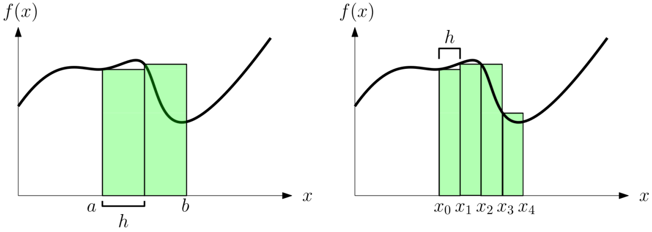
# Lecture 2: Numerical Integration (II)

(Adapted from slides by Uriel Morzan)

Gerald E. Fux
(gfux@ictp.it)

13. Oct. 2023

# Last Lecture: Left Riemann Sum

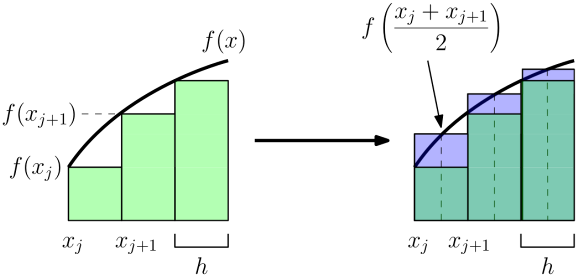



$$I_L(N) \equiv \sum_{k=0}^{N-1} f(x_k) \cdot h \quad \text{with} \quad \begin{cases} h \equiv \frac{b-a}{N} \\ x_k \equiv a + k \cdot h \\ k = 0, \ldots, N-1 \end{cases}$$

Convergence:

$$\int_a^b f(x)\,\mathrm{d}x = I_L(N) + \mathcal{O}\left(\frac{1}{N}\right)$$

# Improved Method (a): Midpoint Method



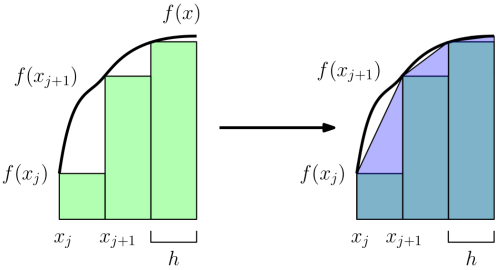

Better coverage
of area under the curve

$$I_M(N) \equiv \sum_{k=0}^{N-1} f\left(\frac{x_k + x_{k+1}}{2}\right) \cdot h \quad \text{with} \quad \begin{cases} h \equiv \frac{b-a}{N} \\ x_k \equiv a + k \cdot h \\ k = 0, \ldots, N-1 \end{cases}$$

Better convergence:

$$\int_a^b f(x)\,\mathrm{d}x = I_M(N) + \mathcal{O}\left(\frac{1}{N^2}\right) = I_M(h) + \mathcal{O}\left(h^2\right)$$

# Improved Method (b): Trapezoidal Method




Area of trapeze starting in $x = x_j$:

$$A_j = \frac{h}{2}\left(f(x_j) + f(x_{j+1})\right)$$

$$I_T(N) \equiv \sum_{k=0}^{N-1} \left[f(x_k) + f(x_{k+1})\right] \cdot \frac{h}{2} \quad \text{with} \quad \begin{cases} h \equiv \frac{b-a}{N} \\ x_k \equiv a + k \cdot h \\ k = 0, \dots, N \end{cases}$$

Convergence (same as the midpoint method):

$$\int_a^b f(x)\,\mathrm{d}x = I_T(h) + \mathcal{O}\left(h^2\right)$$

# Improved Method (b): Trapezoidal Method

$$
\begin{aligned}
I_T(N) &\equiv \sum_{k=0}^{N-1} \left[f(x_k) + f(x_{k+1})\right] \cdot \frac{h}{2} \\
&\equiv \frac{h}{2} \left[\underbrace{f(x_0) + f(x_1)} + \underbrace{f(x_1) + f(x_2)} + \underbrace{f(x_2) + f(x_3)} + \dots\right]
\end{aligned}
$$

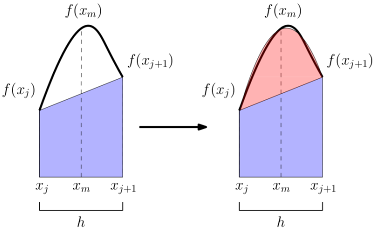
## (Better) reformulation of the trapezoidal method

Note that $f(x_1), f(x_2), \dots, f(x_{N-2})$ each appear twice in the sum. Because it might be very hard to evaluate $f(x)$ it is better to **calculate each $f(x_j)$ only once instead of twice**. We thus implement the method in the rewritten form …

$$
I_T(N) = \frac{h}{2}\left[f(x_0) + \left(\sum_{k=1}^{N-1} 2f(x_k)\right) + f(x_N)\right].
$$

# Improved Method (c): Simpson Method

Next improvement: from Trapeziods $\rightarrow$ to **parabolic arcs.**



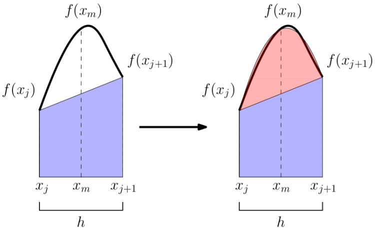

Each arc passes for

- $(x_j, f(x_j))$
- $\left(\frac{x_j + x_{j+1}}{2}, f\left(\frac{x_j + x_{j+1}}{2}\right)\right)$
- $(x_{j+1}, f(x_{j+1}))$

Algebra yields that the area is: $A_j = \frac{h}{6}\left[f(x_j) + 4f\left(\frac{x_j + x_{j+1}}{2}\right) + f(x_{j+1})\right]$

$$I_S(N) \equiv \frac{h}{6}\left[f(x_0) + 2\sum_{k=1}^{N-1} f(x_k) + 4\sum_{k=0}^{N-1} f\left(\frac{x_k + x_{k+1}}{2}\right) + f(x_N)\right] \quad \text{with} \quad \begin{cases} h \equiv \frac{b-a}{N} \\ x_k \equiv a + k \cdot h \\ k = 0, \dots, N \end{cases}$$

# Improved Method (c): Simpson Method




Each arc passes for

- $(x_j, f(x_j))$
- $\left(\frac{x_j + x_{j+1}}{2}, f\left(\frac{x_j + x_{j+1}}{2}\right)\right)$
- $(x_{j+1}, f(x_{j+1}))$

$$I_S(N) \equiv \frac{h}{6}\left[f(x_0) + 2\sum_{k=1}^{N-1} f(x_k) + 4\sum_{k=0}^{N-1} f\left(\frac{x_k + x_{k+1}}{2}\right) + f(x_N)\right] \quad \text{with} \quad \begin{cases} h \equiv \frac{b-a}{N} \\ x_k \equiv a + k \cdot h \\ k = 0, \ldots, N \end{cases}$$

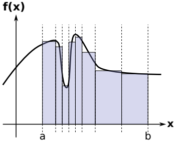
Convergence:

$$\int_a^b f(x)\,\mathrm{d}x = I_S(h) + \mathcal{O}\left(h^4\right)$$

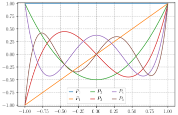
# Advanced Integration Methods

Beyond these basic approaches many advanced / specialized methods exist. E.g.:

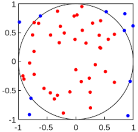
**Adaptive integration**:
Make the grid finer where the function changes faster.




**Gaussian quadratures**:
Mathematically optimal grid.



**Monte Carlo integration**:
Use a randomized grid; best in high dimensions.



# Assignment 9

Write a FORTRAN program that computes $\int_a^b f(x)\,\mathrm{d}x$ for $f(x) = e^x$ using the Midpoint, Trapeze, and Simpson method:

- Write a function (for each of the three methods) that takes the bounds $a$ and $b$, and the desired precision $\epsilon$.
- The function should integrate with the Midpoint/Trapeze/Simpson method, increasing N until the precision is achieved.
- The function should print the result at each step together with the current value for $N$ (this is just for us to see what is happening during the calculation).
- Test the function by calculating $\int_0^1 e^x\,\mathrm{d}x$ with error threshold $10^{-5}$ in the main program and print the result.
- Comment (in the email) about how often the function f(x) is called in total for each method.
- Submit your code as `<surname>-assignment-9.f90` to gfux@ictp.it before the next lesson.

**Hints:**

- You can recycle the previous assignment, adding new functions.