|\frac{1.5 - 1}{1.5}\right| \times 100\% = 33.3\%$$

Terms	Result	ε_t, %	ε_a, %
1	1	39.3	
2	1.5	9.02	33.3
3	1.625	1.44	7.69
4	1.645833333	0.175	1.27
5	1.648437500	0.0172	0.158
6	1.648697917	0.00142	0.0158

Computer solution of Ex. 4.1

```
function [fx,ea,iter] = IterMeth(x,es,maxit)
% Maclaurin series of exponential function
%   [fx,ea,iter] = IterMeth(x,es,maxit)
% input:
%   x = value at which series evaluated
%   es = stopping criterion (default = 0.0001)
%   maxit = maximum iterations (default = 50)
% output:
%   fx = estimated value
%   ea = approximate relative error (%)
%   iter = number of iterations

% defaults:
if nargin<2|isempty(es),es=0.0001;end
if nargin<3|isempty(maxit),maxit=50;end
% initialization
iter = 1; sol = 1; ea = 100;
% iterative calculation
while (1)
  solold = sol;
  sol = sol + x ^ iter / factorial(iter);
  iter = iter + 1;
  if sol~=0
    ea=abs((sol - solold)/sol)*100;
  end
  if ea<=es | iter>=maxit,break,end
end
fx = sol;
end
```

Roundoff Errors

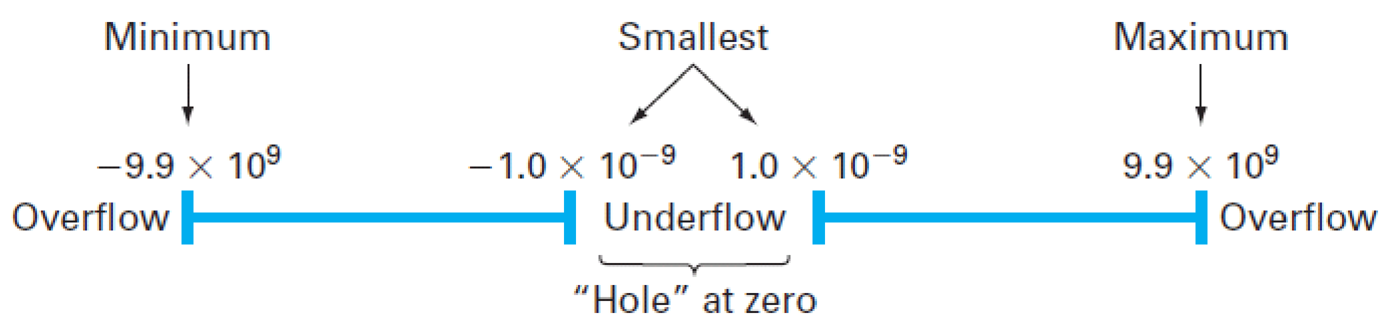
- *Roundoff errors* arise because digital computers cannot represent some quantities exactly. There are two major facets of roundoff errors involved in numerical calculations:
  - Digital computers have size and precision limits on their ability to represent numbers.
  - Certain numerical manipulations are highly sensitive to roundoff errors.

Floating-Point Representation

- Example 4.2
  - Suppose a base-10 computer with a 5-digit word size, one digit is used for the sign, two for the exponent, and two for the mantissa.

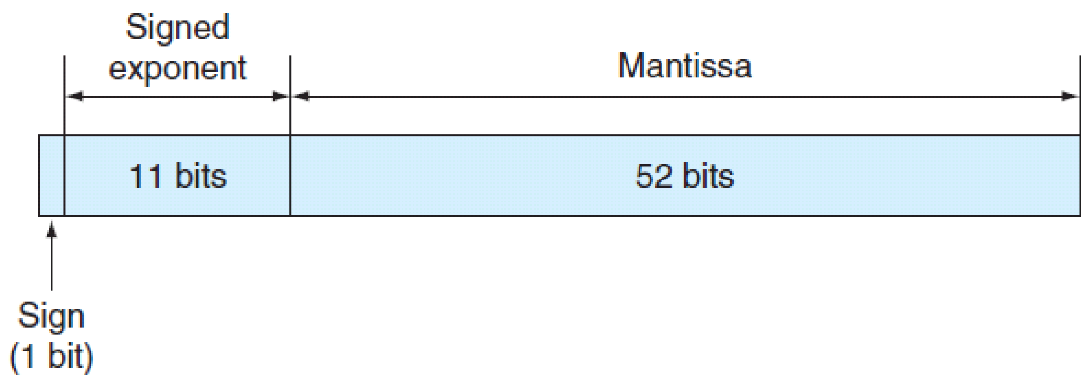
$$s_1 d_1 . d_2 \times 10^{s_0 d_0}$$

  - Largest value: $+9.9 \times 10^{+9}$
  - Smallest value: $+1.0 \times 10^{-9}$



Computer Number Representation

- By default, MATLAB has adopted the IEEE double-precision format in which eight bytes (64 bits) are used to represent floating-point numbers:
  $n = \pm(1+f) \times 2^{e}$



  - The sign is determined by a sign bit
  - The mantissa f is determined by a 52-bit binary number
  - The exponent e is determined by an 11-bit binary number, from which 1023 is subtracted to get e

Floating Point Ranges

−1023, −1024는 특수
목적으로 남겨둠

- The exponent range is −1022 to 1023.
- The largest possible number MATLAB can store has
  - *f* of all 1's, giving a significand (mantissa) of $2 - 2^{-52}$, or approximately 2

$$+1.1111\ldots1111 \times 2^{+1023}$$

  - This yields approximately $2^{1024} = 1.7997 \times 10^{308}$

- The smallest possible number MATLAB can store with full precision has
  - *f* of all 0's, giving a mantissa of 1

$$+1.0000\ldots0000 \times 2^{-1022}$$

  - This yields $2^{-1022} = 2.2251 \times 10^{-308}$

Floating Point Precision

- The 52 bits for the mantissa f correspond to about 15 to 16 base-10 digits.
- The machine epsilon (machine precision) – the maximum relative error between a number
- MATLAB's representation of that number, is thus

$$2^{-52} = 2.2204 \times 10^{-16}$$

Roundoff Errors with Arithmetic Manipulations

- Roundoff error can happen in several circumstances other than just storing numbers – for example:
  - *Large computations* – if a process performs a large number of computations, roundoff errors may build up to become significant

```
function sout = sumdemo()
s = 0;
for i = 1:10000
  s = s + 0.0001;
end
sout = s;
```

컴퓨터는 0.1, 0.01.. 등을 정확히 표현하지 못함

When this function is executed, the result is

```
>> format long
>> sumdemo

ans =
    0.99999999999991
```

Roundoff Errors with Arithmetic Manipulations

- *Adding a Large and a Small Number* – Since the small number's mantissa is shifted to the right to be the same scale as the large number, digits are lost
  - Suppose we add a small number, 0.0010, to a large number, 4000, using a hypothetical computer with the 4-digit mantissa and the 1-digit exponent

$$\begin{array}{r}
0.4000 \times 10^4 \\
0.0000001 \times 10^4 \\
\hline
0.4000001 \times 10^4
\end{array} \Rightarrow 0.4000 \times 10^4$$

Truncation Errors

- *Truncation errors* are those that result from using an approximation in place of an exact mathematical procedure.
- Example 1: approximation to a derivative using a finite-difference equation:

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - ti}$$

- Example 2: The Taylor Series

The Taylor Theorem and Series

- The *Taylor theorem* states that any smooth function can be approximated as a polynomial.
- The *Taylor series* provides a means to express this idea mathematically.

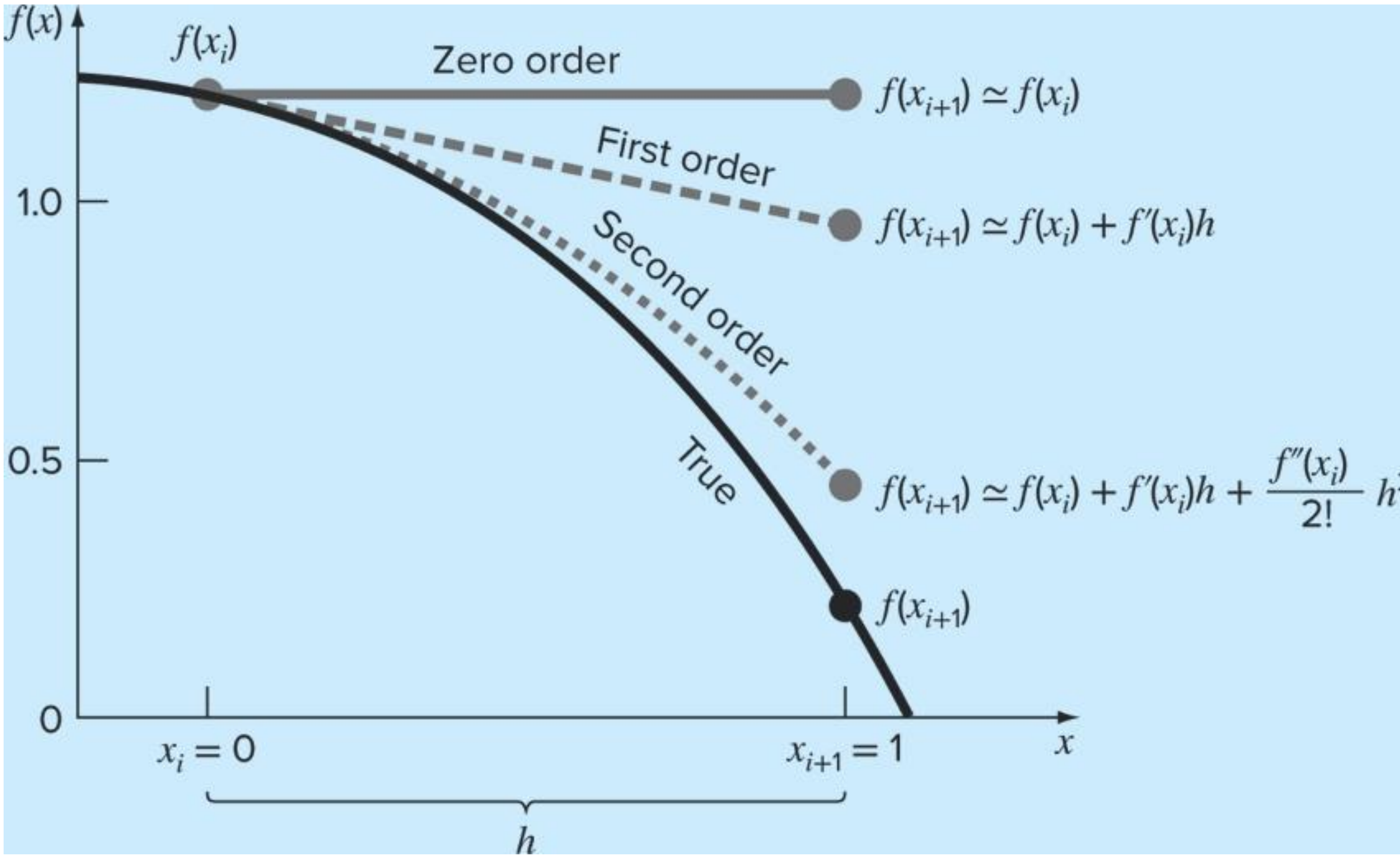
$$f(x_{i+1})$$
$$= f(x_i) + f'(x_i)h + \frac{f''(xi)}{2!}h^2 + \frac{f^{(3)}\,(xi)}{3!}h^3 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + Rn$$

R_1

R_2

R_3

The Taylor Series



The approximation of $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$ at $x = 1$

Truncation Error

- In general, the n–th order Taylor series expansion will be exact for an n–th order polynomial.
- In other cases, the remainder term R_n is of the order of h^{n+1}, meaning:
  - The more terms are used, the smaller the error, and
  - The smaller the spacing, the smaller the error for a given number of terms.

Truncation Error

- Example 4.3: Use Taylor series expansions with $n = 0$ to 6 to approximate $f(x) = \cos(x)$ at $x_{i+1} = \pi/3$ on the basis of the value of $f(x)$ and its derivatives at $x_i = \pi/4$.
  - Zero order approximation: $f\left(\frac{\pi}{3}\right) \cong \cos\left(\frac{\pi}{4}\right) = 0.707106781$

$$\varepsilon_t = \left|\frac{0.5 - 0.707106781}{0.5}\right| 100\% = 41.4\%$$

  - First-order approximation:

$$f\left(\frac{\pi}{3}\right) \cong \cos\left(\frac{\pi}{4}\right) - \sin\left(\frac{\pi}{4}\right)\left(\frac{\pi}{12}\right) = 0.521986659$$

  - Second-order approximation

$$f\left(\frac{\pi}{3}\right) \cong \cos\left(\frac{\pi}{4}\right) - \sin\left(\frac{\pi}{4}\right)\left(\frac{\pi}{12}\right) - \frac{\cos(\pi/4)}{2}\left(\frac{\pi}{12}\right)^2 = 0.497754491$$

Truncation Error

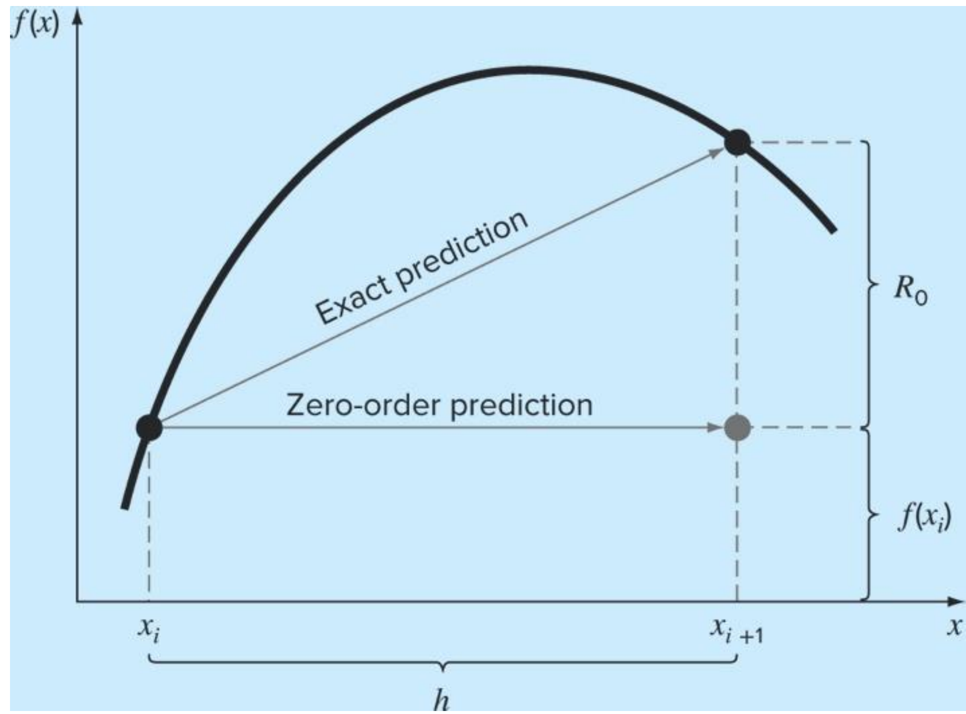
Order n	$f^{(n)}(x)$	$f(\pi/3)$	$\lvert\varepsilon_t\rvert$
0	$\cos x$	0.707106781	41.4
1	$-\sin x$	0.521986659	4.40
2	$-\cos x$	0.497754491	0.449
3	$\sin x$	0.499869147	2.62×10^{-2}
4	$\cos x$	0.500007551	1.51×10^{-3}
5	$-\sin x$	0.500000304	6.08×10^{-5}
6	$-\cos x$	0.499999988	2.44×10^{-6}

Remainder for the Taylor Series Expansion

- Suppose that we truncated the Taylor series expansion [Eq. (4.13)] after the zero-order term to yield

$$f(x_{i+1}) \cong f(x_i)$$

- The remainder (or error) of this prediction, which consists of the infinite series of terms that were truncated



$$R_0 = f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots$$

Numerical Differentiation (1)

- The first order Taylor series can be used to calculate approximations to derivatives:
  - Given: $f(x_{i+1}) = f(x_i) + f'(x_i)h + O(h^2)$
  - Then: $f'(x_i) = \frac{f(x_{i+1}) - f(xi)}{h} - O(h)$

Truncation error

$$\boxed{R_n = O(h^{n+1})}$$

- This is termed a "forward" difference because it utilizes data at *i* and *i*+1 to estimate the derivative.

$$\boxed{\frac{R_1}{t_{i+1} - t_i} = O(t_{i+1} - t_i)}$$

Numerical Differentiation (2)

- There are also backward and centered difference approximations, depending on the points used:
- *Finite difference approximation*
  - Forward:

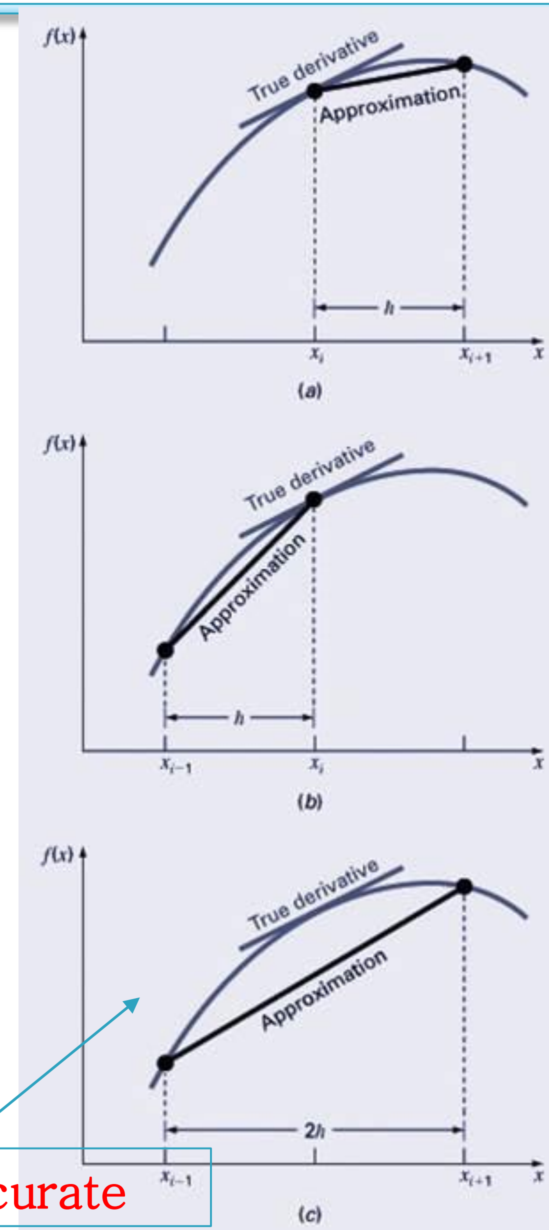
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - O(h)$$

  - Backward:

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} - O(h)$$

  - Centered:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2)$$



more accurate

Numerical Differentiation (3)

- Backward Difference Approximation of the First Derivative
  - The Taylor series can be expanded backward to calculate a previous value on the basis of a present value

$$f(x_{i-1}) = f(x_i) - f'(x_i)h - \frac{f''(x_i)}{2!}h^2 - \cdots \qquad (4.22)$$

  - Truncating this equation after the first derivative and rearranging yields

$$f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{h}$$

  - where the error is $O(h)$.

Numerical Differentiation (4)

- Centered Difference Approximation of the First Derivative
  - subtract Eq. (4.22) from the forward Taylor series expansion

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \cdots$$

to yield

$$f(x_{i+1}) = f(x_{i-1}) + 2f'(x_i)h + 2\frac{f^{(3)}(x_i)}{3!}h^3 + \cdots$$

which can be solved for

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f^{(3)}(x_i)}{6}h^2 + \cdots$$

more accurate representation of the derivative

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2)$$

Truncation error

Finite-Difference Approximations of Derivatives

- Example 4.4: Use forward and backward difference approximations of $O(h)$ and a centered difference approximation of $O(h^2)$ to estimate the first derivative of

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

$$\boxed{\text{true value as } f'(0.5) = -0.9125.}$$

- For h=0.5

Solution. For $h = 0.5$, the function can be employed to determine

$$\begin{aligned} x_{i-1} &= 0 & f(x_{i-1}) &= 1.2 \\ x_i &= 0.5 & f(x_i) &= 0.925 \\ x_{i+1} &= 1.0 & f(x_{i+1}) &= 0.2 \end{aligned}$$

These values can be used to compute the forward difference [Eq. (4.21)],

$$f'(0.5) \cong \frac{0.2 - 0.925}{0.5} = -1.45 \qquad |\varepsilon_t| = 58.9\%$$

the backward difference [Eq. (4.23)],

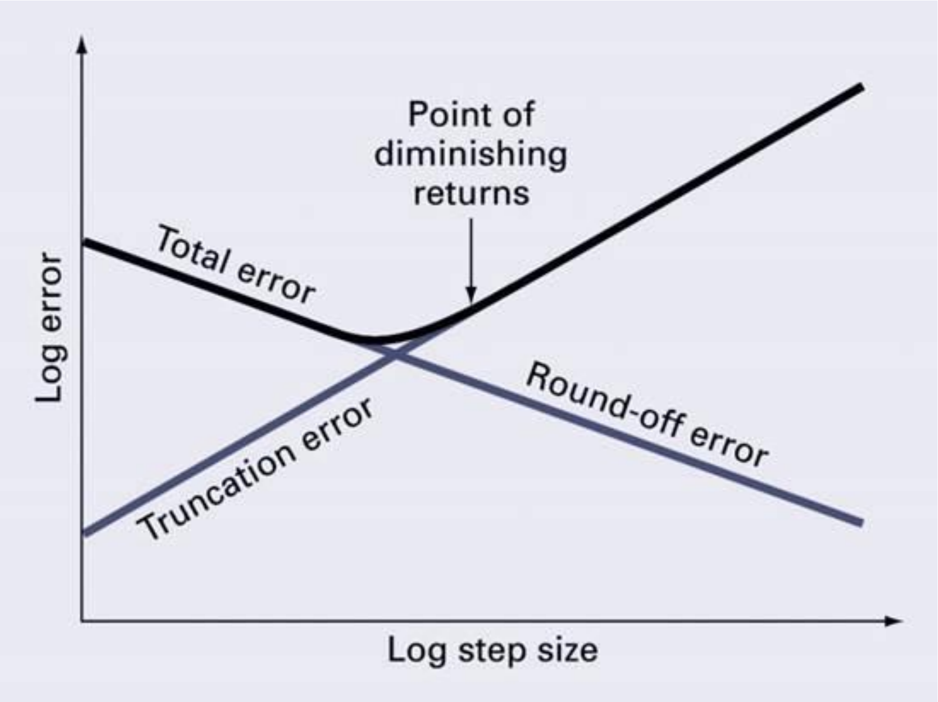
$$f'(0.5) \cong \frac{0.925 - 1.2}{0.5} = -0.55 \qquad |\varepsilon_t| = 39.7\%$$

and the centered difference [Eq. (4.25)],

$$f'(0.5) \cong \frac{0.2 - 1.2}{1.0} = -1.0 \qquad |\varepsilon_t| = 9.6\%$$

Total Numerical Error

- The *total numerical error* is the summation of the truncation and roundoff errors.
- The truncation error generally *increases* as the step size increases, while the roundoff error *decreases* as the step size increases – this leads to a point of diminishing returns for step size.



Total Numerical Error

- Example 4.5 : we used a centered difference approximation of $O(h^2)$ to estimate the first derivative of the following function at $x = 0.5$

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

```
>> ff=@(x) -0.1*x^4-0.15*x^3-0.5*x^2-0.25*x+1.2;
>> df=@(x) -0.4*x^3-0.45*x^2-x-0.25;
>> diffex(ff,df,0.5,11)

  step size    finite difference     true error
 1.0000000000 -1.2625000000000  0.3500000000000
 0.1000000000 -0.9160000000000  0.0035000000000
 0.0100000000 -0.9125350000000  0.0000350000000
 0.0010000000 -0.9125003500001  0.0000003500000
 0.0001000000 -0.9125000349985  0.0000000034998
 0.0000100000 -0.9125000003318  0.0000000000332
 0.0000010000 -0.9125000000542  0.0000000000054
 0.0000001000 -0.9124999945031  0.0000000005497
 0.0000000100 -0.9125000333609  0.0000000033361
 0.0000000010 -0.9125001998944  0.0000000199894
 0.0000000001 -0.9125007550059  0.0000000755006
```

Total Numerical Error

- An optimal step size

$$h_{opt} = \sqrt[3]{\frac{3\varepsilon}{M}}$$

M is a maximum absolute value of the third derivative of f(x)

- In this example,

$$M = \left|f^{(3)}(0.5)\right| = |-2.4(0.5) - 0.9| = 2.1$$

- MATLAB's roundoff error is about $\epsilon = 0.5 \times 10^{-16}$.

$$h_{opt} = \sqrt[3]{\frac{3(0.5 \times 10^{-16})}{2.1}} = 4.3 \times 10^{-6}$$

Other Errors

- Blunders – errors caused by malfunctions of the computer or human imperfection.
- Model errors – errors resulting from incomplete mathematical models.
- Data uncertainty – errors resulting from the accuracy and/or precision of the data.