ENM 360: Introduction to Data-driven Modeling

Lecture #3: Function approximation

Paris Perdikaris
September 9, 2021

UNIVERSITY *of* PENNSYLVANIA

Example #1: Atmospheric science

	δ_K			
Latitude	$K = 0.67$	$K = 1.5$	$K = 2.0$	$K = 3.0$
65	-3.1	3.52	6.05	9.3
55	-3.22	3.62	6.02	9.3
45	-3.3	3.65	5.92	9.17
35	-3.32	3.52	5.7	8.82
25	-3.17	3.47	5.3	8.1
15	-3.07	3.25	5.02	7.52
5	-3.02	3.15	4.95	7.3
-5	-3.02	3.15	4.97	7.35
-15	-3.12	3.2	5.07	7.62
-25	-3.2	3.27	5.35	8.22
-35	-3.35	3.52	5.62	8.8
-45	-3.37	3.7	5.95	9.25
-55	-3.25	3.7	6.1	9.5

Table 3.1. Variation of the average yearly temperature on the Earth for four different values of the concentration K of carbon acid at different latitudes

Example #2: Finance

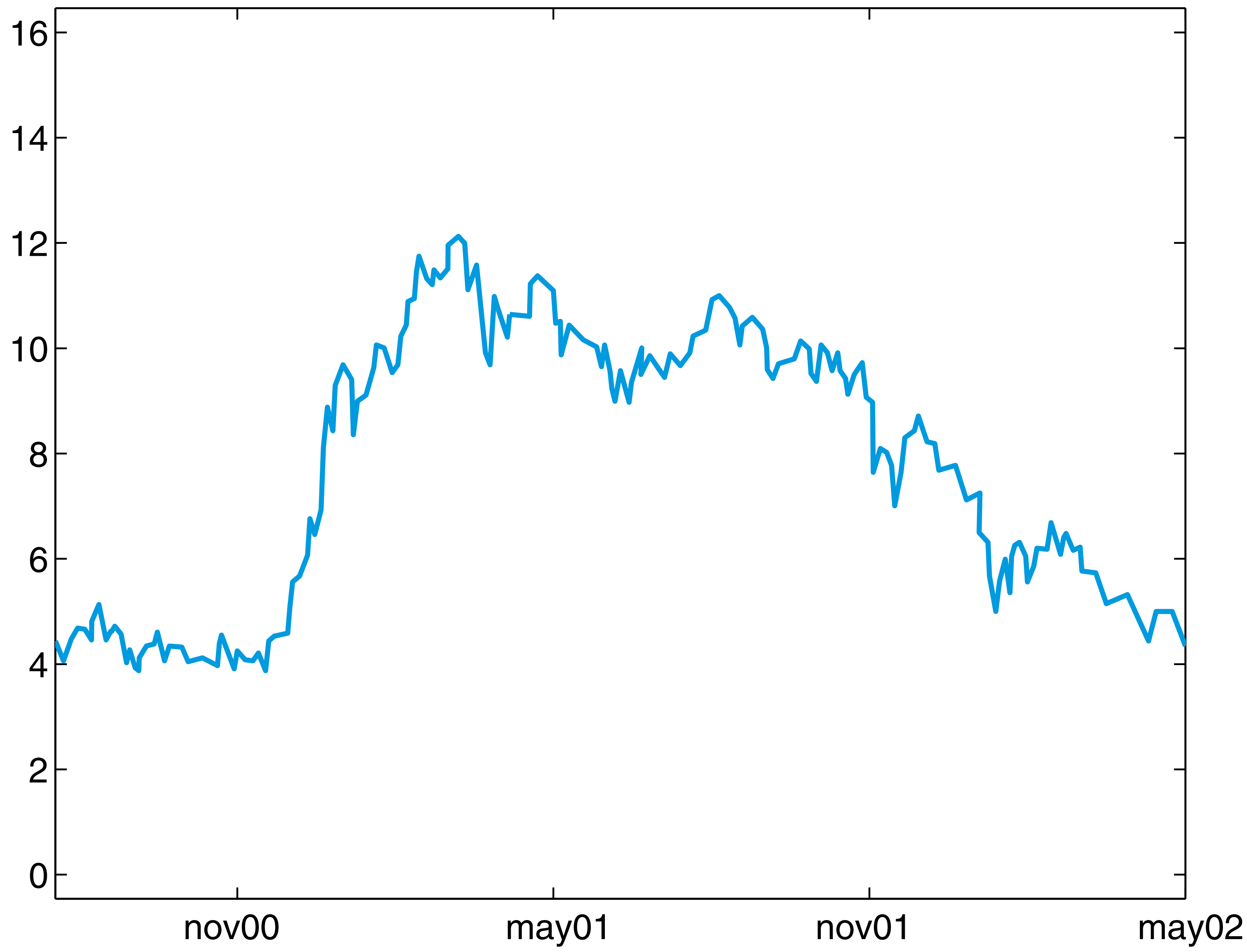


Fig. 3.1. Price variation of a stock over two years

Example #3: Biomechanics

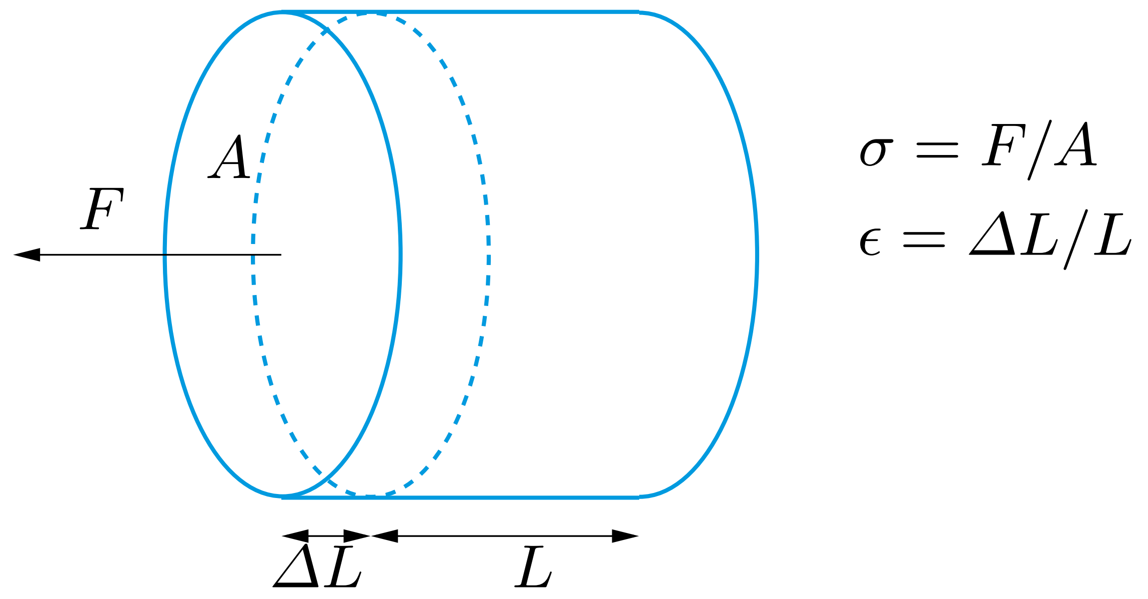


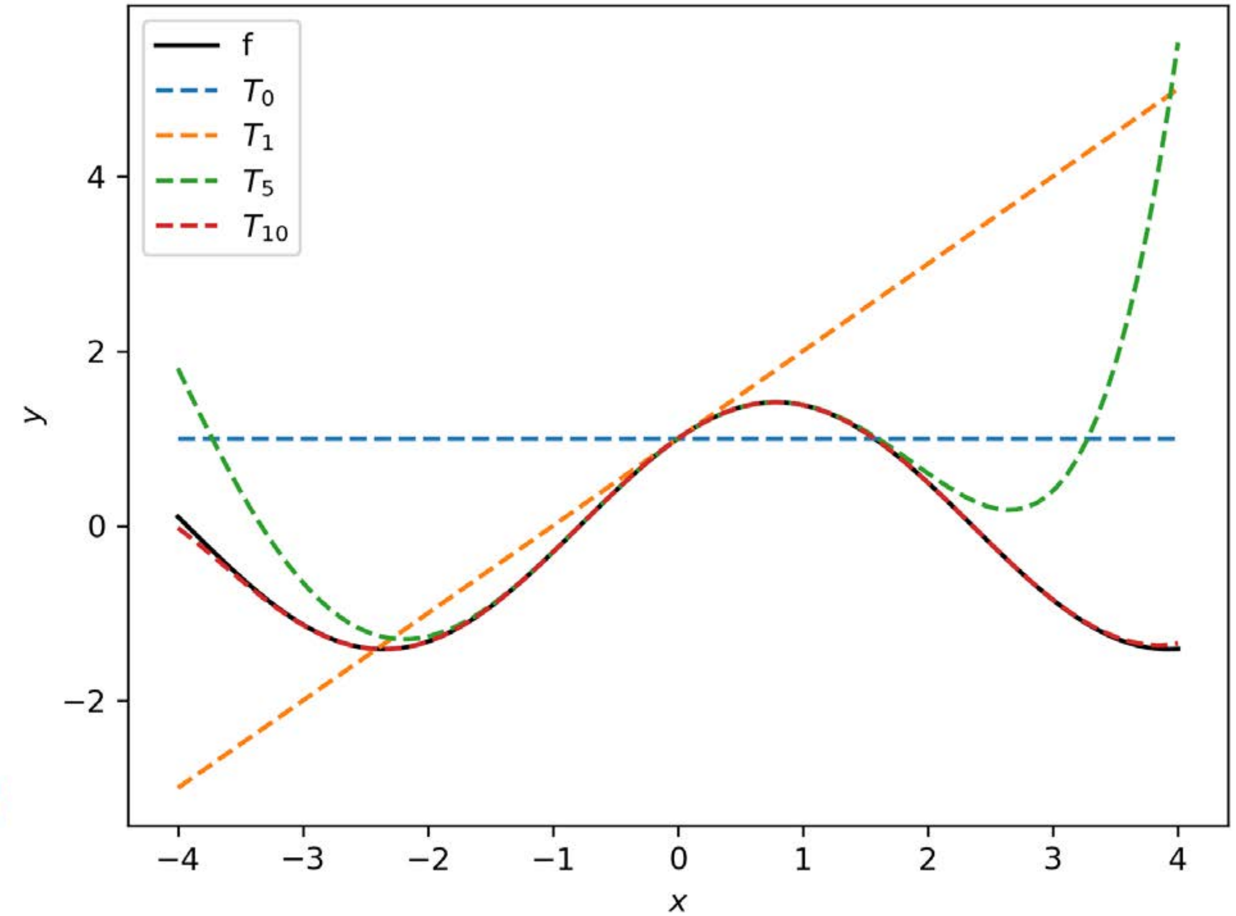
Fig. 3.2. A schematic representation of an intervertebral disc

test	stress σ	stress ϵ	test	stress σ	stress ϵ
1	0.00	0.00	5	0.31	0.23
2	0.06	0.08	6	0.47	0.25
3	0.14	0.14	7	0.60	0.28
4	0.25	0.20	8	0.70	0.29

Table 3.2. Values of the deformation for different values of a stress applied on an intervertebral disc

Local approximation with Taylor series

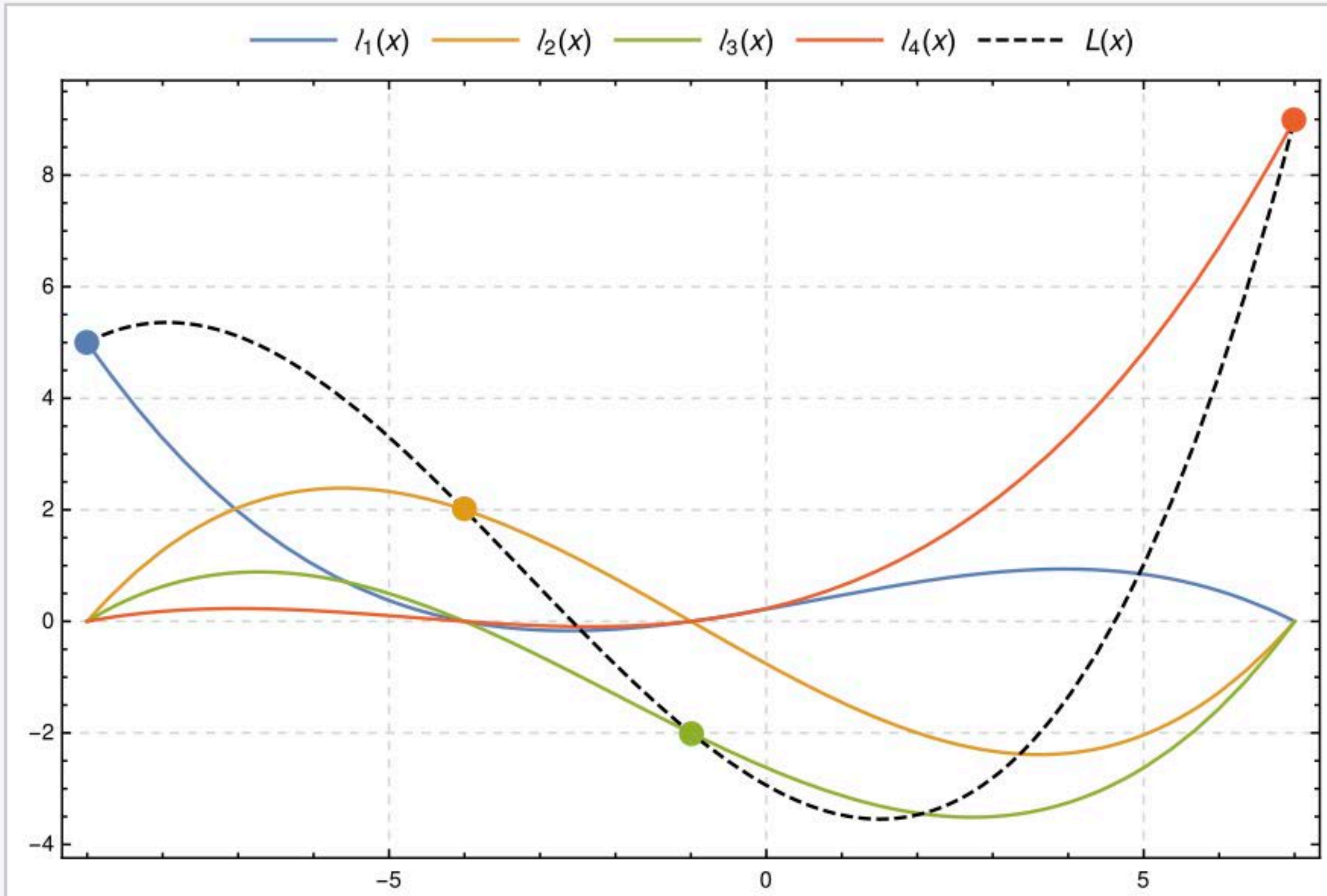
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Aug 28 12:27:37 2018

@author: paris
"""

import autograd.numpy as np
from autograd import grad
from scipy.special import factorial
import matplotlib.pyplot as plt

if __name__ == '__main__':

    def f(x):
        return np.sin(x) + np.cos(x)

    def TaylorSeries(f, x, x0, n = 2):
        T = f(x0)*np.ones_like(x)
        grad_f = grad(f)
        for i in range(0, n):
            T += grad_f(x0)*(x-x0)**(i+1) / factorial(i+1)
            grad_f = grad(grad_f)
        return T


    N = 100
    x = np.linspace(-4.0,4.0,N)
    y = f(x)

    x0 = 0.0

    n = [0, 1, 5, 10]
    plt.figure(1)
    plt.plot(x, y, 'k-', label = 'f')
    for i in range(0, len(n)):
        T = TaylorSeries(f, x, x0, n[i])
        plt.plot(x, T, '--', label = '$T_{%d}$' % (n[i]))
    plt.xlabel('$x$')
    plt.ylabel('$y$')
    plt.legend()
```



$$T_n(x) := \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

Interpolation with Lagrange polynomials

$$f(x) = \sum_{k=1}^{n} y_k \phi_k(x), \quad \phi_k(x) = \prod_{\substack{0 \le k \le n \\ k \ne j}} \frac{x - x_j}{x_k - x_j}$$



This image shows, for four points ((−9, 5), (−4, 2), (−1, −2), (7, 9)), the (cubic) interpolation polynomial $L(x)$ (dashed, black), which is the sum of the *scaled* basis polynomials $y_0\ell_0(x)$, $y_1\ell_1(x)$, $y_2\ell_2(x)$ and $y_3\ell_3(x)$. The interpolation polynomial passes through all four control points, and each *scaled* basis polynomial passes through its respective control point and is 0 where x corresponds to the other three control points.

Runge’s phenomenon

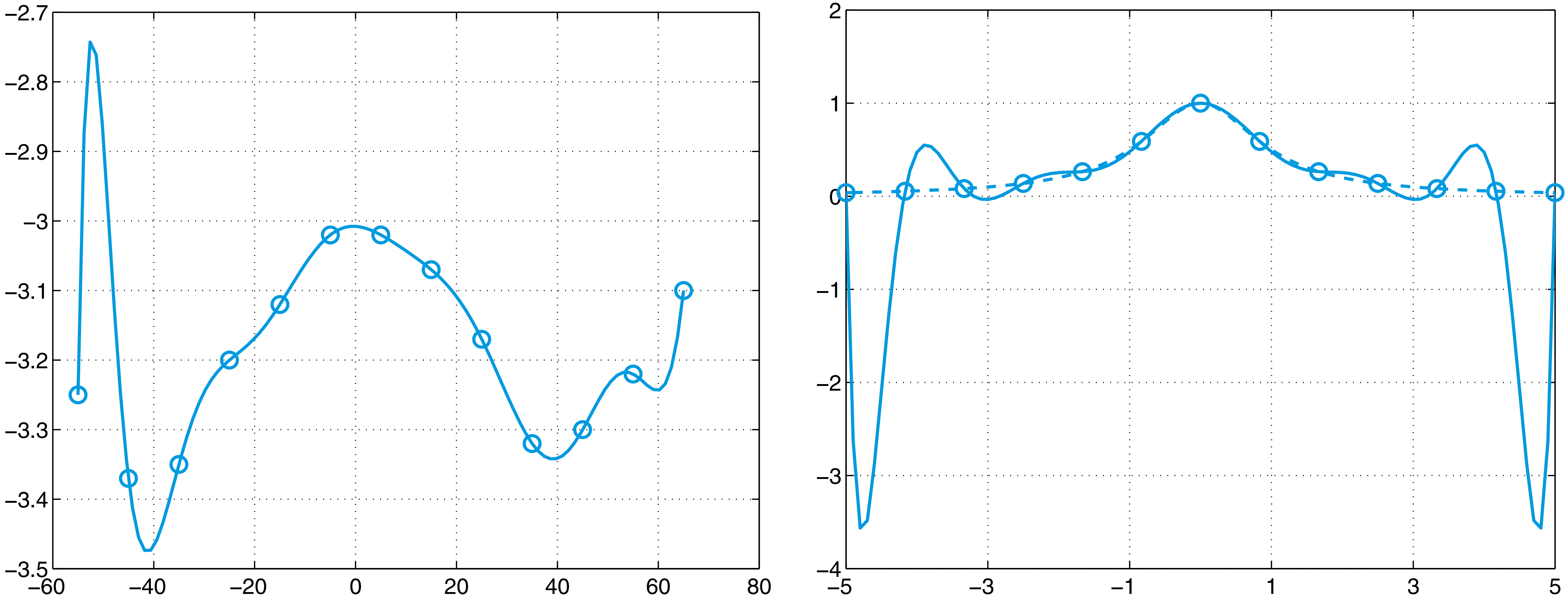


Fig. 3.6. Two examples of Runge’s phenomenon: to the left, Π_{12} computed for the data of Table 3.1, column $K = 0.67$; to the right, $\Pi_{12}f$ (*solid line*) computed on 13 equispaced nodes for the function $f(x) = 1/(1+x^2)$ (*dashed line*)

Interpolation with trigonometric polynomials

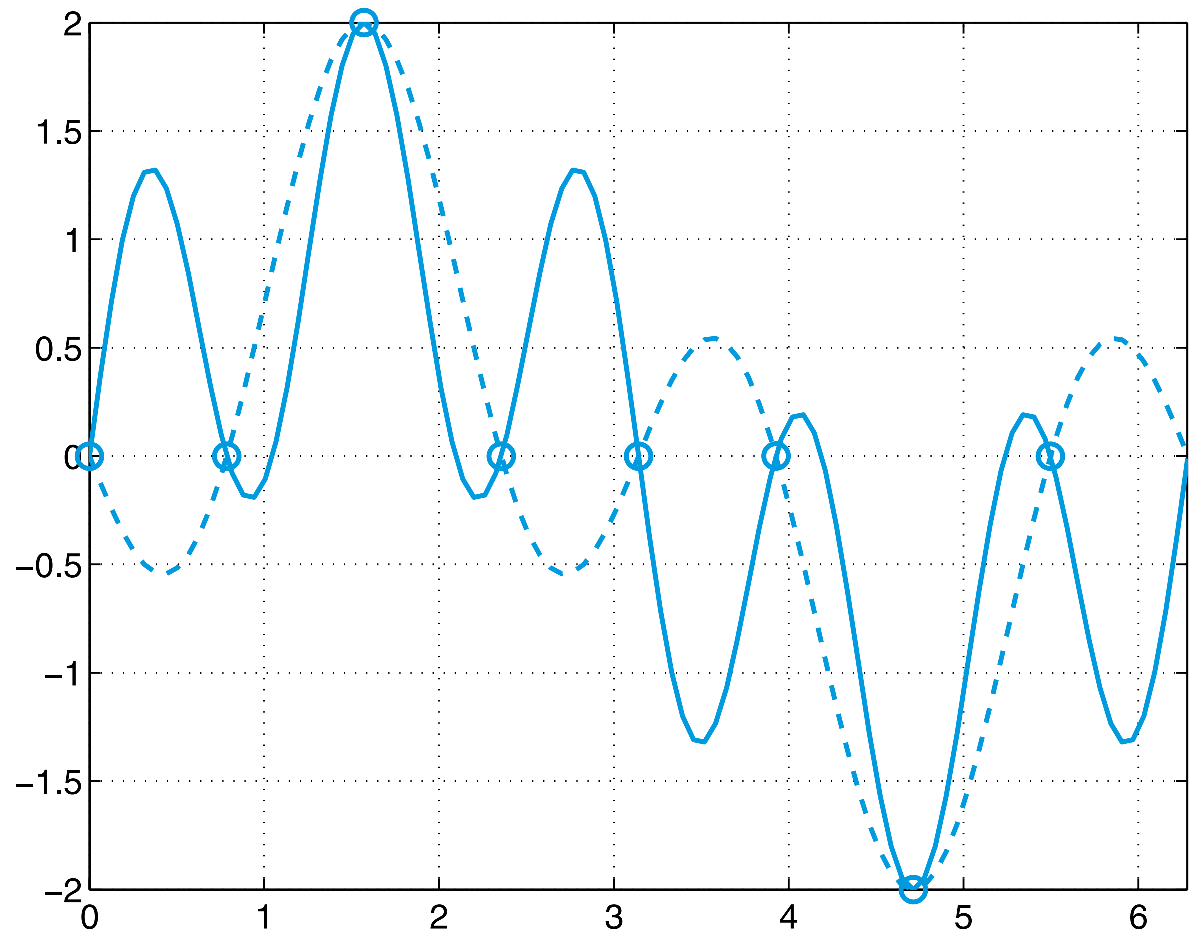
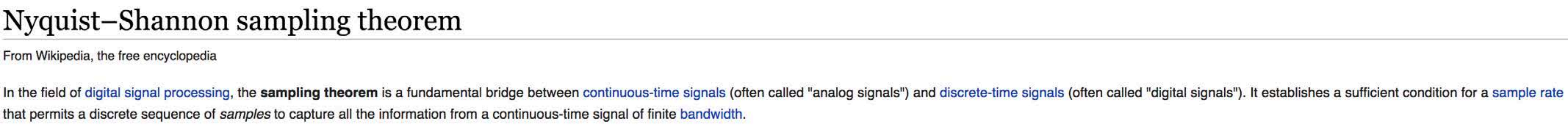


Fig. 3.9. The effects of aliasing: comparison between the function $f(x) = \sin(x)+\sin(5x)$ (*solid line*) and its trigonometric interpolant (3.11) with $M = 3$ (*dashed line*)



Nyquist–Shannon sampling theorem

From Wikipedia, the free encyclopedia

In the field of digital signal processing, the **sampling theorem** is a fundamental bridge between continuous-time signals (often called "analog signals") and discrete-time signals (often called "digital signals"). It establishes a sufficient condition for a sample rate that permits a discrete sequence of *samples* to capture all the information from a continuous-time signal of finite bandwidth.

Numerical differentiation with finite differences

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & 1 & -2 & 1 & & \\ & & & 1 & -2 & 1 & \\ & & & & \ddots & \ddots & \ddots \\ & & & & & 1 & -2 \end{bmatrix}$$

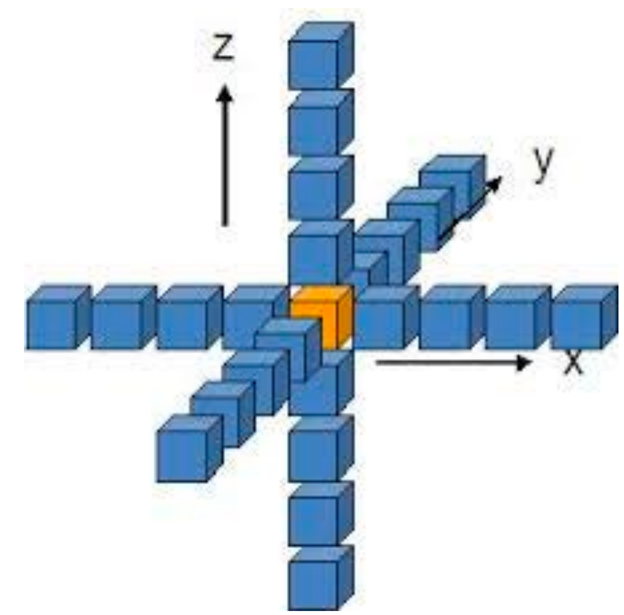
Central difference stencil for second derivative approximation in 1D

$$\left[\begin{array}{cccc:cccc:cccc:cccc}
\mathbf{4} & -1 & & & -1 & & & & & & & & & & & \\
-1 & \mathbf{4} & -1 & & & -1 & & & & & & & & & & \\
 & -1 & \mathbf{4} & -1 & & & -1 & & & & & & & & & \\
 & & -1 & \mathbf{4} & & & & -1 & & & & & & & & \\ \hdashline
-1 & & & & \mathbf{4} & -1 & & & -1 & & & & & & & \\
 & -1 & & & -1 & \mathbf{4} & -1 & & & -1 & & & & & & \\
 & & -1 & & & -1 & \mathbf{4} & -1 & & & -1 & & & & & \\
 & & & -1 & & & -1 & \mathbf{4} & & & & -1 & & & & \\ \hdashline
 & & & & -1 & & & & \mathbf{4} & -1 & & & -1 & & & \\
 & & & & & -1 & & & -1 & \mathbf{4} & -1 & & & -1 & & \\
 & & & & & & -1 & & & -1 & \mathbf{4} & -1 & & & -1 & \\
 & & & & & & & -1 & & & -1 & \mathbf{4} & & & & -1 \\ \hdashline
 & & & & & & & & -1 & & & & \mathbf{4} & -1 & & \\
 & & & & & & & & & -1 & & & -1 & \mathbf{4} & -1 & \\
 & & & & & & & & & & -1 & & & -1 & \mathbf{4} & -1 \\
 & & & & & & & & & & & -1 & & & -1 & \mathbf{4}
\end{array}\right]$$

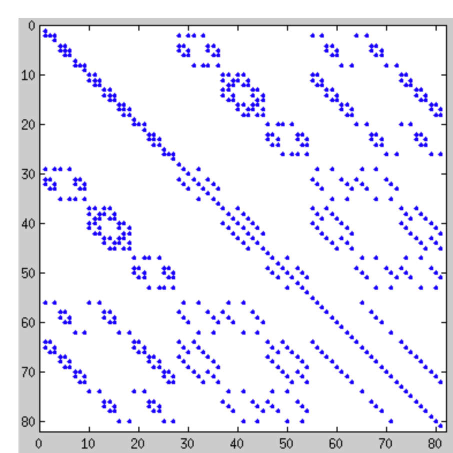
Central difference stencil for second derivative approximation in 2D

1D

2D



3D



Sparsity pattern of a realistic FD stencil matrix

Numerical integration: The midpoint rule

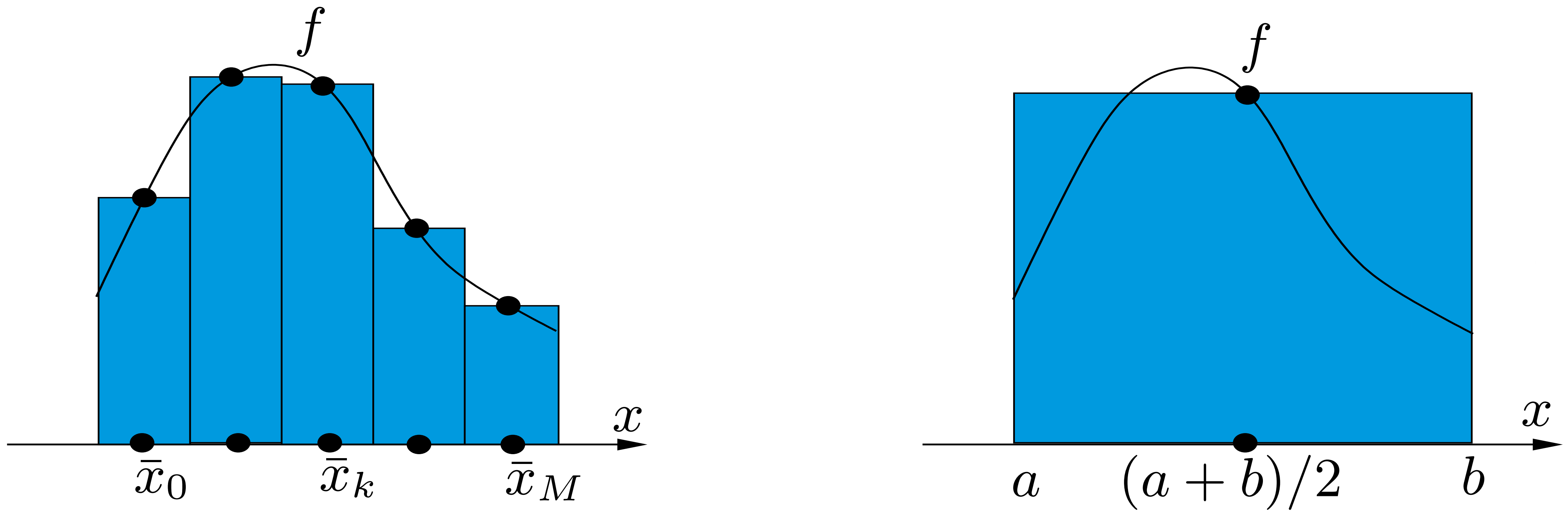


Fig. 4.3. The composite midpoint formula (*left*); the midpoint formula (*right*)

$$\boxed{I^c_{mp}(f) = H \sum_{k=1}^{M} f(\bar{x}_k)}$$

$$\boxed{I_{mp}(f) = (b-a) f[(a+b)/2]}$$

Numerical integration: The trapezoidal rule

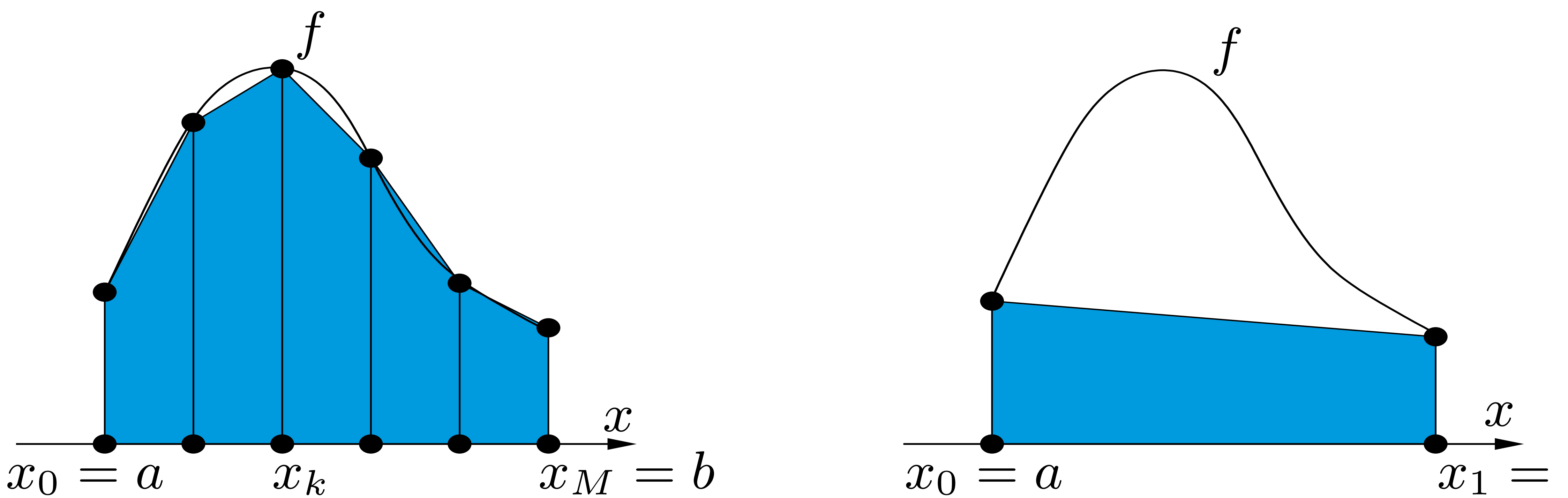


Fig. 4.4. Composite trapezoidal formula (*left*); trapezoidal formula (*right*)

$$\boxed{\begin{aligned} I_t^c(f) &= \frac{H}{2}\sum_{k=1}^{M}[f(x_k)+f(x_{k-1})] \\ &= \frac{H}{2}[f(a)+f(b)] + H\sum_{k=1}^{M-1} f(x_k) \end{aligned}}$$

$$\boxed{I_t(f) = \frac{b-a}{2}[f(a)+f(b)]}$$

Numerical integration: Simpson's rule

$$I_s(f) = \frac{b-a}{6}\left[f(a) + 4f((a+b)/2) + f(b)\right]$$

Simpson's formula

$$I_s^c(f) = \frac{H}{6}\sum_{k=1}^{M}\left[f(x_{k-1}) + 4f(\bar{x}_k) + f(x_k)\right]$$

The composite Simpson's rule

Gauss-Legendre quadrature

$$I_s(f) = \sum_{j=1}^{n} w_j f(x_j)$$

w_j *weights*

x_j *nodes*

n	x_j	w_j
1	$\{\pm 1/\sqrt{3}\}$	$\{1\}$
2	$\{\pm\sqrt{15}/5, 0\}$	$\{5/9, 8/9\}$
3	$\left\{\pm(1/35)\sqrt{525 - 70\sqrt{30}},\right.$	$\{(1/36)(18 + \sqrt{30}),$
	$\left.\pm(1/35)\sqrt{525 + 70\sqrt{30}}\right\}$	$(1/36)(18 - \sqrt{30})\}$
4	$\left\{0, \pm(1/21)\sqrt{245 - 14\sqrt{70}}\right.$	$\{128/225, (1/900)(322 + 13\sqrt{70})$
	$\left.\pm(1/21)\sqrt{245 + 14\sqrt{70}}\right\}$	$(1/900)(322 - 13\sqrt{70})\}$

Table 4.1. Nodes and weights for some quadrature formulae of Gauss-Legendre on the interval $(-1, 1)$. Weights corresponding to symmetric couples of nodes are reported only once