# Introduction to Computational Thinking and Programming for CFD

## Module 13251

Dr. rer. nat. Marten Klein
Numerical Fluid and Gas Dynamics
BTU Cottbus - Senftenberg

Brandenburg
University of Technology
Cottbus - Senftenberg

# 4 Elementary Numerical Methods: Differentiation

## 4.1 Preliminary remarks on numerical errors

# Number representation

- For CFD we need **numeric data types** i.e. int, float, double.
- Any real number 'x' can be represented as a **binary number sequence**

$$x = \pm ( a_n . 2^n + a_{n-1} . 2^{n-1} + \ldots + a_0 . 2^0 + a_{-1} . 2^{-1} + \ldots)$$

where each $a_i$ is either 0 or 1, i.e. $a_i \in \{0, 1\} \; \forall i$ .

# Integer (whole) numbers

- For **whole numbers** (integers: type int) one usually uses 32 bits (4 bytes) int, of which 1 bit is reserved for the sign. With this one can represent all numbers between $-2^{31}$ and $2^{31} - 1 \approx 2 \cdot 10^9$.

- **Arithmetic operations** are always exact for integers.

- However, **overflows** can occur due to too large/small results.

# Fixed point numbers

- Alternative concepts are **fixed point numbers**, where $n_1$ binary digits are stored <u>before</u> the decimal point and $n_2$ digits <u>after</u> the decimal point.
- Again, the arithmetics is exact.
- Such numbers are suitable for accounting calculations but are too inflexible for the requirements of CFD.

# Floating point numbers

- In most cases, we use **floating point numbers** (*floats*: type float & double). These are commonly stored as **64-bit (8 bytes) double precission** numbers.

- Each floating-point number *x* is divided into three sections

$$x = (-1)^{\mathrm{Sign}} \cdot \mathrm{Mantissa} \cdot 2^{\mathrm{Exponent}}$$

- The size (memory allocation) for any such number is specified in the **IEEE-754 standard**:

  - The <u>sign</u> needs 1 bit in memory.

  - 52 bits are reserved for the <u>mantissa</u>. This means that there are only about **15-16 significant digits** ($2^{52} \approx 5 \times 10^{15}$). Further digits are cut off, which leads to **rounding errors**.

  - A floating point number is said to be normalized if the first digit of the mantissa is not zero.

  - The <u>exponent</u> is in the last 11 bits. Because $2^{11} = 2048$, the exponent is restricted to the range between −1022 and 1024. Because $2^{1024} \approx 10^{308}$, the representable numbers are between $10^{-308}$ and $10^{308}$ surplus the corresponding negative values and zero.

# Rounding error: annihilation

- The arithmetics of floating point numbers is not exact due to the limited length of the mantissa and carry-over to the exponent upon mathematical operations.

- For example, **numbers that are too small disappear when added**

$$1 + x = 1 \text{ for } 0 \le x < 2.22 \cdot 10^{-16}$$

- Such inaccuracies lead to astonishing results that must be taken into account when creating algorithms.

Example: $(1 + x_1) - (1 + x_2) \neq= x_1 - x_2$ for small $x_1, x_2 << 1$

# Rounding error: cast type

- Another inaccuracy comes from **converting from decimal inputs**:

  Example: $x = 0.6$

- The representation, i.e., the output to 16 digits with *print(‘%1.16f’ % x)* gives:

  0.599 999 999 999 999 98

- The missing portion of $2 \cdot 10^{-17}$ must be cut off when converting 0.6 to a floating point double!

- A practical problem arises when an interval [*a*, *b*] is to be divided into *N* subintervals (→ exercise: grid generation). The range limit *b* has a rounding error after *numpy.linspace(a, b, N)* was called.

# Summary of numerical errors

1. Rounding errors are unavoidable with **floating point numbers**. However, suitable data types help to keep the error small.

   Example: *double* instead of *single* precision floats

2. **Discretization errors** result from the numerical approximation of differential equations or numerical integration. These usually dominate and must be quantified and controlled.

   Example: *suitable grids*

3. There are also **model errors**. First of all, these have nothing to do with numerics but are due to the mathematical or physical simplification of a problem. For example:

   - Assumption of an incompressible flow
   - Using a reduced chemical mechanism in a reactive flow

# 4 Elementary Numerical Methods: Differentiation

## 4.2 Interlude/Reminder: Structured grids

# Structured Grids

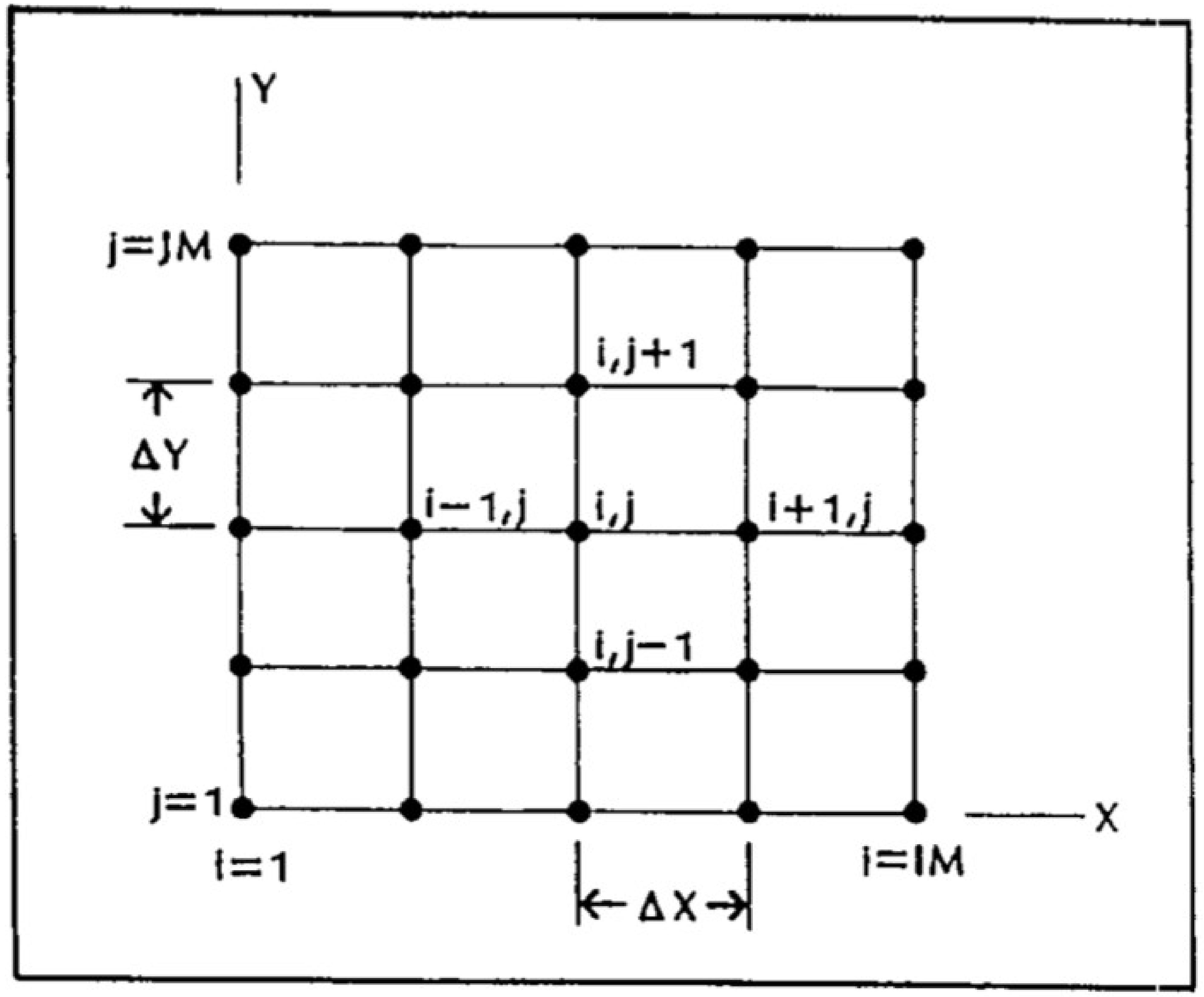



Figure 1-6. Sketch illustrating the nomenclature of computational space.

Aus: K.A. Hoffmann & S.T. Chiang, *Computational Fluid Dynamics: Volume I*, Engineering Education System, 2000.

# 4 Elementary numerical methods: Differentiation

## 4.3 Discretization of the derivative

# Problem and simplification

- The following derivations occur in the basic equations:
  - temporal: $\partial\rho /\partial t$
  - spatial: $\nabla \cdot \mathrm{u} = \frac{\partial u_i}{\partial x_i}$ oder $\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}$
- These terms need to be evaluated numerically in CFD.
- The **basic problem** can be formulated as follows:
  - A function $f(x)$ is known.
  - For this function, we seek the derivative $f'(x)$.
- $f(x)$ is usually <u>not</u> known in detail. Instead, only discrete nodal values $y_i = f(x_i)$ are given at the nodal points $x_i$ of the grid.
- So we are looking for ways to calculate $f'(x)$ **approximately** using the given nodal values (array elements) $y_i$ and grid points (vertex coordinate values) $x_i$.

# Discretization: grid and nodal values I

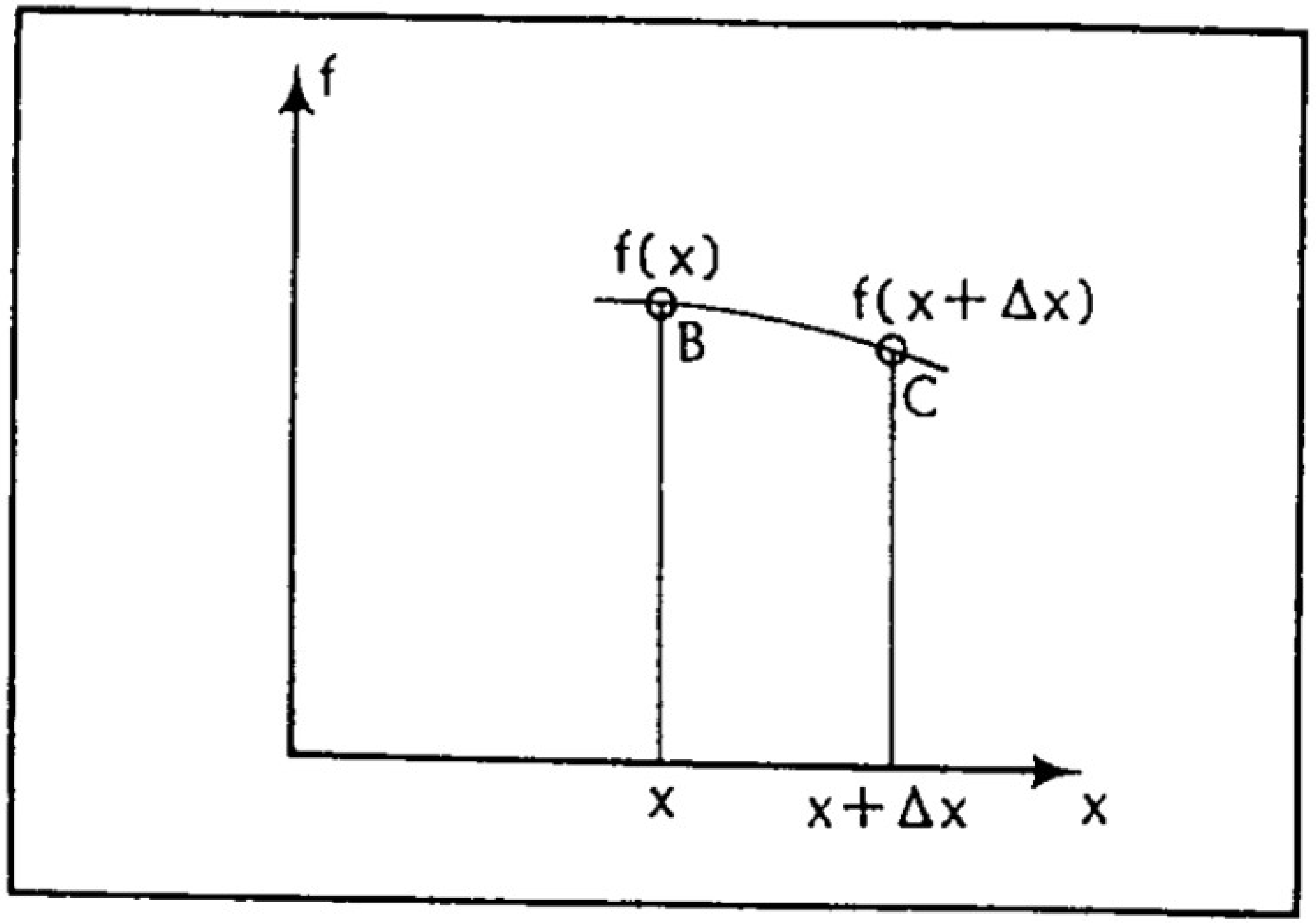



Figure 2-1. Illustration of grid points used in Equation (2-3).

**From: K.A. Hoffmann & S.T. Chiang, Computational Fluid Dynamics: Volume I, Engineering Education System, 2000.**

# Discretization: grid and nodal values II

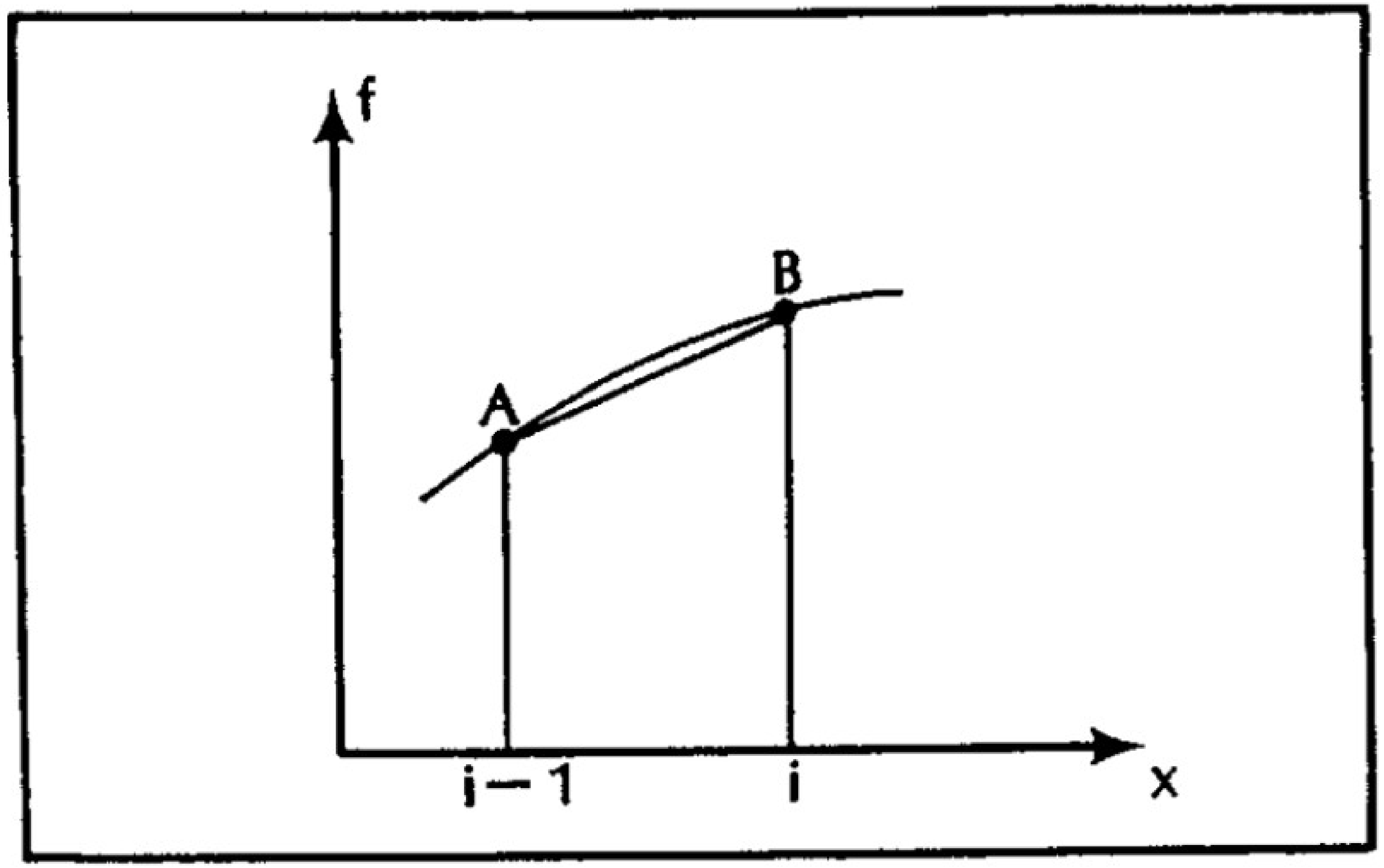



Figure 2-2. Illustration of grid points used in Equation (2-6).

**From: K.A. Hoffmann & S.T. Chiang, Computational Fluid Dynamics: Volume I, Engineering Education System, 2000.**

# Definition of the derivative

- **Limit of the difference = differential**, the **quotient** is the **derivative**

$$\boxed{f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x} \stackrel{!}{=} \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}}$$

- Geometrically, *f′*(*x*) corresponds to the slope of the tangent to the graph of the function *f* at point *x.*

- **Question**: How can *f′(x)* be approximated?

# Approximations of the derivative: Finite difference stencils

- Finite Difference Method (FDM): $\cancel{\lim_{\Delta x \to 0}}$

  Terminate the limit value process at a finite but ”*small*” $\Delta x$ !

- **Forward difference**

$$f_{\mathrm{V}}'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

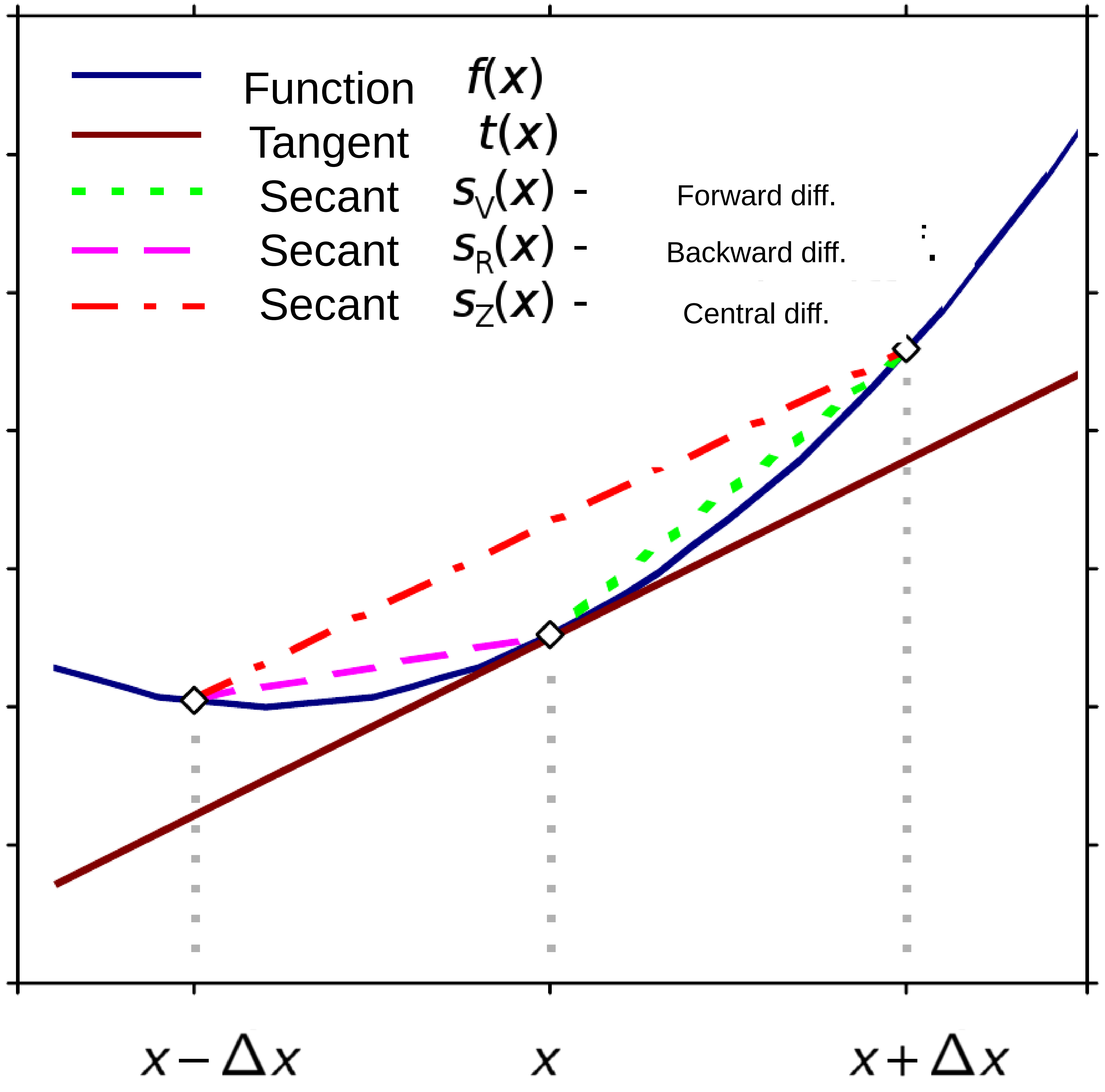
- **Backward difference**

$$f_{\mathrm{R}}'(x) = \frac{f(x) - f(x-\Delta x)}{\Delta x}$$

- **Central difference** (‘symmetric’ difference)

$$f_{\mathrm{Z}}'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$$

# Graphical interpretation




**After: S. M. Holzer, Institute for Mathematics and Building Informatics, University of the Federal Armed Forces Munich, 2002.**

# Wait!

- Apparently, we have

$$f'(x) \approx f'_\mathrm{V}(x) \approx f'_\mathrm{R}(x) \approx f'_\mathrm{Z}(x)$$

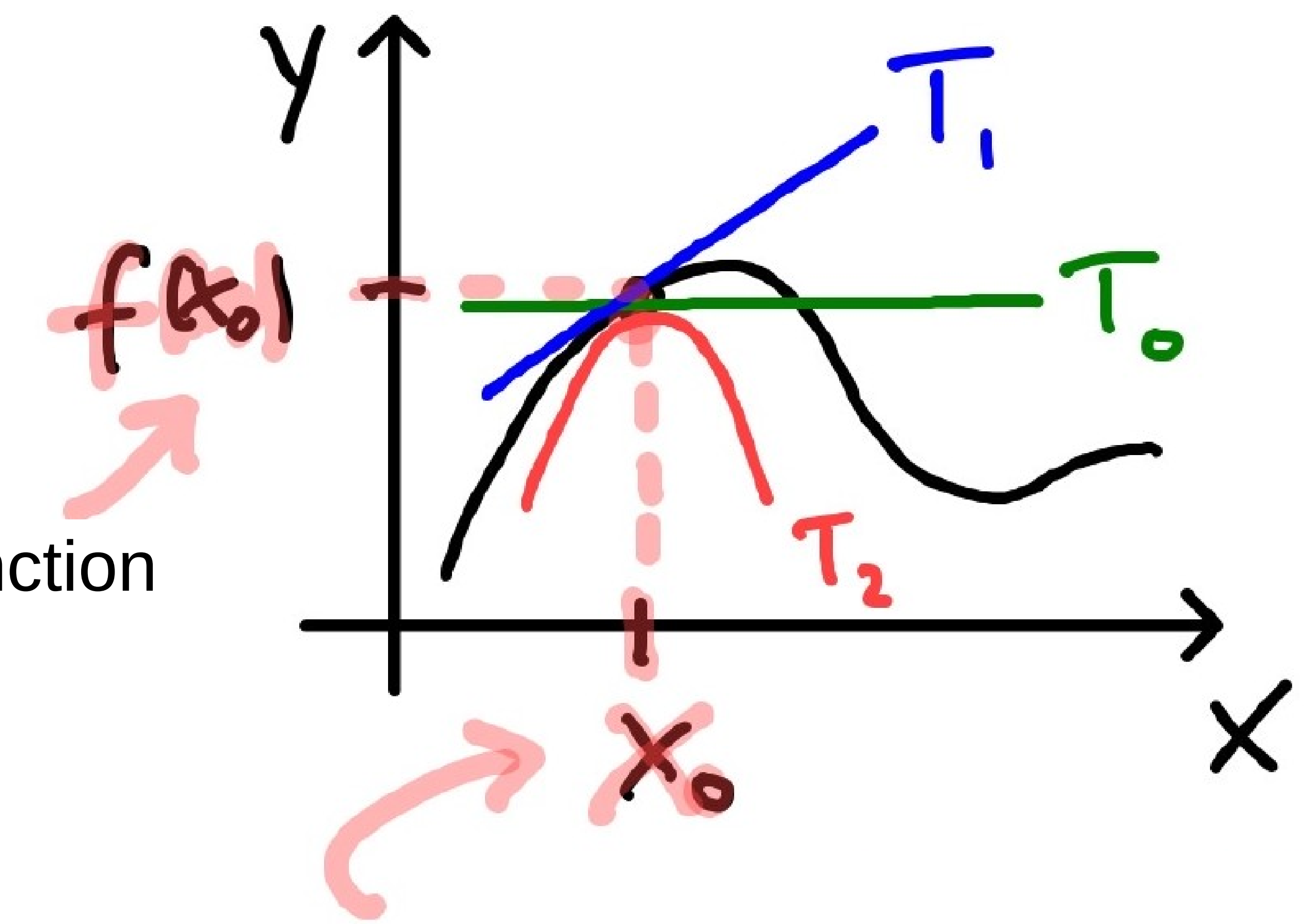
- **But**,
    - Does that always apply? How `good’ are these approximations?
    - How large is the error?
    - What is the impact of rounding and discretization errors? Is there an `optimal’ step size Δx?

# Interlude: Taylor series

- The **Taylor series** $T_N(x; x_0)$ denotes the expansion of a function $f(x)$ around a point $x_0$ into an order $N$ polynomial.

$$T_N(x; x_0) = \sum_{n=0}^{N} \frac{1}{n!} \cdot f^{(n)}(x_0) \cdot (x - x_0)^n$$

- **Note**: If $f(x)$ is sufficiently smooth, the sum converges quickly. The smallest $n = 0, 1, 2, \ldots$ then make the largest contribution.




Known function value

Reference point

# Discretization error: Forward and backward difference

- **The idea for analyzing FDM-approximations**
  - Insert the Taylor series around $x$ for $y_{i\pm1} = f(x \pm \Delta x)$
  - Error of the derivation: $\varepsilon = f'_{\mathrm{FDM}}(x) - f'(x)$
- **Forward difference** (backward difference is analogous)

$$\begin{aligned}
\varepsilon_{\mathrm{V}} &= f'_{\mathrm{V}}(x) - f'(x) \\
&= \frac{\cancel{f(x)} + \Delta x \cdot f'(x) + \frac{1}{2} \cdot \Delta x^2 \cdot f''(x) + \ldots - \cancel{f(x)}}{\Delta x} - f'(x) \\
&= \cancel{f'(x)} + \frac{1}{2} \cdot \Delta x^{1} \cdot f''(x) + \ldots - \cancel{f'(x)} = O(\Delta x)
\end{aligned}$$

- Forward (and backward) differencing is first-order accurate $O(\Delta x)$.
- The error of the derivative decreases linearly with $\Delta x$. Double mesh size means factor 2 larger error.

# Discretization error of central difference

$$
\begin{aligned}
\varepsilon_Z &= f_Z'(x) - f'(x) \\
&= \frac{\cancel{f(x)} + \Delta x \cdot f'(x) + \cancel{\tfrac{1}{2} \cdot \Delta x^2 \cdot f''(x)} + \tfrac{1}{6} \cdot \Delta x^3 \cdot f'''(x) + \ldots}{2\Delta x} \\
&\quad - \frac{\cancel{f(x)} - \Delta x \cdot f'(x) + \cancel{\tfrac{1}{2} \cdot \Delta x^2 \cdot f''(x)} - \tfrac{1}{6} \cdot \Delta x^3 \cdot f'''(x) + \ldots}{2\Delta x} - f'(x) \\
&= \cancel{f'(x)} + \frac{1}{6} \cdot \Delta x^{2} \cdot f'''(x) + \ldots - \cancel{f'(x)} = O(\Delta x^2)
\end{aligned}
$$

The central difference is of second order accuracy $O(\Delta x^2)$.

- The error of the derivation thus decreases quadratically with $\Delta x$. Half mesh size means ¼ of the error.
- The central difference is more accurate than the forward or backward difference, but at the expanse of a larger stencil.
- **But**: It depends on the behavior of the higher derivatives: $f''(x)$ for $f'_{V/R}$ $f'''(x)$ for $f'_Z$.

# Consistency

- The **consistency** of the listed FDM approximations follows from the definition of the derivative, yielding the leading order error $\varepsilon = O(\Delta x^p)$, with $p > 0$: The error must vanish in the limit of small meshes.

$$\lim_{\Delta x \to 0} f'_{\mathrm{FDM}}(x) = \lim_{\Delta x \to 0} \left[ f'(x) + \underbrace{O(\Delta x^p)}_{\to 0} \right] \stackrel{!}{=} f'(x)$$

# Minimum error and optimal step size

**Q:** Which method allows to achieve a relative error of the order of $10^{-16}$ (double precision)?

- Based on $O(\Delta x^p)$One would perhaps naively expect that . . .
  - $\Delta x \leq 10^{-8}$ for the **central difference** and
  - $\Delta x \leq 10^{-16}$ applies to the **forward & backward difference**.
- For these small numbers, however, the *round-off error due to the division by* $\Delta x$ must be taken into account. It is of the order $10^{-16}/\Delta x$. i.e., the loss of digits in the mantissa due to carry-over to the exponent.
- The actual error results from the sum of the **discretization error** and the **round-off error**.
- The “optimal” $\Delta x$, i.e. the $\Delta x$ for which the **total error is minimal**, is approximately reached when both error contributions are of the same magnitude.

# Estimation of minimum error and optimal step size

- **Forward difference** (backward difference analogous)

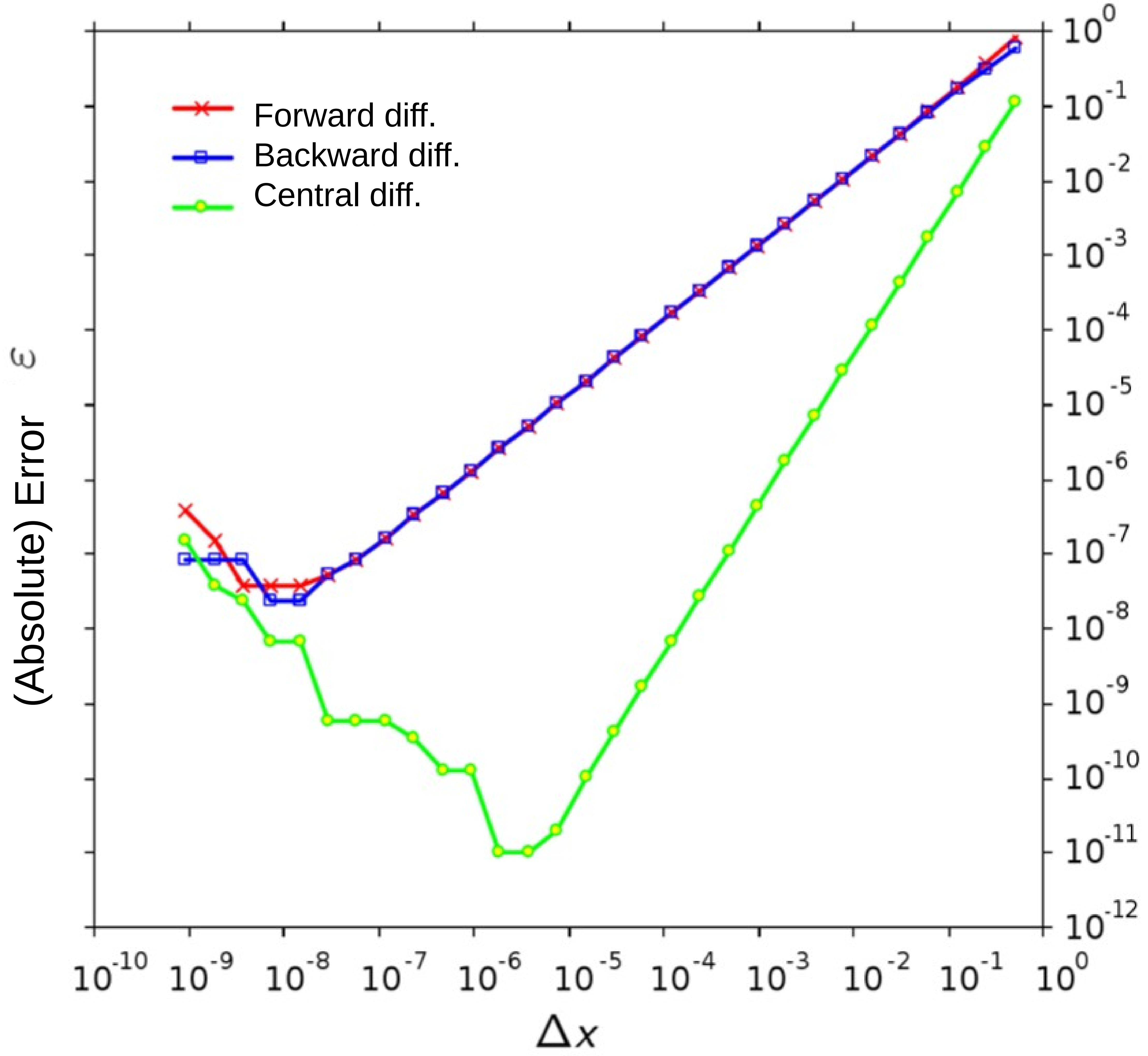
$$\frac{10^{-16}}{\Delta x_{\mathrm{min}}} \overset{!}{\simeq} \Delta x_{\mathrm{min}} \quad \Rightarrow \quad \Delta x_{\mathrm{min}} \simeq 10^{-8}, \quad \varepsilon_{\mathrm{V,min}} \simeq 10^{-8}$$

- **Central difference**

$$\frac{10^{-16}}{\Delta x_{\mathrm{min}}} \overset{!}{\simeq} \Delta x_{\mathrm{min}}^2 \quad \Rightarrow \quad \Delta x_{\mathrm{min}} \approx 10^{-5}, \quad \varepsilon_{\mathrm{Z,min}} \simeq 10^{-10}$$

- **Note:**
  - The minimum error is limited.
  - A numerical method with *higher-order accuracy* achieves a smaller error with a larger step size.

# Convergence




← error of $f'(x)$

using:

$f(x) = e^x$

evaluated at:

$x = 2$

**According to: S. M. Holzer, Institute for Mathematics and Building Informatics, University of the Federal Armed Forces Munich, 2002.**

# 4 Elementary numerical methods: Differentiation

## 4.4. Excursion: Second Derivative

# Second derivative I

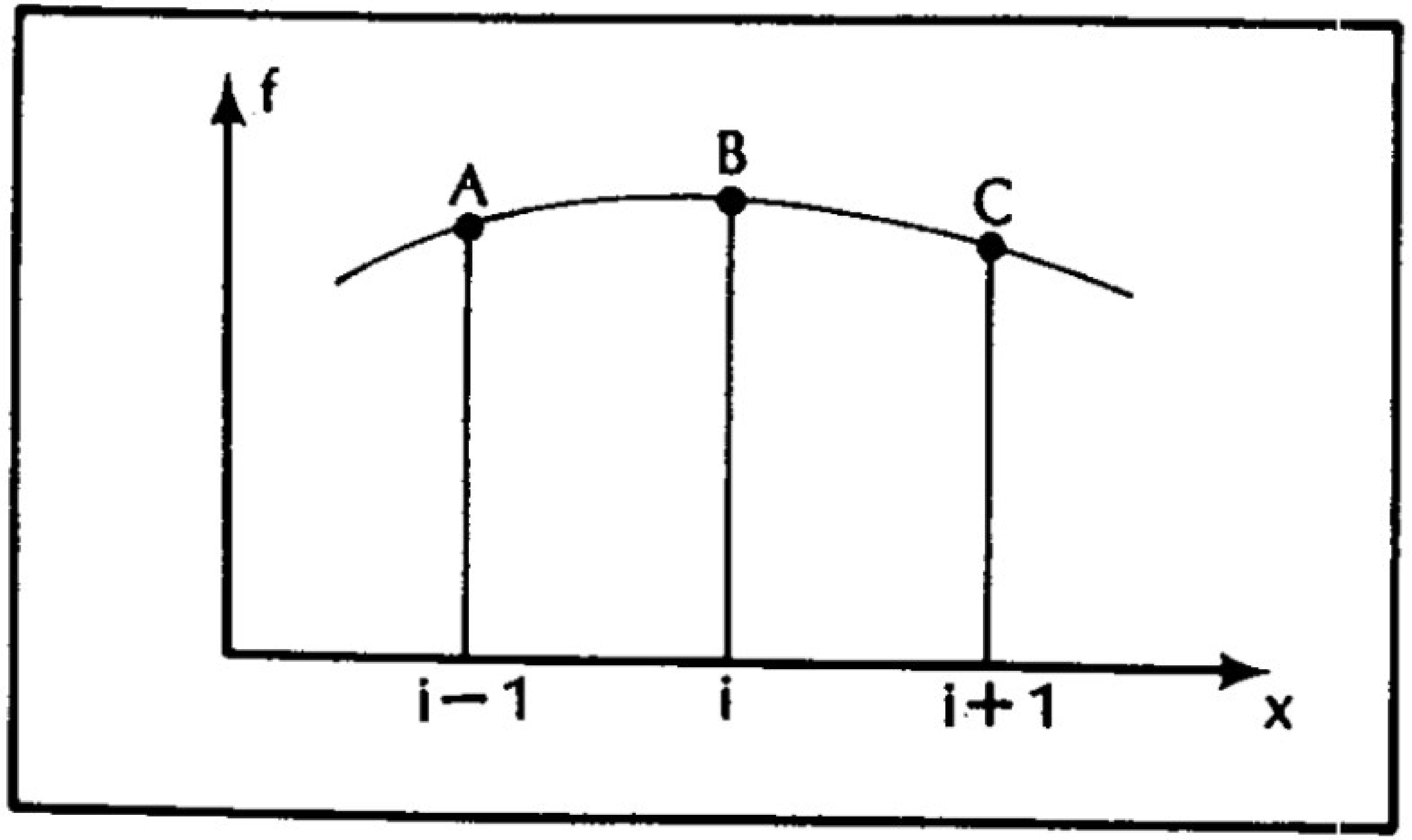



Figure 2-3. Illustration of grid points used in Equation (2-10).

Aus: K.A. Hoffmann & S.T. Chiang, *Computational Fluid Dynamics: Volume I*, Engineering Education System, 2000.

# Second derivative II

- **3-point difference stencil** approximates the second derivative

$$f''(x) \approx f''_Z(x) = \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2}$$

- This scheme is of **second order** accuracy

$$\varepsilon_{Z,2} = f''_Z(x) - f''(x) = O(\Delta x^2)$$

**(without derivation)**

# Keywords

- Definition of the derivative
- Finite Difference Method (FDM)
- difference stencil
- Taylor series
- Numerical errors