Numerical Analysis and Computational Mathematics

MATH–456

CSE — Computational Science and Engineering, EPFL

A.Y. 2015/16

Dr. Luca Dede'

The content of these lecture notes is mainly taken and elaborated from [2] and [1].
Written with LaTeX, based on "The Legrand Orange Book LaTeXTemplate"
Lausanne, December 15, 2015

Contents

1 Introduction to Numerical Analysis 1

1.1 Machine Representation of Real Numbers 1

1.1.1 Floating point numbers 1

1.1.2 Floating point arithmetic 2

1.2 Mathematical and Numerical Problems 3

1.2.1 The mathematical problem 3

1.2.2 The numerical problem 5

1.2.3 Choice of a numerical method 7

2 Nonlinear Equations 9

2.1 Bisection Method 9

2.1.1 Foundation of the bisection method 9

2.1.2 Algorithm and properties 11

2.1.3 Stopping criterion 13

2.2 Newton Method 13

2.2.1 Newton method 13

2.2.2 Modified Newton method 15

2.2.3 Stopping criterion for Newton method 16

2.2.4 Inexact and quasi–Newton methods 17

2.2.5 Newton method for systems of nonlinear equations 18

2.3 Fixed Point Iterations 19

2.3.1 Nonlinear equations, zeros, fixed points, and iteration functions 19

2.3.2 Fixed point iterations algorithm 20

2.3.3 Convergence properties of fixed point iterations 21

2.3.4 Stopping criterion for fixed point iterations 24

2.3.5 The Newton method as a fixed point iterations method 24

2.3.6 Fixed point iterations for vector valued functions 26

3 Approximation of Functions and Data 27
3.1 Motivations and Examples 27
3.1.1 Approximation of functions by Taylor's polynomials 28
3.2 Interpolation 29
3.2.1 Lagrange interpolating polynomials 29
3.2.2 Trigonometric interpolation 35
3.2.3 Piecewise polynomial interpolation 35
3.2.4 Spline functions 37
3.3 Least–Squares Method 38

4 Numerical Differentiation 41
4.1 Goal and Examples 41
4.2 Finite Differences Schemes 41
4.2.1 Forward and backward finite differences 41
4.2.2 Centered finite differences 42

5 Numerical Integration 45
5.1 Goal and Classification of Quadrature Formulas 45
5.2 Mid–point Quadrature Formulas 46
5.3 Trapezoidal Quadrature Formulas 48
5.4 Simpson Quadrature Formulas 49
5.5 Interpolatory Quadrature Formulas 50
5.5.1 Gauss–Legendre quadrature formulas 52
5.5.2 Gauss–Legendre–Lobatto quadrature formulas 53
5.6 Numerical Integration in Multiple Dimensions 54

6 Linear Systems 55
6.1 Motivations, Examples, and Classification of Methods 55
6.1.1 Goals, examples, and notation 55
6.1.2 Linear systems and complexity 56
6.1.3 Classification of methods for linear systems 57
6.2 Direct Methods 57
6.2.1 "Simple" linear systems 57
6.2.2 LU factorization method 59
6.2.3 Cholesky factorization method 65
6.2.4 Thomas algorithm 66
6.2.5 Accuracy of the numerical solution computed with direct methods 67
6.3 Iterative Methods 70
6.3.1 The general scheme 70
6.3.2 Splitting methods 71
6.3.3 Jacobi and Gauss–Seidel methods 72
6.3.4 Preconditioned Richardson methods 75
6.3.5 Gradient methods 77
6.3.6 Conjugate gradient methods 78
6.3.7 Stopping criterion for iterative methods 80
6.4 A Brief Comparison of Direct and Iterative Methods 81

7 Approximation of Eigenvalues 83
7.1 Definitions and Examples 83
7.2 Power Method 85
7.3 Inverse Power Method 86
7.4 Power and Inverse Power Methods with Shift 87

8 Ordinary Differential Equations 89
8.1 Introduction and Examples 89
8.1.1 The Cauchy problem 89
8.1.2 Well–posedness of the Cauchy problem 90
8.2 Numerical Approximation of Ordinary Differential Equations 91
8.2.1 Forward Euler method 91
8.2.2 Backward Euler method 91
8.2.3 Crank–Nicolson method 92
8.2.4 Heun method 93
8.2.5 Error analysis of the methods 93
8.2.6 Stability of the numerical methods: zero– and absolute stability 94
8.2.7 Runge–Kutta methods 97
8.2.8 Multistep methods 98
8.3 Numerical Approximation of Systems of Ordinary Differential Equations 99
8.3.1 The Cauchy problem, examples, and definitions 99
8.3.2 θ–method 100
8.4 Numerical Approximation of High Order Ordinary Differential Equations 102
8.4.1 Second order ODEs 102
8.4.2 General high order ODEs 104

Bibliography 105

1. Introduction to Numerical Analysis

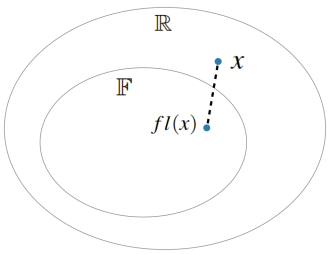
1.1 Machine Representation of Real Numbers

The computer can only represent and operate with a finite set of real numbers $x \in \mathbb{R}$.

1.1.1 Floating point numbers

Definition 1.1 The set of *floating point numbers* $\mathbb{F}$ is the subset of real numbers which can be represented at the computer, i.e. $\mathbb{F} \subset \mathbb{R}$, with $\dim(\mathbb{F}) < +\infty$. In general, $\mathbb{F} = \mathbb{F}_0 \cup \{0\}$, with $\mathbb{F}_0$ the floating point numbers excluding the zero.

We indicate with $fl(x)$ the *floating point representation* of the real number $x \in \mathbb{R}$.



♦ **Example 1.1** $x = \frac{1}{3} = 0.\overline{3} = 0.\underbrace{333\cdots3}_{\infty \text{ digits}}$, while $fl(x) = 0.\underbrace{333\cdots3}_{N \text{ digits}}$, with $N < +\infty$. ♦

The set $\mathbb{F}_0 = \mathbb{F}_0(\beta, t, L, U)$ is characterized by four parameters β, t, L, and U such that any real number $x \in \mathbb{F}_0$ can be written as:

$$x = (-1)^s \, m \beta^{e-t} = (-1)^s \, (a_1 a_2 \cdots a_t)_\beta \, \beta^{e-t},$$

where:

- β is the *base* (the numerical system);
- $m = (a_1 a_2 \cdots a_t)_\beta$ is the *mantissa* with t *number of digits* such that $0 < a_1 \leq \beta - 1$ and $0 \leq a_i \leq \beta - 1$ for $i = 2, \ldots, t$;
- $e \in \mathbb{Z}$ is the *exponent* such that $L \leq e \leq U$, with $L < 0$ and $U > 0$;
- $s = \{0, 1\}$ is the *sign*.

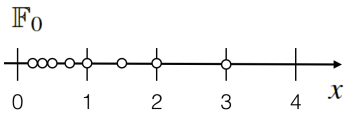
Once $\mathbb{F}_0(\beta,t,L,U)$ is characterized, $x \in \mathbb{F}_0$ is fully represented by s, m, and e. The minimum and maximum positive real numbers which can be represented at the calculator are $x_{min} = \beta^{L-1}$ and $x_{max} = \beta^U\left(1-\beta^{-t}\right)$, respectively.

Definition 1.2 The *epsilon machine* $\varepsilon_M := \beta^{1-t}$ is the minimum real number greater than zero such that $fl(1+\varepsilon_M) > 1$. The *roundoff error* $\frac{1}{2}\varepsilon_M$ is an upper bound of the relative error in the representation of a real number $x \in \mathbb{R}\setminus\{0\}$, i.e.:

$$\frac{|x-fl(x)|}{|x|} \leq \frac{1}{2}\varepsilon_M.$$

Remark 1.1 Even if the roundoff error $\frac{1}{2}\varepsilon_M$ is "small", i.e. the relative error is "small", the absolute error $|x-fl(x)|$ may be very "large", especially if $|x|$ is "large".

♦ **Example 1.2** We consider the set of floating point numbers $\mathbb{F}_0(2,2,-1,2)$, i.e. with $\beta = 2$ (numerical system in base 2), $t = 2$ (number of digits), $L = -1$, and $U = 2$. Then, we have $\varepsilon_M = \beta^{1-t} = \frac{1}{2}$, $x_{min} = \beta^{L-1} = \frac{1}{4}$, and $x_{max} = \beta^U\left(1-\beta^{-t}\right) = 3$. The values allowed for the exponent e are -1, 0, 1, and 2. The mantissa is $m = (a_1a_2)_\beta$ since $t = 2$; then, since $\beta = 2$, we have $a_1 = 1$, while a_2 is either 0 or 1. The values allowed for m are therefore $m = (10)_2 = 2$ or $(11)_2 = 3$. For the sign $s = 0$, the positive real numbers in $\mathbb{F}_0$ are $x = m\beta^{e-t} = m2^{e-2}$ as summarized in the following table.

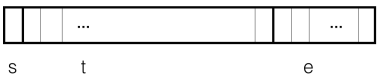


e	-1	0	1	2
$m = (10)^2 = 2$	$\frac{1}{4}$	$\frac{1}{2}$	1	2
$m = (11)^2 = 3$	$\frac{3}{8}$	$\frac{3}{4}$	$\frac{3}{2}$	3

♦

Remark 1.2 The larger is $|fl(x)|$, the lesser dense are the numbers in $\mathbb{R}$.

Remark 1.3 For 64–bit computing (CPUs) with base $\beta = 2$, 1 digit is reserved for the sign s, 52 digits are generally used for t, while 11 for the exponent e.



Remark 1.4 In MATLAB, for 64–bit CPUs, the number of digits t used for the mantissa m is actually $52+1 = 53$. Indeed, since the base $\beta = 2$ is used, the first digit a_1 is always equal to 1. Therefore, we have $\varepsilon_M = 2^{1-53} \simeq 2\cdot 10^{-16}$; in addition, we have $x_{min} \simeq 10^{-308}$, and $x_{max} \simeq 10^{308}$.

1.1.2 Floating point arithmetic

Algebraic operations on floating point numbers $\mathbb{F}$ do not enjoy the same properties of real numbers $\mathbb{R}$. *Round–off errors* may propagate and grow depending on the count and type of algebraic operations involved in the computations.

♦ **Example 1.3** For any $x \in \mathbb{R}\setminus\{0\}$, we have $\dfrac{(1+x)-1}{x} \equiv 1$. However, in floating point arithmetic $\dfrac{fl(1+fl(x))-1}{fl(x)} = y$, where y is a real number generally different than 1. If we try to verify the

first identity in MATLAB, we obtain that $y \neq 1$, with the following errors depending on the chosen value of x.

x	10^{-10}	10^{-14}	10^{-15}	10^{-16}
relative error	$8 \cdot 10^{-6}\%$	$8 \cdot 10^{-2}\%$	11%	100%

♦

The term *flops* is used to indicate the number of floating point operations.

1.2 Mathematical and Numerical Problems

We recall some basic notions of numerical approximation of mathematical problems.

1.2.1 The mathematical problem

Let us start by considering a *physical problem* (PP) endowed with a *physical solution*, symbolically indicated with x_{ph}, which depends on some data d. Then, the *mathematical problem* (MP) represents the mathematical formulation of the PP with the *mathematical solution* x. We indicate the MP as:

$$F(x;d) = 0, \tag{1.1}$$

with $x \in \mathscr{X}$ and $d \in \mathscr{D}$, where $\mathscr{X}$ and $\mathscr{D}$ are suitable spaces. The error between the physical and mathematical solutions is called *model error* $e_m := x_{ph} - x$.

♦ **Example 1.4** We consider as PP a body falling under the action of external forces, including the gravity, and as physical solution x_{ph} the velocity of the body at a given time $t_f > 0$. In order to define the associated MP, we recall the following model:

$$\text{find } V(t) \ : \ \begin{cases} m\dot{V}(t) = f_{ext}(t) & \text{for } t > 0, \\ V(0) = 0, \end{cases}$$

where $V(t)$ is the body velocity, m its mass, and $f_{ext}(t)$ the external forces. By identifying the mathematical solution x with $V(t_f)$, i.e. $x = V(t_f)$, we have the following MP:

$$F(x;d) = x - \int_0^{t_f} \frac{f_{ext}(t)}{m}\, dt = 0,$$

where the data are $d = \{t_f, m, f_{ext}(t)\}$. ♦

Remark 1.5 Before solving a MP, one needs to ensure that it is *well–posed*.

Before introducing the notion of well–posedness of a MP, we recall the following definition.

Definition 1.3 The solution $x \in \mathscr{X}$ of the MP $F(x;d) = 0$ is *continuously dependent on the data* $d \in \mathscr{D}$ if and only if, for all δd such that $(d + \delta d) \in \mathscr{D}$ and δx such that $F(x + \delta x; d + \delta d) = 0$, there exist two constants $\eta_0 = \eta_0(d) > 0$ and $K_0 = K_0(d)$ such that

$$\|\delta d\| \leq \eta_0 \implies \|x\| \leq K_0 \|\delta d\|, \tag{1.2}$$

with $\|\cdot\|$ a suitable norm.

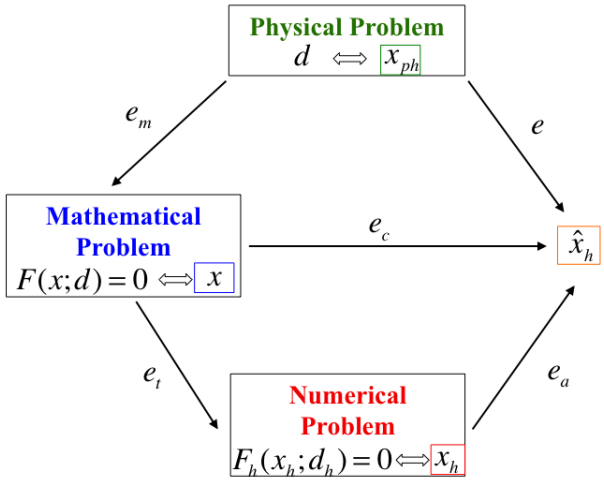


Figure 1.1: Schematic representation of the physical, mathematical, and numerical problems and their solutions.

♦ **Example 1.5** We consider the following MPs.

- $F(x;d) = x - \gamma d = 0$ for some $\gamma \in \mathbb{R}$ positive, $x \in \mathscr{X} \equiv \mathbb{R}$, and $d \in \mathscr{D} \equiv \mathbb{R}$. We have $F(x+\delta x; d+\delta d) = (x+\delta x) - \gamma(d+\delta d) = 0$, from which we obtain $\delta x = \gamma \delta d$. Hence, if we choose $\eta_0 = 1$ and $K_0 = \gamma$, the condition (1.2) is satisfied and $x \in \mathscr{X}$ is continuously dependent on $d \in \mathscr{D}$.
- $F(x;d) = x^2 - d = 0$ for $x \in \mathscr{X} \equiv \mathbb{R}$ and $d \in \mathscr{D} \equiv \mathbb{R}$. We observe that $x = \pm\sqrt{d}$ if $d \geq 0$, while $\pm\sqrt{|d|}\,i$ if $d < 0$; in this latter case, $x \in \mathbb{C}$, i.e. $x \notin \mathscr{X}$, for which x is not continuously dependent on $d \in \mathscr{D} \equiv \mathbb{R}$.

♦

Definition 1.4 The MP $F(x;d) = 0$ of Eq. (1.1) is *well–posed* (stable) if and only if there exists a unique solution $x \in \mathscr{X}$ which is continuously dependent on the data $d \in \mathscr{D}$.

MP which are formally well–posed may exhibits "large" variations of the solution x even for "small" changes of the data d. A measure of this sensitivity is given by the conditioning number of the MP.

Definition 1.5 The *relative conditioning number* of the MP $F(x;d) = 0$ for the data $d \in \mathscr{D}$ is:

$$K(d) := \sup_{\substack{\delta d \,:\, (d+\delta d)\in\mathscr{D} \\ \text{and } \|\delta d\|\neq 0}} \left\{ \frac{\|\delta x\|/\|x\|}{\|\delta d\|/\|d\|} \right\}.$$

Remark 1.6 The relative conditioning number of a MP is such that $K(d) \geq 1$ by definition. If $K(d)$ is "small", the MP is *well–conditioned*; if $K(d)$ is "large", the MP is *ill–conditioned*.

♦ **Example 1.6** We consider the MP $F(x;d) = dx - \alpha = 0$ for some $\alpha \in \mathbb{R}$, with $x \in \mathscr{X} \equiv \mathbb{R}$ and $d \in \mathscr{D} \equiv \mathbb{R}$. We have $F(x+\delta x; d+\delta d) = (d+\delta d)(x+\delta x) - \alpha = 0$, from which we obtain $\dfrac{\delta x}{x} = -\dfrac{d}{d+\delta d}\dfrac{\delta d}{d}$. We have $K(d) \simeq \sup\limits_{\substack{\delta d \,:\, (d+\delta d)\in\mathscr{D} \\ \text{and } \|\delta d\|\neq 0}} \left|\dfrac{d}{d+\delta d}\right|$, which can be "large" if $\delta d \simeq -d$. ♦

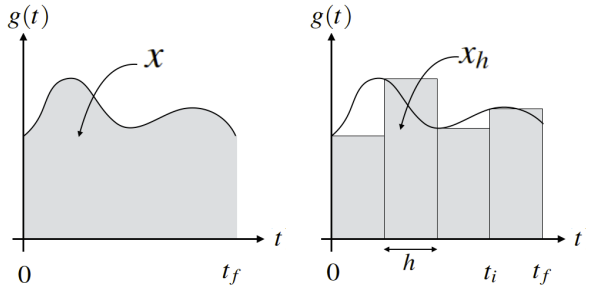
1.2.2 The numerical problem

The *numerical problem* (NP) is an approximation of the MP (1.1); we indicate its *numerical solution* as x_h, with h a suitable *discretization* parameter (in other instances, n is used). The error between the mathematical and numerical solutions is called *truncation error* $e_t := x - x_h$ (see also Fig. 1.1). We refer to the NP as:

$$F_h(x_h; d_h) = 0, \tag{1.3}$$

where $x_h \in \mathscr{X}_h$ and $d_h \in \mathscr{D}_h$, with $\mathscr{X}_h$ and $\mathscr{D}_h$ suitable spaces. The final solution $\widehat{x}_h$ is generally affected by the roundoff error $e_r := x_h - \widehat{x}_h$. The truncation and roundoff errors determine the *computational error* $e_c := x - \widehat{x}_h = e_t + e_r$. We remark that typically $|e_r| \ll |e_t|$, for which e_t is often identified with e_c.

♦ **Example 1.7** For the MP $F(x;d) = x - \int_0^{t_f} g(t)\,dt = 0$ with the data $d = \{t_f, g(t)\}$, we can consider the NP $F_h(x_h;d_h) = x_h - h\sum_{i=0}^{n-1} g(t_i) = 0$, where $t_i = ih$ for $i = 0,\ldots,n$, with $h = \dfrac{t_f}{n}$.



♦

Remark 1.7 As for the MP, also for the NP we need to ensure that it is well–posed.

Definition 1.6 The NP problem $F_h(x_h;d_h) = 0$ of Eq. (1.3) is *well–posed* (stable) if and only if there exists a unique solution $x_h \in \mathscr{X}_h$ which is continuously dependent on the data $d_h \in \mathscr{D}_h$.

Definition 1.7 The *relative conditioning number* of the NP $F_h(x_h;d_h) = 0$ for the data $d_h \in \mathscr{D}_h$ is:

$$K_h(d_h) := \sup_{\substack{\delta d_h \,:\, (d_h+\delta d_h)\in\mathscr{D}_h \\ \text{and } \|\delta d_h\|\neq 0}} \left\{ \frac{\|\delta x_h\|/\|x_h\|}{\|\delta d_h\|/\|d_h\|} \right\}.$$

For a NP the concept of *consistency* with the MP must be introduced.

Definition 1.8 The NP (1.3) is *consistent* if and only if $\lim_{h\to 0} F_h(x;d) = F(x;d) = 0$, with $d \in \mathscr{D}_h$.

Definition 1.9 The NP (1.3) is *strongly consistent* if and only if $F_h(x;d) \equiv F(x;d) = 0$ for all $h > 0$, with $d \in \mathscr{D}_h$.

♦ **Example 1.8** We consider two different NP associated to the MP $F(x;d) = x - d = 0$, with $d = \sqrt{2}$, for which $x = \sqrt{2}$.

- We define the NP $F_n(x_n;d) = x_{n+1} - \frac{3}{4}x_n - \frac{1}{2x_n} = 0$ for $n \geq 0$, with $x_0 = 1$; in this case n stands for the discretization parameter and indicates the iterate number. Since $F_n(x;d) = \sqrt{2} - \frac{3}{4}\sqrt{2} - \frac{1}{2\sqrt{2}} = 0$ for all $n \geq 0$, the NP is strongly consistent.
- We set now the NP $F_n(x_n;d) = x_{n+1} - \frac{3}{4}x_n - \frac{1}{2x_n} + \frac{1}{(1+n)^5} = 0$ for $n \geq 0$, with $x_0 = 1$. We observe that $F_n(x;d) = \frac{1}{(1+n)^5} \neq 0$ for $n \geq 0$, for which the NP is not strongly consistent. However, NP is consistent since $\lim_{n\to+\infty} F_n(x;d) = 0$.

♦

Another important concept of a NP is the *convergence*.

Definition 1.10 Let $x(d)$ be the mathematical solution of the MP $F(x(d);d) = 0$ of Eq. (1.1) and $x_h(d+\delta d_h)$ be the numerical solution of the NP $F_h(x_h(d+\delta d_h);d+\delta d_h) = 0$. Then, the NP (1.3) is *convergent* if and only if for all $\varepsilon > 0$, there exists $h_0 = h_0(\varepsilon) > 0$ and $\Delta = \Delta(h_0,\varepsilon)$ such that, for all $h < h_0$ and for all admissible δd_h for which $\|\delta d_h\| \leq \Delta$, the condition $\|x(d) - x_h(d+\delta d_h)\| \leq \varepsilon$ is satisfied.

Remark 1.8 If the NP is convergent, the computational error tends to zero, i.e. $\lim_{\substack{h\to 0 \\ (\text{or } n\to+\infty)}} e_c = 0$.

An important aspect related to the converge of a NP is the *convergence order*; with this aim, we redefine in the following the computational error e_c as $e_c = |x - \widehat{x}_h|$.

Definition 1.11 If the computational error $e_c \leq C h^p$, with C a positive constant independent of h and p, then the NP is *convergent* with *order* p.

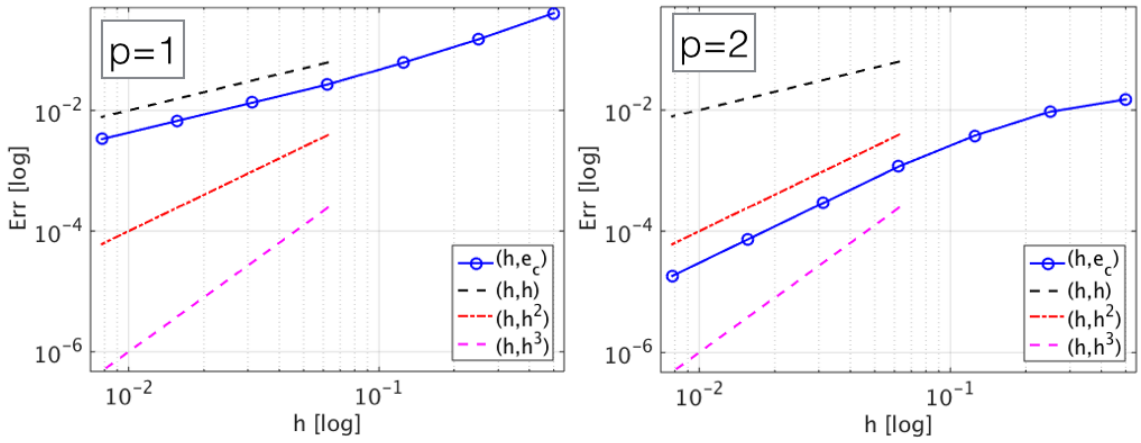
Remark 1.9 If there exists a positive constant $\widetilde{C} \leq C$ independent of h and p such that $\widetilde{C}h^p \leq e_c \leq Ch^p$, then we can write $e_c \simeq Ch^p$.

Remark 1.10 If one can assume that $e_c \simeq Ch^p$, then the convergence order p can be estimated in two manners by considering a MP for which the (exact) mathematical solution x is known.

- *Algebraically*. First, the errors e_{c1} and e_{c2} associated to two values of the discretization parameters h_1 and h_2 (which are sufficiently "small") are computed, respectively; then, by setting $e_{c1} \simeq Ch_1^p$ and $e_{c2} \simeq Ch_2^p$ and observing that $\frac{e_{c1}}{e_{c2}} = \left(\frac{h_1}{h_1}\right)^p$, p is estimated as:

$$p = \frac{\log(e_{c1}/e_{c2})}{\log(h_1/h_2)}.$$

- *Graphically*. The errors e_c computed for different values of h are plotted vs. h in *log–log scales*. Since $\log e_c = \log(Ch^p) = \log C + p\log h$, we have $p = \text{atan}(\theta)$, where θ is the slope of the (h,e_c) curve, which is a straight line in log–log scales. Instead of computing the angle θ, one can graphically verify if the curves (h,e_c) and (h,h^p) are *parallel* in log–log scales.



Remark 1.11 A NP must be well–posed (well–conditioned), consistent, and convergent.

Theorem 1.1 — Lax–Richtmeyer, equivalence. If the NP $F_h(x_h; d_h) = 0$ for $x_h \in \mathscr{X}_h$ and $d_h \in \mathscr{D}_h$ is consistent, then it is well–posed if and only if it is convergent (i.e. $x_h \to x$).

The equivalence theorem is useful since it allows to verify only two of the required properties of a NP problem to yield the third one; specifically, we observe that in general it is "easy" to show the consistency of a NP, while it may be "difficult" to show its well–posedness and/or convergence.

Remark 1.12 Following the equivalence theorem, if the NP is consistent and well–posed then it is also convergent; similarly, if the NP is consistent and convergent then it is also well–posed.

1.2.3 Choice of a numerical method

The choice of a numerical method (NP) to approximate the solution x of a MP should take into account for:

- the (mathematical) properties of the MP;
- the computational efficiency in terms of: expected convergence order of the error, flops involved in the computation, CPU available, memory access and storage.

Remark 1.13 If m indicates the size of the NP, the flops may depend on m in different manners, according to the following Table.

	$O(1)$	$O(m)$	$O(m^\gamma)$	$O(\gamma^m)$	$O(m!)$
flops	independent	linear	polynomial	exponential	factorial

♦ **Example 1.9** For $A \in \mathbb{R}^{m\times m}$, the computation of $\det(A)$ by means of the Cramer rule involves approximately $O(m!)$ flops. The estimated times associated to the computation of $\det(A)$ for matrices of size m by means of a calculator with a $1\,GHz = 10^9$ flops/s CPU are reported below.

m	5	10	15	20
$m!$	120	$\sim 10^6$	$\sim 10^{12}$	$\sim 10^{18}$
CPU time	$\sim 10^{-7}$ s	$\sim 10^{-3}$ s	~ 30 min	~ 77 years

♦

Corrections to Chapter 1

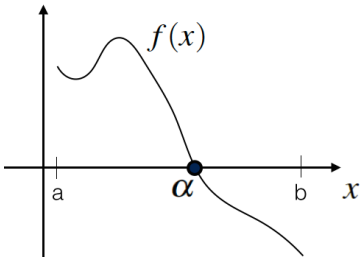
- Equation (1.2), it should be written

$$\|\delta d\| \leq \eta_0 \implies \|\delta x\| \leq K_0 \|\delta d\|.$$

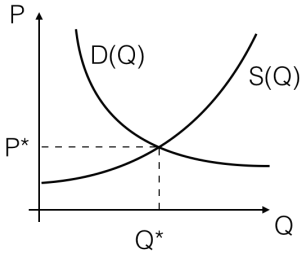
- The first sentence in Remark 1.6 is false. The condition number can be smaller than one. (The condition number for matrices, defined in Chapter 5, *cannot* be smaller than one).

2. Nonlinear Equations

The goal is to *numerically approximate* the *zero* $\alpha \in \mathbb{R}$ of a function $f(x)$ in the interval $I = (a,b) \subseteq \mathbb{R}$. The problem is also commonly referred as the numerical solution of a *nonlinear equation*.



♦ **Example 2.1** *Supply and demand*: microeconomic model of price determination of a good in a competitive market. The unit price P of a good varies until an equilibrium between supply and demand quantities (Q) is met.



$P = S(Q)$ is the function representing the supply of the good; the quantity of the good increases if the price increases. $P = D(Q)$ is the demand function; the demand of the good increases if its price decreases. (Q^*, P^*) is the equilibrium point for which $P^* = S(Q^*) = D(Q^*)$; if $x = Q$, one needs to solve the nonlinear equation $f(x) = S(x) - D(x) = 0$.

♦

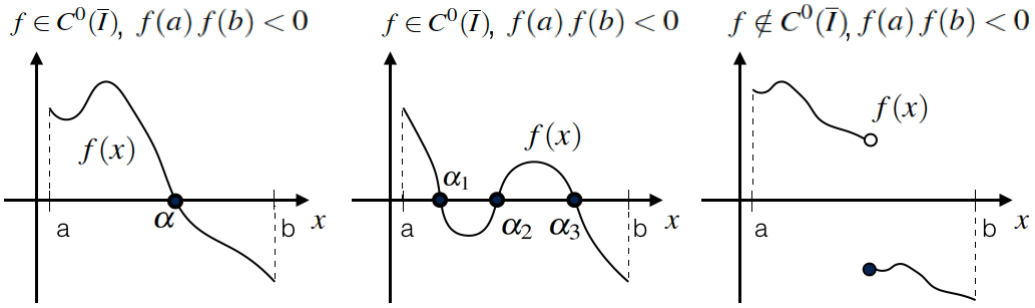
2.1 Bisection Method

We consider the *bisection method* for the approximation of the zero $\alpha \in I$ of a function $f(x)$.

2.1.1 Foundation of the bisection method

Theorem 2.1 — Zeros of a continuous function. Let $f(x)$ be a continuous function in the interval $I = (a,b)$, that is $f \in C^0(\bar{I}) \equiv C^0([a,b])$. If $f(a)\,f(b) < 0$, then there exists *at least* a zero $\alpha \in I$ of the function $f(x)$.

♦ **Example 2.2** We illustrate some examples for functions $f(x)$ such that $f(a)\,f(b) < 0$.



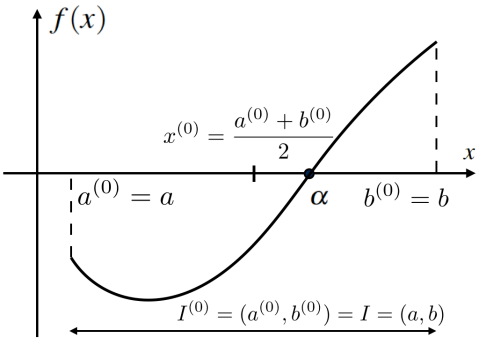
♦

Let us assume that there exists an *unique* zero $\alpha \in I$ of a function $f \in C^0(\bar{I})$ such that $f(a)\,f(b) < 0$. Then, the bisection method searches the zero α by recursively approximating it with the sequence of *mid–points* of the subintervals $I^{(k)}$ of I for which the function $f(x)$ features changes of sign.

♦ **Example 2.3** We illustrate the bisection method and algorithm in the following pictures.

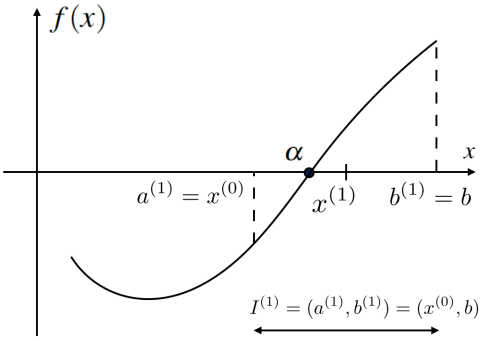
Step 0.

$I^{(0)} = \left(a^{(0)}, b^{(0)}\right) = I = (a,b)$ and
$x^{(0)} = \dfrac{a^{(0)} + b^{(0)}}{2} = \dfrac{a+b}{2}$



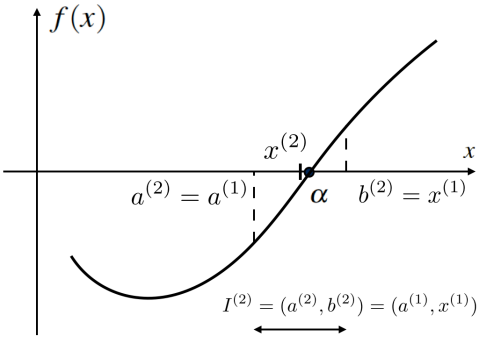
Step 1.

Since $f\left(x^{(0)}\right) f\left(b^{(0)}\right) < 0$:
$a^{(1)} = x^{(0)}$, $b^{(1)} = b$,
$I^{(1)} = (a^{(1)}, b^{(1)}) = (x^{(0)}, b)$, and
$x^{(1)} = \dfrac{a^{(1)} + b^{(1)}}{2} = \dfrac{x^{(0)} + b}{2}$.



Step 2.

Since $f\left(x^{(1)}\right) f\left(a^{(1)}\right) < 0$:
$a^{(2)} = a^{(1)}$, $b^{(2)} = x^{(1)}$,
$I^{(2)} = (a^{(2)}, b^{(2)}) = (a^{(1)}, x^{(1)})$, and
$x^{(2)} = \dfrac{a^{(2)} + b^{(2)}}{2} = \dfrac{a^{(1)} + x^{(1)}}{2}$.



♦

2.1.2 Algorithm and properties

We report the algorithm and the numerical properties of the bisection method.

Algorithm 2.1: Bisection method

set $k = 0$, $a^{(0)} = a$, $b^{(0)} = b$, and $x^{(0)} = \dfrac{a^{(0)} + b^{(0)}}{2}$;
for $k = 1, 2, \ldots$, *until a stopping criterion is satisfied* **do**
  if $f\left(x^{(k-1)}\right) = 0$ **then**
    set $\alpha = x^{(k-1)}$ and terminate the loop;
  else
    if $f\left(x^{(k-1)}\right) f\left(a^{(k-1)}\right) < 0$ **then**
      set $a^{(k)} = a^{(k-1)}$ and $b^{(k)} = x^{(k-1)}$;
    end
    if $f\left(x^{(k-1)}\right) f\left(b^{(k-1)}\right) < 0$ **then**
      set $a^{(k)} = x^{(k-1)}$ and $b^{(k)} = b^{(k-1)}$;
    end
    set $x^{(k)} = \dfrac{a^{(k)} + b^{(k)}}{2}$;
  end
end

Remark 2.1 For the subinterval $I^{(k)} = \left(a^{(k)}, b^{(k)}\right))$ and its midpoint $x^{(k)} = \dfrac{a^{(k)} + b^{(k)}}{2}$ we have that both $x^{(k)}$ and $\alpha \in I^{(k)}$ for all $k \geq 0$. Moreover, since $\left|I^{(k)}\right| := b^{(k)} - a^{(k)} \equiv \dfrac{\left|I^{(k-1)}\right|}{2}$ for all $k \geq 1$, we have:

$$\left|I^{(k)}\right| = \frac{\left|I^{(0)}\right|}{2^k} = \frac{b-a}{2^k} \qquad \text{for all } k \geq 0.$$

Let us indicate the (computational) *error* associated to the bisection method as $e^{(k)} := \left|x^{(k)} - \alpha\right|$.

Remark 2.2 The error $e^{(k)} = \left|x^{(k)} - \alpha\right|$ can be bounded from above by the size of the subinterval $I^{(k+1)}$ for all $k \geq 0$ which plays the role of *error bound*, also known as *error estimator*. We have:

$$e^{(k)} \leq \widetilde{e}^{(k)} := \left|I^{(k+1)}\right| = \frac{b-a}{2^{k+1}} \qquad \text{for all } k \geq 0. \tag{2.1}$$

This implies that the bisection method is *convergent*; indeed $\lim_{k \to +\infty} e^{(k)} = 0$ since $e^{(k)} \leq \widetilde{e}^{(k)}$ for all $k \geq 0$ and $\lim_{k \to +\infty} \widetilde{e}^{(k)} = \lim_{k \to +\infty} \dfrac{b-a}{2^{k+1}} = 0$.

Remark 2.3 Given a tolerance $tol > 0$, one can compute the *minimum number* of *iterations* of the bisection method, say k_{min}, ensuring that the error $e^{(k_{min})}$ is smaller than tol, i.e. $e^{(k_{min})} < tol$.

Indeed, from Eq. (2.1) we have that k_{min} is the smallest integer number such that $\frac{b-a}{2^{k_{min}+1}} < tol$, for which $k_{min} > \log_2\left(\frac{b-a}{tol}\right) - 1$.

We already know that the bisection method is convergent. However, we aim at characterizing such convergence. With this aim, we provide the following definition.

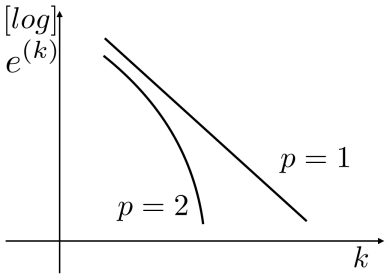
Definition 2.1 An iterative method for the approximation of the zero α of the function $f(x)$ is *convergent* with *order* p if and only if

$$\lim_{k\to+\infty} \frac{\left|x^{(k+1)}-\alpha\right|}{\left|x^{(k)}-\alpha\right|^p} = \mu, \tag{2.2}$$

with $\mu > 0$ a real number independent of k, which is called *asymptotic convergence factor*. In the case of *linear* convergence, i.e. for $p = 1$, we need $0 < \mu < 1$.

♦ **Example 2.4** We report in the following a typical plot of the sequence of errors $e^{(k)}$ vs. the iteration number k for hypothetical iterative methods exhibiting convergence orders $p = 1$ and 2.

The *logarithmic scale* is used on the error axis, while the linear scale on the axis of the iteration number. We notice that the *linear* convergence ($p = 1$) is graphically represented by a straight line, whose slope depends on the asymptotic convergence factor μ. A parabola is obtained instead for the *quadratic* convergence ($p = 2$).



♦

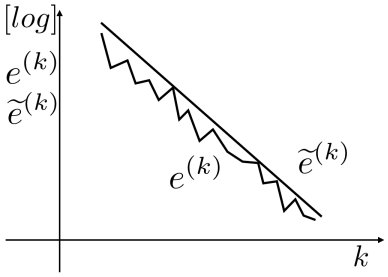
Remark 2.4 For the bisection method the error may not be monotonically convergent, i.e. it is possible that $e^{(k+1)}) \geq e^{(k)}$ for some $k \geq 0$; therefore, even if the bisection method is convergent, a convergence order can not be established according to Eq. (2.2). Similarly, the *absolute residual* $r^{(k)} := \left|f\left(x^{(k)}\right)\right|$ is not monotonically decreasing, in general.

Remark 2.5 For the bisection method, the sequence of error estimators $\left\{\widetilde{e}^{(k)}\right\}$ is *linearly convergent* according to Eq. (2.2) with $p = 1$ and $\mu = \frac{1}{2}$; indeed:

$$\frac{\widetilde{e}^{(k+1)}}{\widetilde{e}^{(k)}} = \frac{(b-a)/2^{k+2}}{(b-a)/2^{k+1}} = \frac{1}{2} \qquad \text{for all } k \geq 0.$$

♦ **Example 2.5** We highlight the typical behavior of the sequence of errors $e^{(k)}$ and error estimators $\widetilde{e}^{(k)}$ vs. the iteration number k obtained for the *bisection method*.

The *logarithmic scale* is used on the error axis, while the linear scale on the axis of the iteration number. Following Remarks 2.4 and 2.5 we graphically highlight that a convergence order cannot be established for the error, while the convergence is linear for the error estimator.



♦

2.1.3 Stopping criterion

The *stopping criterion* of the bisection algorithm is based on the *error estimator* (bound) $\widetilde{e}^{(k)}$ of Eq. (2.1). In Algorithm 2.1, we considered the following stopping criterion in pseudocode (in place of the *for* loop) by indicating with tol a prescribed tolerance and k_{max} the maximum number of iterations allowed.

Algorithm 2.2: Bisection method. Stopping criterion

```
...;
while (ẽ^(k) ≥ tol & k < k_max) do
    ...;
end
```

2.2 Newton Method

We consider the Newton method and its variants for approximating the zero α of the function $f(x)$.

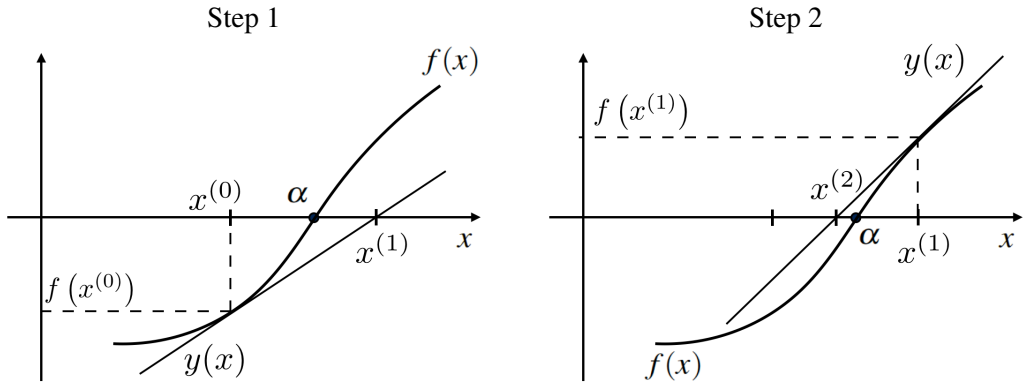
2.2.1 Newton method

Let us assume that $f \in C^0(I)$ and is *differentiable* in the interval $I = (a,b) \subseteq \mathbb{R}$. Given a generic iterate $x^{(k)} \in I$, the equation of the tangent line to the curve $(x, f(x))$ at the coordinate $x^{(k)}$ is $y(x) = f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(x - x^{(k)}\right)$. If we assume that $y\left(x^{(k+1)}\right) = 0$, then we compute the iterate $x^{(k+1)}$ as:

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)} \qquad \text{for all } k \geq 0, \tag{2.3}$$

given some *initial guess* $x^{(0)}$ and provided that $f'\left(x^{(k)}\right) \neq 0$ for all $k \geq 0$. Eq. (2.3) is named *Newton iterate*. One obtains the zero α as the limit of the sequence of iterates $\left\{x^{(k+1)}\right\}_{k=0}^{+\infty}$ which solve the tangent equation to the curve $(x, f(x))$ in $\left\{x^{(k)}\right\}_{k=0}^{+\infty}$, respectively.

♦ **Example 2.6** We graphically highlight the Newton method in the following pictures by reporting the first two Newton iterates.



♦

In summary, the *Newton method* is applicable to a function $f \in C^0(I)$ which differentiable in I; then, given $x^{(0)} \in I$, the Newton method consists in sequentially applying the Newton iterate (2.3), provided that $f'\left(x^{(k)}\right) \neq 0$ for all $k \geq 0$.

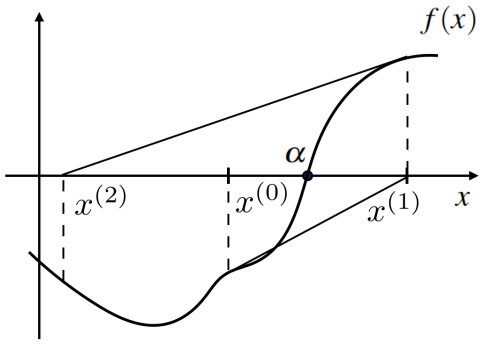
Algorithm 2.3: Newton method

set $k = 0$ and the initial guess $x^{(0)}$;
while *(stopping criterion is false)* **do**
  $x^{(k+1)} = x^{(k)} - \dfrac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}$;
  set $k = k+1$;
end

Remark 2.6 Let us assume that $f \in C^2(I)$, then the *Taylor* expansion of $f(x)$ around $x^{(k)}$ reads $f\left(x^{(k+1)}\right) = f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\delta^{(k)} + O\left((\delta^{(k)})^2\right)$, where $\delta^{(k)} := x^{(k+1)} - x^{(k)}$ for $k \geq 0$ is the *difference of successive iterates*. If $f\left(x^{(k+1)}\right) = 0$, then the Newton method represents the first order approximation of the Taylor expansion of $f(x)$ around $x^{(k)}$; we observe that in order to satisfy this assumption, one needs $\delta^{(k)}$ to be "small".

Remark 2.7 The choice of the *initial guess* $x^{(0)}$ is crucial for the success of the Newton method. Indeed, one needs to choose $x^{(0)}$ *"sufficiently" close* to the zero α. As a matter of fact, the sequence of Newton iterates $\left\{x^{(k+1)}\right\}_{k=0}^{+\infty}$ may diverge, instead of converging to α, if the initial guess $x^{(0)}$ is *not* "sufficiently" close to the zero α. Since α is unknown, the choice of $x^{(0)}$ may not be trivial; in this respect, the plot of the function or the *bisection* method can be used to select $x^{(0)}$ "sufficiently" close to α.

♦ **Example 2.7** The following example illustrates that the Newton iterates are not converging to the zero α, due to the fact that $x^{(0)}$ is not "sufficiently" close to α.



♦

Remark 2.8 For (linear) affine functions in the form $f(x) = cx + d$ with c and $d \in \mathbb{R}$, the Newton method converges to the zero $\alpha = -\dfrac{d}{c}$ in 1 iteration regardless of the choice of $x^{(0)}$. Indeed, we have from Eq. (2.3) that $x^{(1)} = x^{(0)} - \dfrac{f\left(x^{(0)}\right)}{f'\left(x^{(0)}\right)} = x^{(0)} - \dfrac{cx^{(0)} + d}{c} = -\dfrac{d}{c} = \alpha$ for all $x^{(0)} \in \mathbb{R}$.

We characterize in the following the convergence properties of the Newton method.

Proposition 2.2 — Convergence of the Newton method. If $f \in C^1(I)$, $x^{(0)}$ is "sufficiently" close to $\alpha \in I$, and $f'(\alpha) \neq 0$, then the Newton method is *convergent* to α, provided that $f'\left(x^{(k)}\right) \neq 0$ for all $k \geq 0$.

Proposition 2.3 — Convergence order of the Newton method. Let us indicate with I_α a neighborhood of α. If $f \in C^2(I_\alpha)$, $x^{(0)}$ is "sufficiently" close to α, and $f'(\alpha) \neq 0$, then the Newton method is *convergent with order* 2 (quadratically) to α, provided that $f'\left(x^{(k)}\right) \neq 0$ for all $k \geq 0$. In particular, we have:

$$\lim_{k \to +\infty} \frac{x^{(k+1)} - \alpha}{\left(x^{(k)} - \alpha\right)^2} = \frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)};$$

following Eq. (2.2), $p = 2$ is the convergence order and $\mu = \frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)}$ the asymptotic convergence factor.

Proof. The proof is based on the interpretation of the Newton method as a fixed point iterations method; see Sec. 2.3.5. ∎

Definition 2.2 Let $f \in C^m(I_\alpha)$, with $m \in \mathbb{N}$ such that $m \geq 1$. The *zero* $\alpha \in I_\alpha$ is said to be of *multiplicity* m if $f^{(i)}(\alpha) = 0$ for all $i = 0, \ldots, m-1$ and $f^{(m)}(\alpha) \neq 0$. If the previous condition is satisfied for $m = 1$, the zero α is called *simple*, otherwise is *multiple*.

Proposition 2.4 — Convergence order of the Newton method, zero multiple. If $f \in C^2(I_\alpha) \cap C^m(I_\alpha)$ and $x^{(0)}$ is "sufficiently" close to the zero α of multiplicity $m > 1$, then the Newton method is *convergent with order* 1 (linearly) to α, provided that $f'\left(x^{(k)}\right) \neq 0$ for all $k \geq 0$. In particular, following Eq. (2.2), we have:

$$\lim_{k \to +\infty} \frac{\left|x^{(k+1)} - \alpha\right|}{\left|x^{(k)} - \alpha\right|} = \mu,$$

with $p = 1$ the convergence order and $\mu \in (0, 1)$ the asymptotic convergence factor.

Remark 2.9 If the zero α is simple $m = 1$, the Newton method converges at least quadratically according to Proposition 2.3. Conversely, if the zero α is multiple ($m > 1$), the Newton method only converges linearly according to Proposition 2.4. We observe that, generally, the higher is the convergence order, fewer iterations are necessary to reach a prescribed value of the error, i.e. the method results to be more efficient.

Examples of linear and quadratic convergence are graphically highlighted in Example 2.4.

2.2.2 Modified Newton method

Let us assume that $f \in C^m(I_\alpha)$, with $\alpha \in I_\alpha$ and $m \geq 1$ the multiplicity of α. The k–the iterate of the *modified Newton method* reads:

$$x^{(k+1)} = x^{(k)} - m \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)} \qquad \text{for all } k \geq 0, \tag{2.4}$$

given the initial guess $x^{(0)}$ and provided that $f'\left(x^{(k)}\right) \neq 0$ for all $k \geq 0$. Following Algorithm 2.3, we have the following one for the modified Newton method.

Algorithm 2.4: Modified Newton method

choose m;
set $k = 0$ and the initial guess $x^{(0)}$;
while *(stopping criterion is false)* **do**
  $x^{(k+1)} = x^{(k)} - m\dfrac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}$;
  set $k = k+1$;
end

Remark 2.10 The modified Newton method requires the a priori knowledge of the multiplicity m of the zero α. The latter can be eventually estimated by means of suitable numerical approaches.

The convergence properties of the modified Newton method are characterized by the following Proposition.

Proposition 2.5 — Convergence order of the modified Newton method. If $f \in C^2(I_\alpha) \cap C^m(I_\alpha)$, with $m \geq 1$ the multiplicity of the zero $\alpha \in I_\alpha$, and $x^{(0)}$ is "sufficiently" close to α, then the modified Newton method is *convergent with order* 2 (quadratically) to α, provided that $f'\left(x^{(k)}\right) \neq 0$ for all $k \geq 0$.

♦ **Example 2.8** We approximate the zero $\alpha = 0$ of the function $f(x) = \sin^m(x)$ in the interval $I = \left(-\dfrac{\pi}{2}, \dfrac{\pi}{2}\right)$, with $m = 1, 2, 3, \ldots$; with this aim we consider the Newton and modified Newton methods. We observe that $f'(x) = m\sin^{m-1}(x)\cos(x)$, for which $f'(\alpha) = 1$ if $m = 1$ and $f'(\alpha) = 0$ for $m \geq 2$; the zero α is simple for $m = 1$, but multiple (m times) for $m \geq 2$. If we set the initial guess $x^{(0)} = \dfrac{\pi}{6}$, the first iterate of the Newton method yields $x^{(1)} = \dfrac{\pi}{6} - \dfrac{1}{\sqrt{3}m}$, for which $x^{(1)}$ is farther and farther from α (and closer and closer to $x^{(0)}$), the larger is m. Conversely, for the modified Newton method we have from Eq. (2.4) that $x^{(1)} = \dfrac{\pi}{6} - \dfrac{1}{\sqrt{3}}$, regardless of the value of $m \geq 1$. ♦

2.2.3 Stopping criterion for Newton method

We consider different stopping criterion for the Newton method and its variants. Since the zero α is in general unknown, the error $e^{(k)} = \left|x^{(k)} - \alpha\right|$ is also unknown; therefore, we need a suitable *error estimator* (error indicator) $\widetilde{e}^{(k)}$ such that $\widetilde{e}^{(k)} \simeq e^{(k)}$. By referring e.g. to the Newton and modified Newton Algorithms 2.3 and 2.4, the iterations are stopped for $k = k_{min}$ such that $\widetilde{e}^{(k_{min})} < tol$, with tol a prescribed tolerance, or when the maximum number of iterations is reached; see e.g. Algorithm 2.2.

First, we consider the criterion based on the *difference of successive iterates*, for which the error estimator is chosen as:

$$\widetilde{e}^{(k)} = \begin{cases} \left|\delta^{(k-1)}\right| & \text{if } k \geq 1 \\ tol + 1 & \text{if } k = 0 \end{cases} \qquad \text{with } \delta^{(k)} := x^{(k+1)} - x^{(k)} \qquad \text{for } k \geq 0.$$

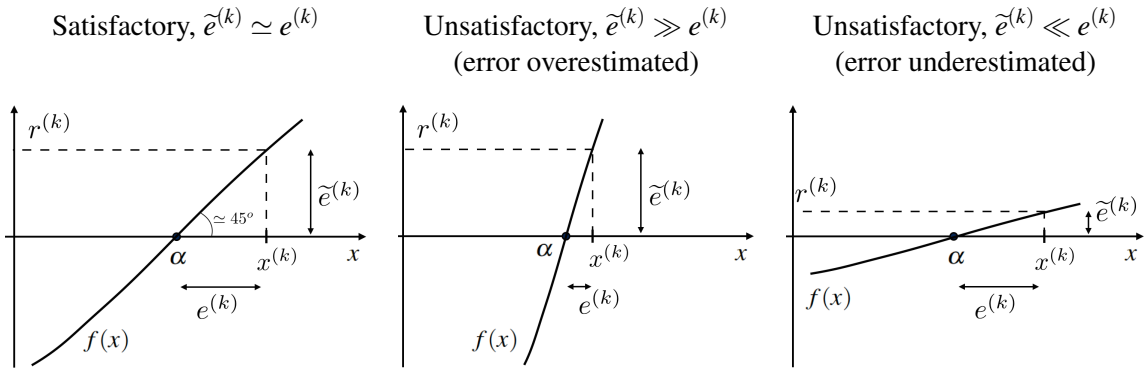
The criterion is *satisfactory* if the zero α is simple; this can be shown by interpreting the Newton method as a fixed point iterations method; see Sec. 2.3.5.

Another criterion is based on the absolute *residual*, for which:

$$\widetilde{e}^{(k)} = \left| r^{(k)} \right| \qquad \text{with } r^{(k)} := f\left(x^{(k)}\right) \qquad \text{for } k \geq 0.$$

This criterion is *satisfactory* if $\left|f'(x)\right| \simeq 1$ for $x \in I_\alpha$ a neighborhood of α, for which $\widetilde{e}^{(k)} \simeq e^{(k)}$. Conversely, the criterion is *unsatisfactory* if $\left|f'(x)\right| \gg 1$ or $\left|f'(x)\right| \simeq 0$ for $x \in I_\alpha$. Specifically, if $\left|f'(x)\right| \gg 1$ for $x \in I_\alpha$, the error is *overestimated* by the error estimator ($\widetilde{e}^{(k)} \gg e^{(k)}$), for which more Newton iterations than necessary are performed; therefore, the stopping criterion is inefficient. If $\left|f'(x)\right| \simeq 0$ for $x \in I_\alpha$, the error is *underestimated* by the error estimator ($\widetilde{e}^{(k)} \ll e^{(k)}$), for which the Newton iterations are prematurely stopped since the error is larger than predicted by the estimator.

♦ **Example 2.9** The following examples graphically illustrate the situations for which the criterion based on the *residual* is satisfactory or unsatisfactory.



♦

2.2.4 Inexact and quasi–Newton methods

The Newton and modified Newton methods require the evaluation of the first derivative of the function $f(x)$; see Eqs. (2.3) and (2.4). However, in some cases of practical interest, the evaluation of $f'(x)$ may be "difficult" or computationally expensive. Therefore, by referring e.g. to the Newton iterate (2.3), $f'\left(x^{(k)}\right)$ can be approximated by a computationally feasible quantity $q^{(k)} \simeq f'\left(x^{(k)}\right)$. *Inexact* or *quasi–Newton methods* are based on the use of an approximate value of $f'\left(x^{(k)}\right)$. The general quasi–Newton iterate reads:

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{q^{(k)}} \qquad \text{for all } k \geq 0,$$

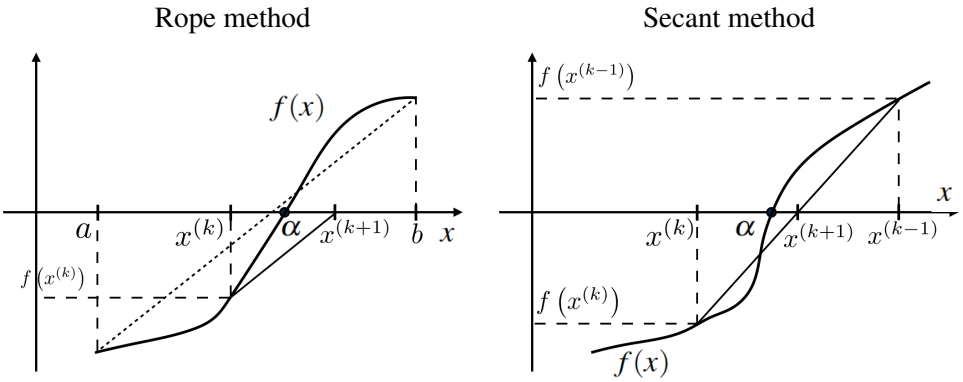
with $q^{(k)}$ determining the method. We consider the following cases:

- for $q^{(k)} \equiv f'\left(x^{(k)}\right)$, we obtain the standard Newton method;
- for $q^{(k)} = \dfrac{f(b)-f(a)}{b-a}$ for all $k \geq 0$, with $\alpha \in (a,b)$, we have the *rope method*;
- for $q^{(k)} = \dfrac{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}{x^{(k)} - x^{(k-1)}}$ for all $k \geq 1$, we have the *secant method*[1].

We observe that the secant method converges with order $p = 1.6$ if the zero α is simple.

[1] For the secant method, $q^{(0)}$ can be set as for the rope method.

♦ **Example 2.10** The following examples graphically illustrate the *rope* and *secant* methods at the general iterate $x^{(k)}$.



♦

2.2.5 Newton method for systems of nonlinear equations

The Newton method can be used to approximate the solution of systems of nonlinear equations. Given $\mathbf{F} : \mathbb{R}^d \to \mathbb{R}^d$, for some $d \geq 1$, the problem consists in finding the vector $\boldsymbol{\alpha} \in \mathbb{R}^d$ such that $\mathbf{F}(\boldsymbol{\alpha}) = \mathbf{0}$. More specifically, we have:

$$\mathbf{x} = \begin{Bmatrix} x_1 \\ \vdots \\ x_d \end{Bmatrix} \quad \text{and} \quad \mathbf{F}(\mathbf{x}) = \begin{Bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_d(\mathbf{x}) \end{Bmatrix} = \begin{Bmatrix} f_1(x_1,\ldots,x_d) \\ \vdots \\ f_d(x_1,\ldots,x_d) \end{Bmatrix}.$$

Definition 2.3 Let $\mathbf{F} : \mathbb{R}^d \to \mathbb{R}^d$ be differentialble in $I_{\mathbf{x}} \subseteq \mathbb{R}^d$ a neighborhood of $\mathbf{x} \in \mathbb{R}^d$, then its *Jacobian* in $\mathbf{x}$ is $J_{\mathbf{F}} : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ such that $(J_{\mathbf{F}}(\mathbf{x}))_{ij} = \dfrac{\partial f_i}{\partial x_j}(\mathbf{x})$ for $i, j = 1,\ldots,d$.

The *Newton method* is applicable to a system of equations $\mathbf{F} \in C^0(I_{\boldsymbol{\alpha}})$ which is differentiable in $I_{\boldsymbol{\alpha}} \subseteq \mathbb{R}^d$, a neighborhood of $\boldsymbol{\alpha}$; then, given $x^{(0)} \in I_{\boldsymbol{\alpha}}$, the Newton method consists in sequentially applying the following Newton iterate:

$$\text{solve } J_{\mathbf{F}}\left(\mathbf{x}^{(k)}\right)\boldsymbol{\delta}^{(k)} = -\mathbf{F}\left(\mathbf{x}^{(k)}\right) \quad \text{and} \quad \text{set } \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}^{(k)} \qquad \text{for all } k \geq 0, \tag{2.5}$$

provided that $\det\left(J_{\mathbf{F}}\left(\mathbf{x}^{(k)}\right)\right) \neq 0$ for all $k \geq 0$. The Newton algorithm using the stopping criterion based on the difference of successive iterates is reported in the following.

Remark 2.11 At each Newton iterate, one needs to solve a linear system, unless $d = 1$ for which $J_{\mathbf{F}}\left(x^{(k)}\right) \equiv f'\left(x^{(k)}\right)$. We also observe that the Newton iterate (2.5) can be obtained by the first order expansion of $\mathbf{F}(\mathbf{x})$ around $\mathbf{x}^{(k)}$ as $\mathbf{F}\left(\mathbf{x}^{(k)}\right) + J_{\mathbf{F}}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) = \mathbf{0}$.

Regarding the convergence of the Newton method for systems of nonlinear equations, we state the following.

Proposition 2.6 If $\mathbf{F} \in C^2(I_{\boldsymbol{\alpha}})$, with $I_{\boldsymbol{\alpha}} \subseteq \mathbb{R}^d$ a neighborhood of $\boldsymbol{\alpha}$, $\mathbf{x}^{(0)} \in \mathbb{R}^d$ is "sufficiently" close to $\boldsymbol{\alpha}$, and $\det\left(J_{\mathbf{F}}(\boldsymbol{\alpha})\right) \neq 0$, then the Newton method converges with order $p = 2$.

♦ **Example 2.11** Let us consider the system of nonlinear equations $\mathbf{F}(\mathbf{x}) = \begin{Bmatrix} \sin(x_1x_2)+x_2 \\ x_1+x_2-\frac{1}{2}e^{-x_1x_2} \end{Bmatrix}$, with the zero $\boldsymbol{\alpha} = \begin{Bmatrix} \frac{1}{2} \\ 0 \end{Bmatrix}$. Its Jacobian is $J_{\mathbf{F}}(\mathbf{x}) = \begin{bmatrix} x_2\cos(x_1x_2) & x_1\cos(x_1x_2)+1 \\ 1+\frac{x_2}{2}e^{-x_1x_2} & 1+\frac{x_1}{2}e^{-x_1x_2} \end{bmatrix}$, for which $\det(J_{\mathbf{F}}(\boldsymbol{\alpha})) = -\frac{3}{2} \neq 0$. ♦

2.3 Fixed Point Iterations

We consider the fixed point iterations method both to find the fixed point of an iteration function, as well as to solve nonlinear equations.

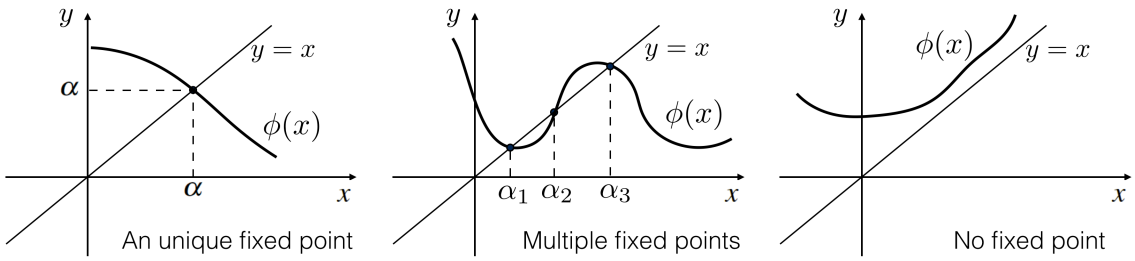
2.3.1 Nonlinear equations, zeros, fixed points, and iteration functions

Given a function $f : \mathbb{R} \to \mathbb{R}$, we aim at finding the zero α (i.e. such that $f(\alpha) = 0$). With this goal, we can transform the problem of finding the zero α into a *fixed point iterations problem*.

Definition 2.4 Given the *iteration function* $\phi : [a,b] \subseteq \mathbb{R} \to \mathbb{R}$, we say that $\alpha \in \mathbb{R}$ is a *fixed point* of ϕ if and only if $\phi(\alpha) = \alpha$.

♦ **Example 2.12** For the iteration function $\phi(x) = \cos(x)$ in the interval $[0.1, 1.1]$, we have the fixed point $\alpha = \cos(\alpha) \simeq 0.7391$. ♦

♦ **Example 2.13** We graphically illustrate the fixed points of some iteration functions.



♦

Remark 2.12 The goal is to find the zero α of the nonlinear function $f(x)$. We transform this problem in a fixed point iterations problem by suitably choosing a fixed point iterations function $\phi(x)$ such that $f(\alpha) = 0$ if and only if $\phi(\alpha) = \alpha$ for $\alpha \in [a,b]$. We remark that there are different iterations functions $\phi(x)$ and multiple manners to obtain them in order to achieve this goal.

♦ **Example 2.14** The simplest manner to obtain $\phi(x)$ from $f(x)$ is based on the following steps. Since $f(\alpha) = 0$, we have $f(\alpha)+\alpha = \alpha$, for which we can set $\phi(x) = f(x)+x$. We remark that this is very often not a "good" choice for the iteration function. ♦

♦ **Example 2.15** We consider $f(x) = 2x^2 - x - 1$ for which we are interested in the zero $\alpha = 1$. A possibility consists in setting, following the previous Remark, $\phi_1(x) = f(x)+x = 2x^2-1$. A second possibility can be derived by setting $f(x) = 0$, from which we have $x^2 = \frac{x+1}{2}$ and then $x = \pm\sqrt{\frac{x+1}{2}}$; in this case, we can take $\phi_2(x) = \sqrt{\frac{x+1}{2}}$. ♦

2.3.2 Fixed point iterations algorithm

We state the fixed point iterations algorithm, based on the fixed point iterate:

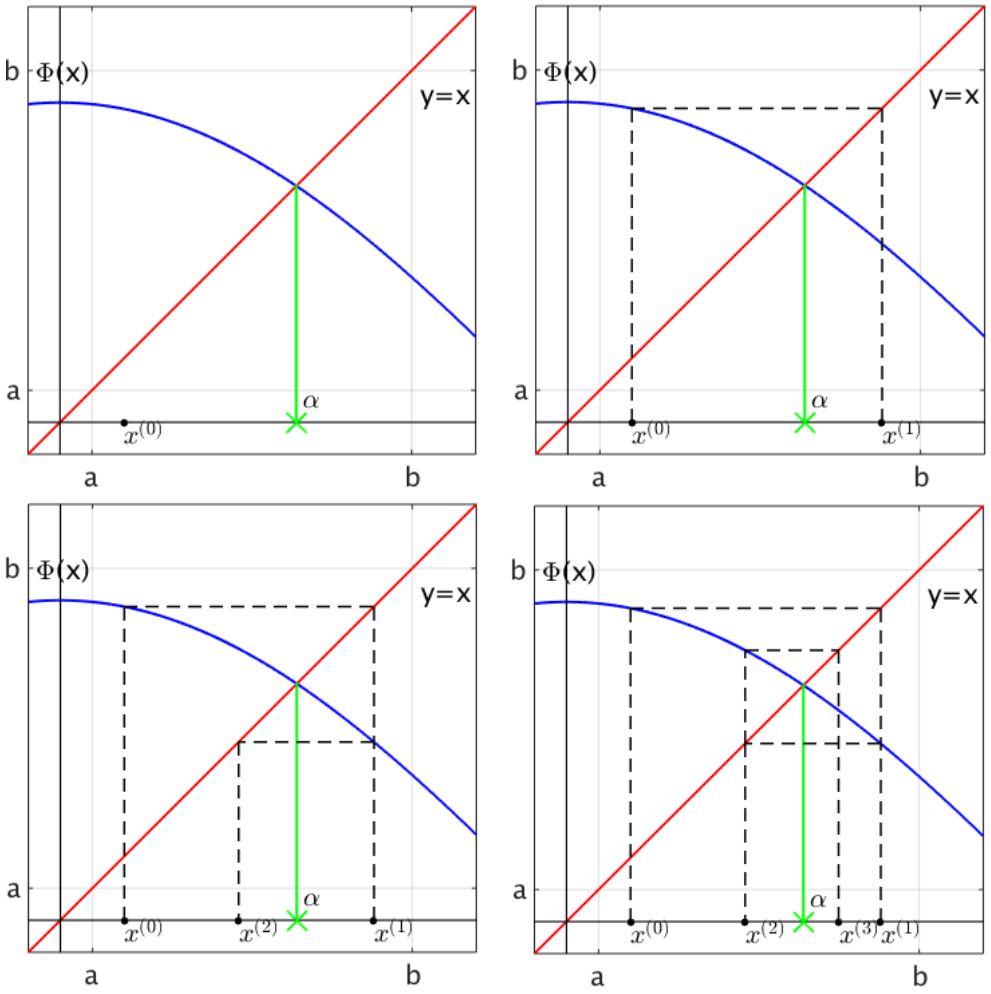
$$x^{(k+1)} = \phi\left(x^{(k)}\right) \qquad k \geq 0, \tag{2.6}$$

for some initial guess $x^{(0)}$.

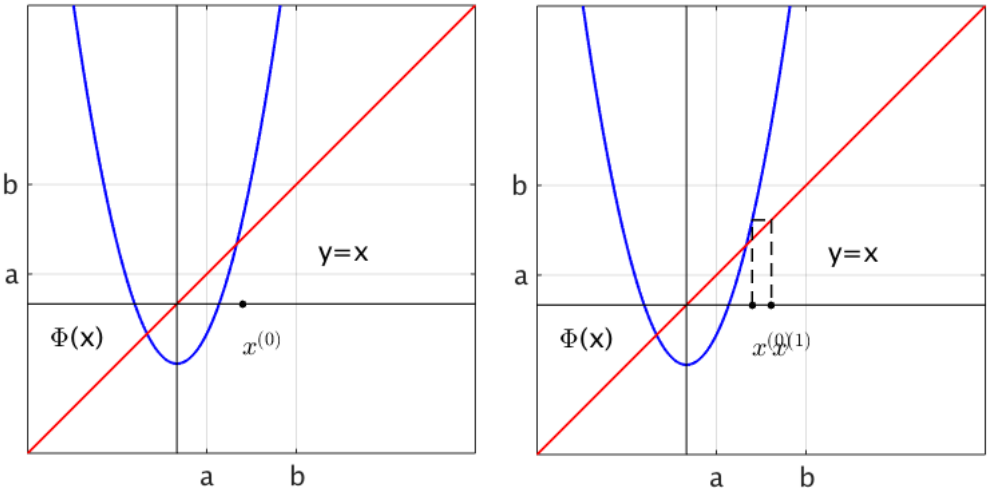
Algorithm 2.5: Fixed point iterations

set $k = 0$ and the initial guess $x^{(0)}$;
while *(stopping criterion is false)* **do**
  $x^{(k+1)} = \phi\left(x^{(k)}\right)$;
  set $k = k+1$;
end

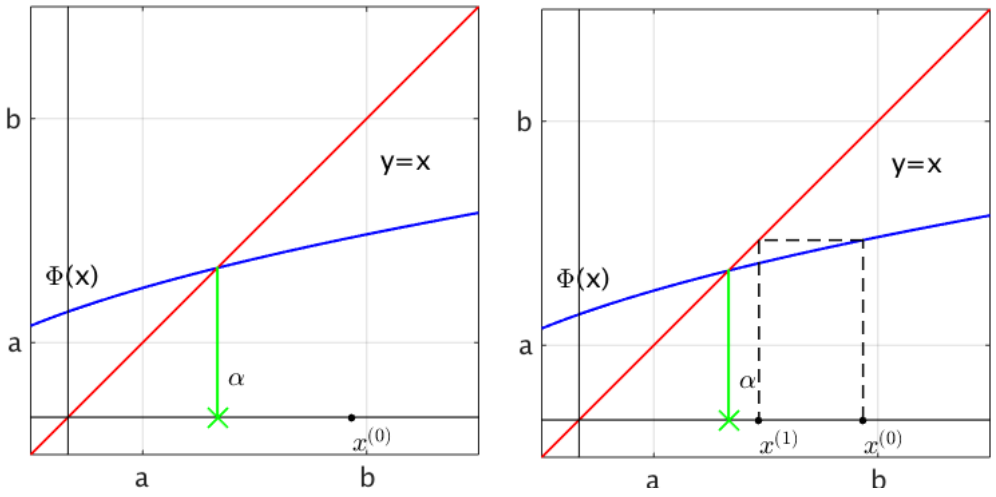
♦ **Example 2.16** We graphically illustrate the fixed point iterations algorithm in the following. First, we consider $\phi(x) = \cos(x)$, with $a = 0.1$, $b = 1.1$, and $x^{(0)} = 0.2$, for which we observe that the algorithm is converging to $\alpha = \cos(\alpha) \simeq 0.7391$.



Then, we consider $\phi(x) = 2x^2 - 1$, with $a = 0.5$, $b = 2$, and $x^{(0)} = 1.1$, for which the algorithm is diverging from the fixed point $\alpha = 1$; we observe that $\phi(x)$ corresponds to the iteration function $\phi_1(x)$ of Example 2.15.



Finally, we consider $\phi(x) = \sqrt{\dfrac{1+x}{2}}$, with $a = 0.5$, $b = 2$, and $x^{(0)} = 1.9$, for which the algorithm converges to $\alpha = 1$; in this case, $\phi(x)$ corresponds to $\phi_2(x)$ in Example 2.15.



♦

2.3.3 Convergence properties of fixed point iterations

We need to identify the properties of the iteration function $\phi(x)$ in terms of existence and uniqueness of the fixed point α, as well as the convergence of the fixed point iterations method.

Proposition 2.7 — Global convergence in an interval. Let us consider the iteration function $\phi : \mathbb{R} \to \mathbb{R}$ and the fixed point iterations of Eq. (2.6).

1. If $\phi \in C^0([a,b])$ and $\phi(x) \in [a,b]$ for all $x \in [a,b]$, then there *exists at least* a fixed point $\alpha \in [a,b]$ of $\phi(x)$.
2. If, in addition to the hypothesis of (1), there exists a constant $L \in [0,1)$ such that $|\phi(x_1) - \phi(x_2)| \leq L\,|x_1 - x_2|$ for all $x_1, x_2 \in [a,b]$, then the fixed point α is *unique* in $[a,b]$ and the fixed point iterations algorithm *converges* ($\lim_{k\to+\infty} x^{(k)} = \alpha$) for all the initial guesses $x^{(0)} \in [a,b]$.

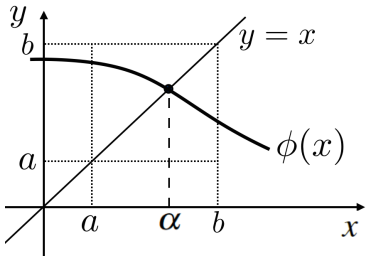
Proof. (1) We show the existence of $\alpha \in [a,b]$ according to the hypothesis (1). We introduce a function $g(x) = \phi(x) - x$ such that $g(\alpha) = 0$. Since $\phi \in C^0([a,b])$, also $g \in C^0([a,b])$. In addition,

since $\phi(x) \in [a,b]$ for all $x \in [a,b]$, we have $g(a) = \phi(a) - a \geq 0$ and $g(b) = \phi(b) - b \leq 0$, for which $g(a)\,g(b) \leq 0$. Since $g(x)$ is satisfying the hypotheses of Theorem 2.1, there exists at least a zero α of $g(x)$ in $[a,b]$; the latter is also a fixed point of $\phi(x)$ (indeed, $g(\alpha) = \phi(\alpha) - \alpha = 0$).

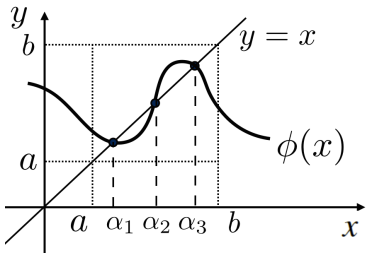
(2) We show the uniqueness of $\alpha \in [a,b]$ and the convergence of the method for all $x^{(0)} \in [a,b]$ according to the hypotheses (2). We assume, by absurd, that there exist two distinct fixed points $\alpha_1 \neq \alpha_2$ such that $\phi(\alpha_1) = \alpha_1$ and $\phi(\alpha_2) = \alpha_2$. According to this assumption, we have $0 < |\alpha_1 - \alpha_2| = |\phi(\alpha_1) - \phi(\alpha_2)| \leq L\,|\alpha_1 - \alpha_2|$; since $L < 1$ by hypothesis, we have $0 < |\alpha_1 - \alpha_2| < |\alpha_1 - \alpha_2|$, which is absurd. Therefore, $\alpha_1 \equiv \alpha_2 = \alpha$, i.e. the fixed point is unique. Regarding the convergence of the method, we observe that the error $e^{(k+1)} = \left|x^{(k+1)} - \alpha\right| = \left|\phi\left(x^{(k)}\right) - \phi(\alpha)\right| \leq \left|x^{(k)} - \alpha\right| = L e^{(k)}$. By recursion, $e^{(k)} \leq L^k e^{(0)}$ for all $k \geq 0$; since $L < 1$, we have $\lim_{k \to +\infty} e^{(k)} = 0$, i.e. the method is convergent for all $x^{(0)} \in [a,b]$. ∎

♦ **Example 2.17** We illustrate the results of Proposition 2.7 with the following examples.

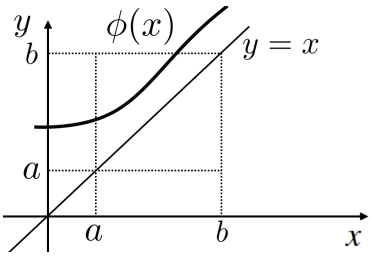
The hypotheses (1) and (2) of Proposition 2.7 are satisfied, therefore there exists an unique fixed point $\alpha \in [a,b]$ and the method converges to α for all $x^{(0)} \in [a,b]$.



The hypotheses (1) are satisfied, but not the hypotheses (2) of Proposition 2.7; therefore, we can only guarantee that there exists at least a fixed point $\alpha \in [a,b]$.



The hypotheses (1) and (2) of Proposition 2.7 are not satisfied, therefore there may not exist any fixed point $\alpha \in [a,b]$.



♦

We consider the following result on the global convergence in the interval $[a,b]$ which uses more restrictive hypotheses on the iteration function $\phi(x)$ with respect to Proposition 2.7.

Proposition 2.8 — Global convergence in an interval. If $\phi \in C^1([a,b])$, $\phi(x) \in [a,b]$ for all $x \in [a,b]$, and $\left|\phi'(x)\right| < 1$ for all $x \in [a,b]$, then there *exists an unique* fixed point $\alpha \in [a,b]$ and the fixed point iterations method *converges* for all $x^{(0)} \in [a,b]$ with *order* at least equal to 1 (linearly), i.e.:

$$\lim_{k \to +\infty} \frac{x^{(k+1)} - \alpha}{x^{(k)} - \alpha} = \phi'(\alpha),$$

with $\phi'(\alpha)$ the asymptotic convergence factor.

We illustrate now some results on the local convergence to the fixed point α, i.e. in a neighborhood of α. First, we recall the Lagrange theorem.

Theorem 2.9 — Lagrange, mean value. If the function $g \in C^1([a,b])$, then there exists $\xi \in (a,b)$ such that $g(a) - g(b) = g'(\xi)(a-b)$.

Proposition 2.10 — Ostrowski, local convergence in a neighborhood of the fixed point. If $\phi \in C^1(I_\alpha)$, with I_α a neighborhood of the fixed point α of $\phi(x)$, and $\left|\phi'(\alpha)\right| < 1$, then, if the initial guess $x^{(0)}$ is "sufficiently" close to α, the fixed point iterations method *converges* with *order* at least equal to 1 (linearly), i.e.:

$$\lim_{k\to+\infty} \frac{x^{(k+1)} - \alpha}{x^{(k)} - \alpha} = \phi'(\alpha),$$

with $\phi'(\alpha)$ the asymptotic convergence factor.

Proof. We only show that the method is at least linearly convergent. Under the hypothesis of the Lagrange Theorem 2.9, we have $x^{(k+1)} - \alpha = \phi\left(x^{(k)}\right) - \phi(\alpha) = \phi'\left(\xi^{(k)}\right)\left(x^{(k)} - \alpha\right)$, for some $\xi^{(k)}$ between α and $x^{(k)}$. If $\lim_{k\to+\infty} x^{(k)} = \alpha$, also $\lim_{k\to+\infty} \xi^{(k)} = \alpha$ and hence $\lim_{k\to+\infty} \frac{x^{(k+1)} - \alpha}{x^{(k)} - \alpha} = \lim_{k\to+\infty} \phi'\left(\xi^{(k)}\right) = \phi'(\alpha)$. ∎

Remark 2.13 Following Proposition 2.10, we observe for $\phi \in C^1(I_\alpha)$ that:

- if $\left|\phi'(\alpha)\right| < 1$, the fixed point iterations converge to α with order at least equal to 1, if $x^{(0)}$ is "sufficiently" close to α;
- if $\left|\phi'(\alpha)\right| \equiv 1$, the convergence of the method to α depends on the properties of $\phi(x)$ in the neighborhood I_α and the choice of the initial guess $x^{(0)}$ (as a matter of fact, the method may converge or diverge);
- if $\left|\phi'(\alpha)\right| > 1$, the convergence of the method to α is impossible, unless $x^{(0)} \equiv \alpha$.

Proposition 2.11 — Local convergence in a neighborhood of the fixed point. If $\phi \in C^2(I_\alpha)$, with I_α a neighborhood of the fixed point α of $\phi(x)$, $\phi'(\alpha) = 0$, and $\phi''(\alpha) \neq 0$, then, if the initial guess $x^{(0)}$ is "sufficiently" close to α, the fixed point iterations method *converges* with *order* 2 (quadratically), i.e.:

$$\lim_{k\to+\infty} \frac{x^{(k+1)} - \alpha}{\left(x^{(k)} - \alpha\right)^2} = \frac{1}{2}\phi''(\alpha),$$

with $\frac{1}{2}\phi''(\alpha)$ the asymptotic convergence factor.

The following result generalizes the previous ones.

Proposition 2.12 — Local convergence in a neighborhood of the fixed point. If $\phi \in C^p(I_\alpha)$ for $p \geq 1$, with I_α a neighborhood of the fixed point α of $\phi(x)$, $\phi^{(i)}(\alpha) = 0$ for all $i = 1, \ldots, p-1$, and $\phi^{(p)}(\alpha) \neq 0$, then, if the initial guess $x^{(0)}$ is "sufficiently" close to α, the fixed point iterations method *converges* with *order* p, i.e.:

$$\lim_{k \to +\infty} \frac{x^{(k+1)} - \alpha}{\left(x^{(k)} - \alpha\right)^p} = \frac{1}{p!}\phi^{(p)}(\alpha),$$

with $\frac{1}{p!}\phi^{(p)}(\alpha)$ the asymptotic convergence factor.

2.3.4 Stopping criterion for fixed point iterations

We need to consider a stopping criterion to terminate the fixed point iterations of Algorithm 2.5. With this aim, we introduce a suitable *error estimator* $\widetilde{e}^{(k)}$ of the error $e^{(k)} := \left|x^{(k)} - \alpha\right|$. Such error estimator is based on the *difference of successive iterates*, i.e.:

$$\widetilde{e}^{(k)} = \begin{cases} \left|\delta^{(k-1)}\right| & \text{if } k \geq 1 \\ tol + 1 & \text{if } k = 0 \end{cases} \qquad \text{with } \delta^{(k)} := x^{(k+1)} - x^{(k)} \qquad \text{for } k \geq 0;$$

$tol > 0$ is a suitable tolerance. The fixed point iterations algorithm is stopped at the first iteration $\overline{k}$ such that $\widetilde{e}^{(\overline{k})} < tol$ or when $\overline{k} = k_{max}$, with k_{max} the maximum number of iterations allowed.

We observe that $\alpha - x^{(k+1)} = \alpha - x^{(k)} + x^{(k)} - x^{(k+1)} = \left(\alpha - x^{(k)}\right) - \delta^{(k)}$. Moreover, if $\phi \in C^1(I_\alpha)$, we have from Theorem 2.9 that $\alpha - x^{(k+1)} = \phi'\left(\xi^{(k)}\right)\left(\alpha - x^{(k)}\right)$ for some $\xi^{(k)}$ between $x^{(k)}$ and α. Therefore, we have $x^{(k)} - \alpha = \phi'\left(\xi^{(k)}\right)\left(x^{(k)} - \alpha\right) - \delta^{(k)}$ and hence:

$$x^{(k)} - \alpha = -\frac{1}{1 - \phi'\left(\xi^{(k)}\right)}\delta^{(k)}, \tag{2.7}$$

for some $\xi^{(k)}$ between $x^{(k)}$ and α. We use the previous result to determine if the stopping criterion based on the difference of successive iterates is satisfactory or not. If $\phi'(x) \simeq 0$ in a neighborhood of α ($\phi'(\alpha) \simeq 0$), the criterion is *satisfactory* since $e^{(k)} \simeq \widetilde{e}^{(k+1)}$. If $\phi'(x) > -1$, but $\phi'(x) \simeq -1$ in a neighborhood of α, the criterion is still *satisfactory* since $e^{(k)} \simeq \frac{1}{2}\widetilde{e}^{(k+1)}$ (the error is *overestimated* by the estimator by a factor 2). Conversely, if $\phi'(x) < 1$, but $\phi'(x) \simeq 1$ in a neighborhood of α, the criterion is *unsatisfactory* since $e^{(k)} \ll \widetilde{e}^{(k+1)}$, that is the error is *underestimated* by the error estimator.

2.3.5 The Newton method as a fixed point iterations method

The *Newton method* (see Sec. 2.2) can be used to find the zero α of a general function $f(x)$, for which the Newton iterate is specified in Eq. (2.3). The problem of finding the zero α of $f(x)$ with the Newton method can be recast in a fixed point iterations method by using the iteration function $\phi_N(x)$ such that $\phi_N(\alpha) = \alpha$. From Eqs. (2.3) and (2.6) the iteration function associated to the Newton method reads:

$$\phi_N(x) = x - \frac{f(x)}{f'(x)}. \tag{2.8}$$

It follows that the properties of the Newton method, including convergence to α can be deduced from those of the iteration function $\phi_N(x)$.

Proposition 2.13 If $f \in C^m(I_\alpha)$, with I_α a neighborhood of the zero α and $m \geq 1$ the multiplicity of α, for the iteration function $\phi_N(x)$ of Eq. (2.8), we have $\phi'_N(\alpha) = 1 - \frac{1}{m}$.

Proof. The proof, reported in Exercises Series 4, is based on rewriting $f(x)$ as $f(x) = (x-\alpha)^m g(x)$ for $x \in I_\alpha$, with the function $g(x)$ such that $g(\alpha) \neq 0$ and $g^{(i)}(\alpha) = 0$ for all $i = 1, \ldots, m$. ∎

Corollary 2.14 If $f \in C^2(I_\alpha)$, α is a zero simple ($m = 1$), and $x^{(0)}$ is "sufficiently" close to α, then the Newton method converges with order 2 (quadratically), indeed:

$$\lim_{k \to +\infty} \frac{x^{(k+1)} - \alpha}{\left(x^{(k)} - \alpha\right)^2} = \frac{1}{2}\phi''_N(\alpha) = \frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}.$$

Proof. The result follows from Propositions 2.11 and 2.12, Eq. (2.8), and Proposition 2.13. ∎

Remark 2.14 From Proposition 2.13, if α is a zero simple ($m = 1$), we have $\phi'_N(\alpha) = 1 - \frac{1}{m} \equiv 0$.

Corollary 2.15 If $f \in C^m(I_\alpha)$, α is a zero of multiplicity $m > 1$, and $x^{(0)}$ is "sufficiently" close to α, then the Newton method converges with order 1 (linearly), indeed:

$$\lim_{k \to +\infty} \frac{x^{(k+1)} - \alpha}{x^{(k)} - \alpha} = \phi'_N(\alpha) = 1 - \frac{1}{m} \neq 0.$$

Proof. The result follows from Proposition 2.10, Eq. (2.8), and Proposition 2.13. ∎

Similarly, to the *modified Newton method* (see Sec. 2.2.2), based on the iterate of Eq. (2.4), we associate the iteration function $\phi_{mN}(x)$ defined as:

$$\phi_N(x) = x - m\frac{f(x)}{f'(x)}, \tag{2.9}$$

where m is the multiplicity of the zero α.

Proposition 2.16 If $f \in C^m(I_\alpha)$, with I_α a neighborhood of the zero α and $m \geq 1$ the multiplicity of α, for the iteration function $\phi_{mN}(x)$ of Eq. (2.9), we have $\phi'_{mN}(\alpha) = 1 - m\frac{1}{m} \equiv 0$ for all $m \geq 1$.

Proof. The result follows analogously to that of Proposition 2.13. ∎

Corollary 2.17 If $f \in C^2(I_\alpha) \cap C^m(I_\alpha)$, α is a zero of multiplicity $m \geq 1$, and $x^{(0)}$ is "sufficiently" close to α, then the modified Newton method converges with order 2 (quadratically).

Proof. The result follows from Propositions 2.11 and 2.12, Eq. (2.9), and Proposition 2.16. ∎

Regarding the quality of the *stopping criterion* based on the *difference of successive iterates* for the Newton method discussed in Sec. 2.2.3, we recall the properties presented in Sec. 2.3.4 for fixed point iterations. From Eq. (2.7) and Proposition 2.13, we remark that:

$$e^{(k)} = \left|x^{(k)} - \alpha\right| \simeq \left|\frac{1}{1 - \phi'_N(\alpha)}\right| \widetilde{e}^{(k+1)} = m\,\widetilde{e}^{(k+1)},$$

where the error estimator $\widetilde{e}^{(k+1)} = \delta^{(k)} = \left| x^{(k+1)} - x^{(k)} \right|$ and $m \geq 1$ is the multiplicity of the zero α. Therefore, if the zero α is simple ($m = 1$), we have $e^{(k)} \simeq \widetilde{e}^{(k+1)}$ and the stopping criterion based on the difference of successive iterates is *satisfactory*. Otherwise, for a zero of multiplicity $m > 1$, and especially for $m \gg 1$, the criterion is *unsatisfactory* since the error is *underestimated* by the error estimator, i.e. $e^{(k)} \gg \widetilde{e}^{(k+1)}$. By using similar arguments for the *modified Newton method*, the stopping criterion based on the difference of successive iterates is *satisfactory*, since $e^{(k)} \simeq \widetilde{e}^{(k+1)}$ regardless of the multiplicity $m \geq 1$ of the zero.

2.3.6 Fixed point iterations for vector valued functions

The fixed point iterations method can be used with vector valued iteration functions $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^d$, for some $d \geq 1$. In this case, the problem consists in finding the vector $\boldsymbol{\alpha} \in \mathbb{R}^d$, the fixed point, such that $\boldsymbol{\phi}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}$. The fixed point iterations method consists in sequentially applying the following fixed point iterate:

$$\mathbf{x}^{(k+1)} = \boldsymbol{\phi}\left(\mathbf{x}^{(k)}\right) \qquad \text{for all } k \geq 0,$$

given the initial guess $\mathbf{x}^{(0)} \in \mathbb{R}^d$; as stopping criterion the one based on the *difference of successive iterates* can be used similarly to Sec. 2.3.4, i.e. $\widetilde{\mathbf{e}}^{(k)} = \left\| \boldsymbol{\delta}^{(k-1)} \right\|_2 < tol$ for $k \geq 1$, with tol a prescribed tolerance and $\boldsymbol{\delta}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$.

Algorithms for Chapter 2

Algorithm 1 Bisection method (Algorithms 2.1 and 2.2)

Input: f, a, b, tol, k_{max}
Output: α

1: Set $k = 0$, $a^{(0)} = a$, $b^{(0)} = b$, $x^{(0)} = (a+b)/2$, $\widetilde{e}^{(0)} = (b-a)/2$
2: **while** ($\widetilde{e}^{(k)} > tol$ and $k < k_{max}$) **do**
3: **if** $f\left(x^{(k)}\right) = 0$ **then**
4: Set $\alpha = x^{(k)}$
5: Return
6: **else if** $f\left(a^{(k)}\right) f\left(x^{(k)}\right) < 0$ **then**
7: Set $a^{(k+1)} = a^{(k)}$, $b^{(k+1)} = x^{(k)}$
8: **else if** $f\left(b^{(k)}\right) f\left(x^{(k)}\right) < 0$ **then**
9: Set $a^{(k+1)} = x^{(k)}$, $b^{(k+1)} = b^{(k)}$
10: **end if**
11: Set $x^{(k+1)} = (a^{(k+1)} + b^{(k+1)})/2$
12: Set $\widetilde{e}^{(k+1)} = (b^{(k+1)} - a^{(k+1)})/2$
13: Set $k = k+1$
14: **end while**
15: Set $\alpha = x^{(k)}$

Algorithm 2 Newton method (Algorithm 2.3)

Input: $f, x^{(0)}, tol, k_{max}$
Output: α

1: Set $k = 0$, $\widetilde{e}^{(0)} = tol + 1$
2: **while** ($\widetilde{e}^{(k)} > tol$ and $k < k_{max}$) **do**
3: Set $x^{(k+1)} = x^{(k)} - f\left(x^{(k)}\right) / f'\left(x^{(k)}\right)$
4: Set $\widetilde{e}^{(k+1)} = \|x^{(k+1)} - x^{(k)}\|$
5: Set $k = k+1$
6: **end while**
7: Set $\alpha = x^{(k)}$

The algorithm for the modified Newton method is analogous, adding the multiplicity m in the input, and modifying line 3 as in Algorithm 2.4.

Algorithm 3 Newton method for systems of nonlinear equations

Input: $\mathbf{F}, \mathbf{x}^{(0)}, tol, k_{max}$
Output: α

Set $k = 0$, $\widetilde{e}^{(0)} = tol + 1$
while ($\widetilde{e}^{(k)} > tol$ and $k < k_{max}$) **do**
    Set $\delta^{(k)} = -\left(J_{\mathbf{F}}\left(\mathbf{x}^{(k)}\right)\right)^{-1} \mathbf{F}\left(\mathbf{x}^{(k)}\right)$
    Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k)}$
    Set $\widetilde{e}^{(k+1)} = \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$
    Set $k = k + 1$
end while
Set $\alpha = \mathbf{x}^{(k)}$

Algorithm 4 Fixed point iterations (Algorithm 2.5)

Input: $\phi, x^{(0)}, tol, k_{max}$
Output: α

Set $k = 0$, $\widetilde{e}^{(0)} = tol + 1$
while ($\widetilde{e}^{(k)} > tol$ and $k < k_{max}$) **do**
    Set $x^{(k+1)} = \phi\left(x^{(k)}\right)$
    Set $\widetilde{e}^{(k+1)} = \|x^{(k+1)} - x^{(k)}\|$
    Set $k = k + 1$
end while
Set $\alpha = x^{(k)}$

Corrections to Chapter 2

- In the hypotheses of Proposition 2.12, it should be $p > 1$.

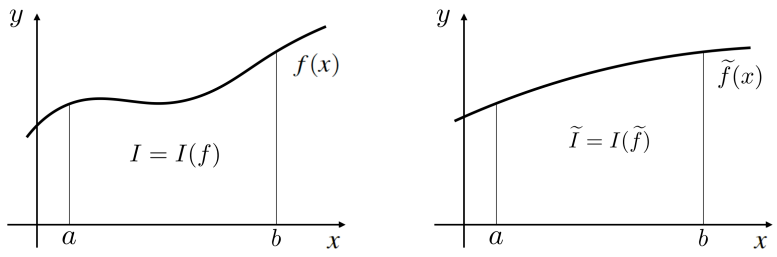
3. Approximation of Functions and Data

We consider the approximation of functions and data, specifically by means of *interpolation* and the *least–squares* method.

3.1 Motivations and Examples

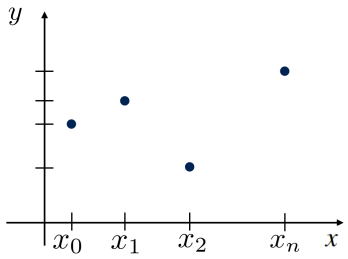
We illustrate through examples the motivations behind the need to approximate functions and data.

♦ **Example 3.1** Let us assume to be interested in computing the integral I of a function $f(x)$ in the interval $[a,b]$, i.e. $I = I(f) = \int_a^b f(x)\,dx$, but we are unable to provide a closed form solution for the function at hand. One possibility consists in approximating $f(x)$ with another function $\widetilde{f}(x)$ which can be integrated in closed form as $\widetilde{I} = I(\widetilde{f}) = \int_a^b \widetilde{f}(x)\,dx$ such that $\widetilde{I} \simeq I$.



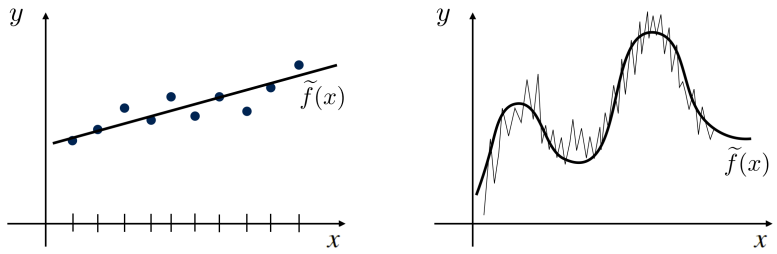
♦

♦ **Example 3.2** By assuming that a function $f(x)$ is known only through its evaluation in a set of $n+1$ nodes $\{x_i\}_{i=0}^n$, i.e. the data couples $\{(x_i, f(x_i))\}_{i=0}^n$, we may be interested in defining an approximation $\widetilde{f}(x)$ of the unknown function $f(x)$.



♦

♦ **Example 3.3** Given a set of data couples $\{(x_i, y_i)\}_{i=0}^{n}$, we may want to determine the intermediate values or make predictions outside the set of $n+1$ nodes $\{x_i\}_{i=0}^{n}$.



♦

3.1.1 Approximation of functions by Taylor's polynomials

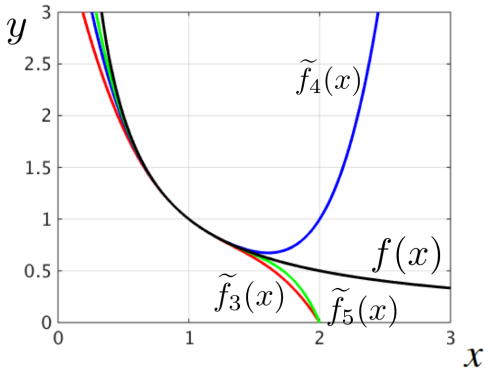
A manner to approximate a function $f \in C^n(I_{x_0})$ in a neighborhood I_{x_0} of a point $x_0 \in \mathbb{R}$ is based on the *Taylor's polynomial* (expansion) of order n. The Taylor's expansion of $f(x)$ around x_0 reads:

$$\widetilde{f}(x) = f(x_0) + \sum_{i=1}^{n} \frac{1}{i!} f^{(i)}(x_0)\,(x - x_0)^i.$$

However, the approximation of $f(x)$ with $\widetilde{f}(x)$ presents some drawbacks. First, the evaluation of n derivatives of $f(x)$ is required, which may be computationally expensive. Then, the Taylor's expansion is accurate only in a neighborhood I_{x_0} of x_0, while generally inaccurate "far" from x_0.

♦ **Example 3.4** We consider the Taylor's expansion of $f(x) = \dfrac{1}{x}$ of order n, say $\widetilde{f}_n(x)$, around $x_0 = 1$.

Since $f^{(i)}(x) = (-1)^i\, i!\, x^{-(i+1)}$ for $i = 0, 1, \ldots, n$, we have $\widetilde{f}(x) = \widetilde{f}_n(x) = 1 + \sum_{i=1}^{n} (-1)^n\,(x-1)^i$. As reported in the figure, the approximations provided by $\widetilde{f}_n(x)$ may be very inaccurate "far" from $x_0 = 1$.



♦

3.2 Interpolation

We introduce and define the concept of interpolation and list different types of interpolations.

Definition 3.1 We consider a set of $n+1$ data couples $\{(x_i,y_i)\}_{i=0}^n$ with $\{x_i\}_{i=0}^n$ $n+1$ *distinct nodes*, i.e. such that $x_i \neq x_j$ for all $i \neq j$ for $i,j = 0,\dots,n$; in the case the function $f(x)$ is known, we set $y_i = f(x_i)$ for all $i = 0,\dots,n$. *Interpolating* the data couples $\{(x_i,y_i)\}_{i=0}^n$ means determining the approximate function $\widetilde{f}(x)$ such that $\widetilde{f}(x_i) = y_i$ for all $i = 0,\dots,n$ or, if $f(x)$ is known, such that $\widetilde{f}(x_i) = f(x_i)$ for all $i = 0,\dots,n$. The function $\widetilde{f}(x)$ is called *interpolant* of the data at the nodes.

There exist different types of interpolation. For example:

- *polynomial* interpolation, for which $\widetilde{f}(x) = a_0 + a_1 x + \cdots + a_n x^n$ for some $n+1$ coefficients $a_0, a_1, \dots$;
- *rational* interpolation, for which $\widetilde{f}(x) = \dfrac{a_0 + a_1 x + \cdots + a_k x^k}{a_{k+1} + a_{k+2} x + \cdots + a_{k+n+1} x^n}$ for some coefficients $a_0, a_1, \dots$ with $k, n \geq 0$;
- *trigonometric* interpolation, for which $\widetilde{f}(x) = \sum_{j=-M}^{M} a_j e^{\imath jx}$, being $\imath$ the imaginary unit ($\imath^2 = -1$) and $e^{\imath jx} = \cos(jx) + \imath \sin(jx)$, for some M and complex coefficients a_j.
- *piecewise polynomial* interpolation;
- *splines*;
- ...

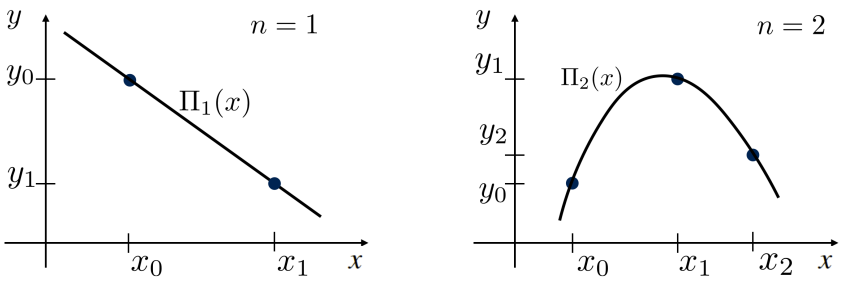
3.2.1 Lagrange interpolating polynomials

We consider the *polynomial* interpolation, specifically determined as *Lagrange interpolating polynomials*. The polynomial interpolation is based on the following result which determines the correspondence between the number of distinct nodes and the degree of the interpolant.

Proposition 3.1 For any set of data couples $\{(x_i,y_i)\}_{i=0}^n$, being $\{x_i\}_{i=0}^n$ $n+1$ distinct nodes, there exists an *unique* polynomial, say $\Pi_n(x)$, of degree less than or equal to n, such that $\Pi_n(x_i) = y_i$ for all $i = 0,\dots,n$. $\Pi_n(x) \in \mathbb{P}_n$ is called *interpolating polynomial* of the data at the nodes $\{x_i\}_{i=0}^n$. If $f(x)$ is a *continuous* function for which $y_i = f(x_i)$ for all $i = 0,\dots,n$, then $\Pi_n f(x) \in \mathbb{P}_n$ is the *interpolating polynomial* of the *function* $f(x)$ at the nodes $\{x_i\}_{i=0}^n$.

We recall that $\mathbb{P}_n$ indicates the set of polynomials of degree less than or equal to n.

♦ **Example 3.5** We illustrate two cases for which $n = 1$ (left) and $n = 2$ (right).



♦

We need to determine the interpolating polynomial $\Pi_n(x)$ (or $\Pi_n f(x)$), which assumes the expression $\Pi_n(x) = a_0 + a_1 x + \cdots + a_n x^n$; the goal consists in computing the coefficients $\{a_i\}_{i=0}^n$ of such

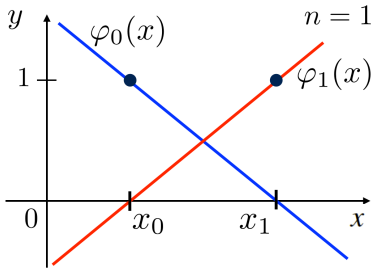
polynomial of degree n. With this aim, we consider a special family of polynomials associated to the $n+1$ distinct nodes $\{x_i\}_{i=0}^n$.

Definition 3.2 For a set of $n+1$ distinct nodes $\{x_i\}_{i=0}^n$, the *Lagrange characteristic* function associated to the node x_k, say $\varphi_k \in \mathbb{P}_n$, is a *polynomial* of degree n such that $\varphi_k(x_i) = \delta_{ki}$ for all $i = 0, \dots, n$, where $\delta_{ki} = \begin{cases} 0 & \text{if } i \neq k \\ 1 & \text{if } i = k \end{cases}$, for which:

$$\varphi_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}.$$

The set $\{\varphi_k(x)\}_{k=0}^n$ is the basis of *Lagrange characteristic polynomials*.

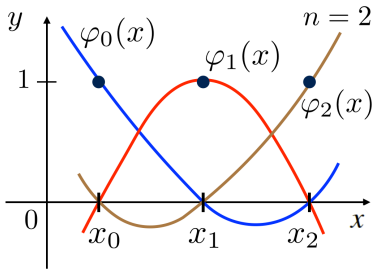
♦ **Example 3.6** We illustrate the bases of Lagrange characteristic polynomials for $n = 1$ and $n = 2$.



$n = 1$

$$\varphi_0(x) = \frac{x - x_1}{x_0 - x_1} \in \mathbb{P}_1$$

$$\varphi_1(x) = \frac{x - x_0}{x_1 - x_0} \in \mathbb{P}_1$$



$n = 2$

$$\varphi_0(x) = \frac{x - x_1}{x_0 - x_1} \frac{x - x_2}{x_0 - x_2} \in \mathbb{P}_2$$

$$\varphi_1(x) = \frac{x - x_0}{x_1 - x_0} \frac{x - x_2}{x_1 - x_2} \in \mathbb{P}_2$$

$$\varphi_2(x) = \frac{x - x_0}{x_2 - x_0} \frac{x - x_1}{x_2 - x_1} \in \mathbb{P}_2$$

♦

Definition 3.3 Given the basis of Lagrange characteristic polynomials $\{\varphi_k(x)\}_{k=0}^n$ associated to the $n+1$ distinct nodes $\{x_i\}_{i=0}^n$, the *Lagrange interpolating polynomial* of the data couples $\{(x_i, y_i)\}_{i=0}^n$ is:

$$\Pi_n(x) = \sum_{k=0}^{n} y_k \, \varphi_k(x).$$

If the function $f(x)$ is given and is continuous, the *Lagrange interpolating polynomial* of the function $f(x)$ at the nodes $\{x_i\}_{i=0}^n$ is:

$$\Pi_n f(x) = \sum_{k=0}^{n} f(x_k) \, \varphi_k(x).$$

Remark 3.1 The Lagrange interpolating polynomial $\Pi_n(x)$ interpolates the data at the nodes; indeed, $\Pi_n(x_i) = \sum_{k=0}^{n} y_k \, \varphi_k(x_i) = \sum_{k=0}^{n} y_k \, \delta_{ki} = y_i$ for all $i = 0, \dots, n$. Analogously, $\Pi_n f(x)$ interpolates the function $f(x)$ at the nodes.

The Lagrange interpolating polynomial $\Pi_n(x) \in \mathbb{P}_n$ uses the basis of Lagrange characteristic polynomials $\{\varphi_k(x)\}_{k=0}^n$ to determine the coefficients $\{a_i\}_{i=0}^n$ of such polynomial of degree n, i.e. $\Pi_n(x) = \sum_{k=0}^{n} y_k\, \varphi_k(x) = a_0 + a_1 x + \cdots + a_n x^n$.

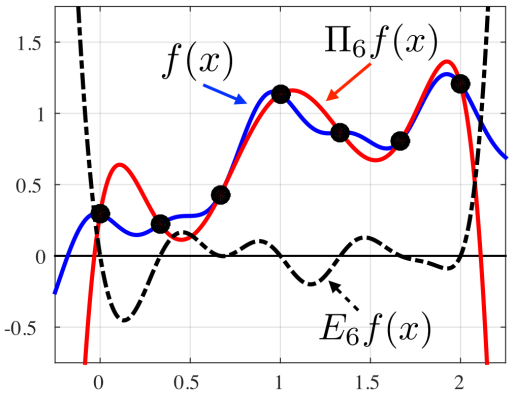
♦ **Example 3.7** We consider the Lagrange polynomial interpolation of the data couples $\{(1,3)\}$, $\{(2,2)\}$, and $\{(4,6)\}$ for which $n=2$. Following Example 3.6, we have $x_0=1$, $x_1=2$, and $x_2=4$, for which $\varphi_0(x) = \frac{1}{3}(x-2)(x-4) = \frac{1}{3}x^2 - 2x + \frac{8}{3}$, $\varphi_1(x) = -\frac{1}{2}(x-1)(x-4) = -\frac{1}{2}x^2 + \frac{5}{2}x + 2$, and $\varphi_2(x) = \frac{1}{6}(x-1)(x-2) = \frac{1}{6}x^2 - \frac{1}{2}x + \frac{1}{3}$. The Lagrange interpolating polynomial of the data is the polynomial of degree $n=2$ $\Pi_2(x) = y_0\,\varphi_0(x) + y_1\,\varphi_1(x) + y_2\,\varphi_2(x) = x^2 - 4x + 6$, being $y_0=3$, $y_1=2$, and $y_3=6$. ♦

Definition 3.4 For a continuous function $f(x)$ and the interval $I=[a,b]$ such that the $n+1$ nodes are ordered as $a = x_0 < x_1 < \cdots < x_n = b$, we define the *error function* $E_n f(x) := f(x) - \Pi_n f(x)$ associated to the interpolating polynomial $\Pi_n f(x)$. The *error* is $e_n(f) := \max_{x\in I} |E_n f(x)|$.

Remark 3.2 $\Pi_n f(x)$ interpolates $f(x)$ at the nodes, indeed $E_n f(x_i) = 0$ for all $i = 0, \ldots, n$.

♦ **Example 3.8** Given the function $f(x) = \sin(x) + \frac{1}{4}\sin\left(2\pi x + \sqrt{3}\right) + \frac{1}{10}\sin\left(4\pi x + \sqrt{7}\right)$, we consider its polynomial interpolation over $n+1$ equally spaced nodes in $I=[0,2]$.

We set the polynomial degree $n=6$ for which we obtain the interpolating polynomial $\Pi_6 f(x)$ of $f(x)$ at the nodes $x_0=0$, $x_1=\frac{1}{3}$, $x_2=\frac{2}{3}$, $x_3=1$, $x_4=\frac{4}{3}$, $x_5=\frac{5}{3}$, and $x_6=2$. We also plot the error function $E_6 f(x) = f(x) - \Pi_6 f(x)$, for which we observe that $E_6 f(x_i) = 0$ for all $i = 0, \ldots, 6$.



♦

Proposition 3.2 Let us consider $n+1$ distinct nodes $\{x_i\}_{i=0}^n$ in an interval $I=[a,b]$ such that $a = x_0 < x_1 < \cdots < x_n = b$ and the polynomial interpolant $\Pi_n f(x)$ of a function $f(x)$ in such nodes. Then, if $f \in C^{n+1}(I)$, for all $x \in I$ there exists $\xi = \xi(x) \in I$ such that:

$$E_n f(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi(x))\, \omega_n(x), \tag{3.1}$$

where $\omega_n(x) := \prod_{i=0}^{n}(x - x_i)$. Moreover, the error $e_n(f)$ is bounded by the error estimator $\widetilde{e}_n(f)$ as:

$$e_n(f) \le \widetilde{e}_n(f) := \frac{1}{(n+1)!} \max_{x\in I} \left| f^{(n+1)}(x) \right| \max_{x\in I} |\omega_n(x)|. \tag{3.2}$$

Proposition 3.3 Let us consider $n+1$ *equally spaced* nodes $\{x_i\}_{i=0}^n$ in the interval $I=[a,b]$ such that $x_i = x_0 + ih$ for $i=0,\dots,n$ with $x_0=a$, $x_n=b$, and $h=\dfrac{b-a}{n}$, then, by recalling the definition of $\omega_n(x)$ in Proposition 3.2, we have:

$$\max_{x\in I}|\omega_n(x)| \le \frac{n!}{4}h^{n+1} = \frac{n!}{4}\left(\frac{b-a}{n}\right)^{n+1}.$$

Therefore, from Eq. (3.2), the error $e_n(f)$ is bounded as:

$$e_n(f) \le \widetilde{e}_n(f) := \frac{h^{n+1}}{4(n+1)}\max_{x\in I}\left|f^{(n+1)}(x)\right| = \frac{1}{4(n+1)}\left(\frac{b-a}{n}\right)^{n+1}\max_{x\in I}\left|f^{(n+1)}(x)\right|. \quad (3.3)$$

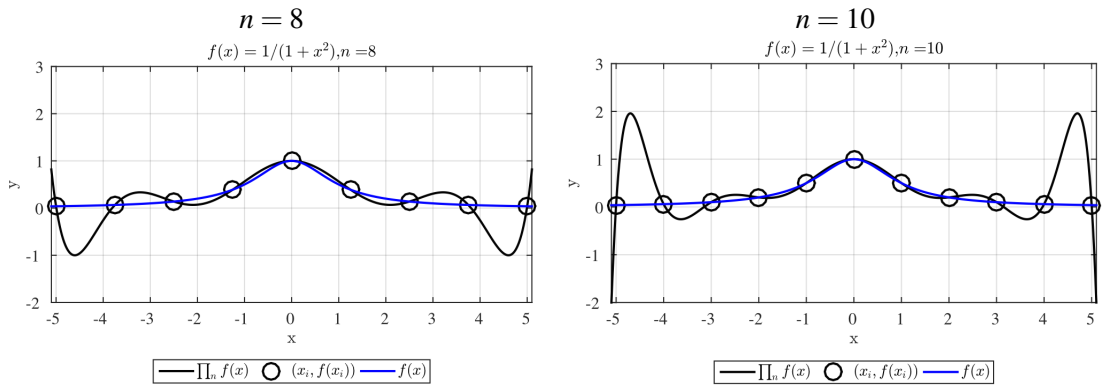
Corollary 3.4 Under the same hypotheses of Proposition 3.3, we have:

$$\max_{x\in I}\left|f'(x)-(\Pi_n f)'(x)\right| \le C_n h^n \max_{x\in I}\left|f^{(n+1)}(x)\right|$$

for some positive constant C_n.

Remark 3.3 If the $n+1$ nodes are equally spaced in the interval I, the error $e_n(f)$ may tend to zero or not for $n\to+\infty$ depending on the function $f(x)$ to be interpolated. From Eq. (3.3), we observe that $\lim_{n\to+\infty}\dfrac{h^{n+1}}{4(n+1)}=0$. Conversely, $\max_{x\in I}\left|f^{(n+1)}(x)\right|$ may grow with n; indeed, there exist functions for which $\lim_{n\to+\infty}\max_{x\in I}\left|f^{(n+1)}(x)\right|=+\infty$. In these cases, the growth of $\max_{x\in I}\left|f^{(n+1)}(x)\right|$ may not be compensated by the decrease of $\dfrac{h^{n+1}}{4(n+1)}$ with n, for which $\lim_{n\to+\infty}\widetilde{e}_n(f)=+\infty$; hence, the error estimator $\widetilde{e}_n(f)$ "blows up" and typically the error $e_n(f)$ behaves in a similar manner. The so called *Runge phenomenon* is an instance of such behavior, for which the error function $E_n f(x)$ tends to "blow up" for increasing values of n in proximity of the borders of the interval I when equally spaced nodes are used for the polynomial interpolation.

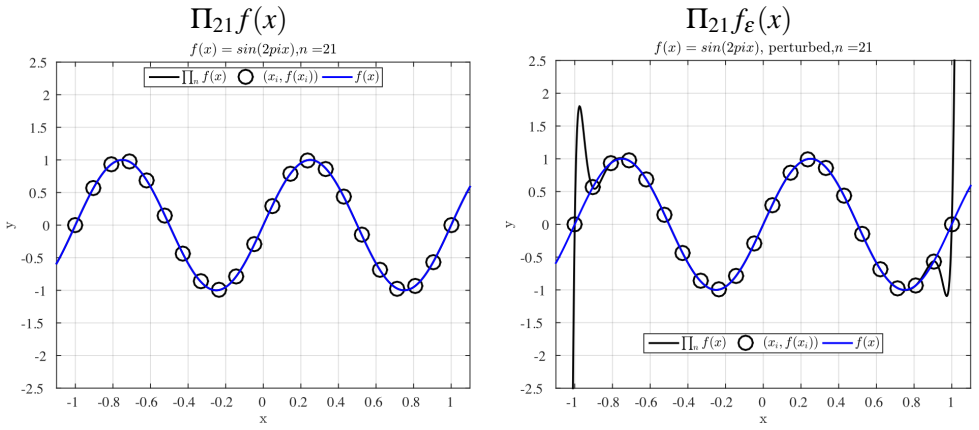
♦ **Example 3.9** We consider the polynomial interpolation of the *Runge function* $f(x)=\dfrac{1}{1+x^2}$ over $n+1$ equally spaced nodes in the interval $I=[-5,5]$. In this case, the interpolating polynomials $\Pi_n f(x)$ of $f(x)$ exhibit the so called Runge phenomenon for increasing values of n as it is visible in proximity of the boundaries of the interval I. Moreover, $\lim_{n\to+\infty} e_n(f)=+\infty$.



♦

Remark 3.4 An other important issue which may arise with polynomial interpolation on *equally spaced* nodes concerns the *stability of the interpolating polynomial*. Indeed, using equally spaced $n+1$ nodes in the interval I may lead to a significant sensitivity of the interpolating polynomial $\Pi_n(x)$, or $\Pi_n f(x)$ if the function $f(x)$ is known, to *perturbations* on the data.

♦ **Example 3.10** We highlight such stability issue by considering the polynomial interpolation of $f(x) = \sin(\pi x)$ over $n+1$ equally spaced nodes in the interval $I = [-1,1]$. By setting $n = 21$ we obtain the polynomial interpolant $\Pi_{21} f(x)$ which qualitatively coincides with $f(x)$. We apply now the polynomial interpolation to a perturbed function $f_\varepsilon(x) = f(x) + \varepsilon(x)$, with $\varepsilon(x)$ a random function such that $|\varepsilon(x)| < 10^{-3}$ for all $x \in I$. Its polynomial interpolant of degree $n = 21$ $\Pi_{21} f_\varepsilon(x)$ is very sensible to this "small" perturbation, being very different from $\Pi_{21} f(x)$.



♦

A remedy to mitigate the Runge phenomenon and stability issues for polynomial interpolation consists in using nodes which are *not* equally spaced in the interval I. The following definition provides a special family of nodes that can be used for polynomial interpolation.

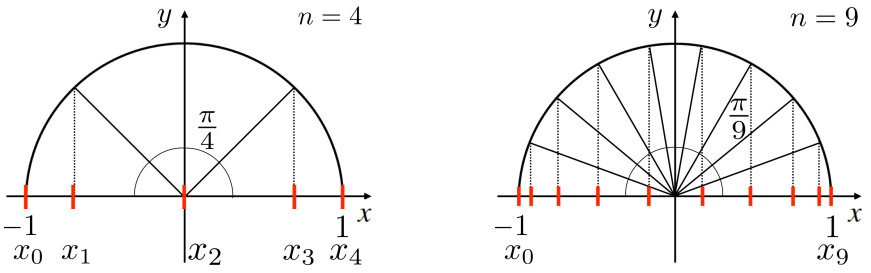
Definition 3.5 For a given $n \geq 1$, the $n+1$ *Chebyshev–Gauss–Lobatto nodes* in the reference interval $\widehat{I} = [-1,1]$ are:

$$\widehat{x}_i = -\cos\left(\frac{\pi}{n} i\right) \qquad i = 0, \ldots, n;$$

in the general interval $I = [a,b]$ the $n+1$ Chebyshev–Gauss–Lobatto nodes are:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \widehat{x}_i \qquad i = 0, \ldots, n.$$

♦ **Example 3.11** We graphically highlight the $n+1$ Chebyshev–Gauss–Lobatto nodes $\{\widehat{x}_i\}_{i=0}^n$ in the reference interval $\widehat{I} = [-1,1]$ for $n = 4$ (left) and $n = 9$ (right).

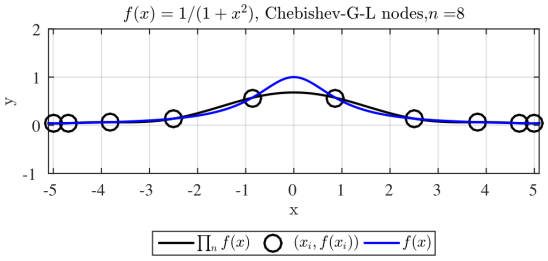


♦

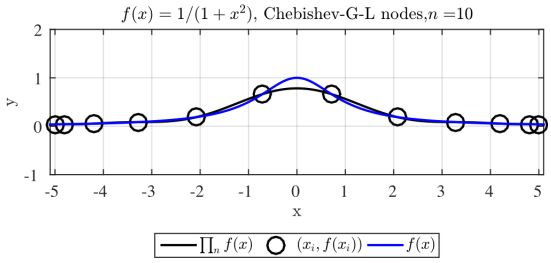
Proposition 3.5 If $f \in C^{n+1}(I)$, with the interval $I = [a,b]$, and the $n+1$ Chebyshev–Gauss–Lobatto are used in I, then $\lim_{n\to+\infty} \Pi_n f(x) = f(x)$ for all $x \in I$, i.e. $\lim_{n\to+\infty} e_n(f) = 0$, and the stability issues are mitigated.

♦ **Example 3.12** By recalling and using the same data of Examples 3.9 and 3.10, we show that the use of Chebyshev–Gauss–Lobatto (CGL) nodes avoids the insurgency of the Runge phenomenon and mitigates the stability issues of polynomial interpolation, respectively.

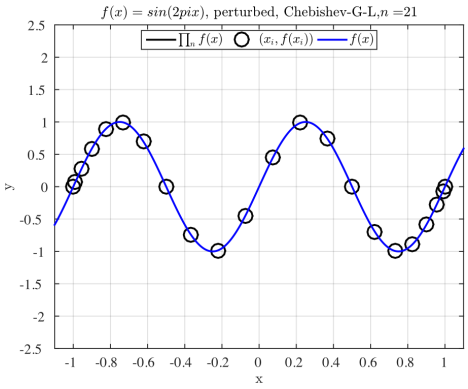
$n = 8$ with CGL nodes



$n = 10$ with CGL nodes



$\Pi_{21} f_\varepsilon(x)$ with CGL nodes



♦

As anticipated, the Lagrange interpolating polynomial $\Pi_n(x) \in \mathbb{P}_n$ uses the basis of Lagrange characteristic polynomials $\{\varphi_k(x)\}_{k=0}^n$ to determine the coefficients $\{a_i\}_{i=0}^n$ of this polynomial, i.e. $\Pi_n(x) = \sum_{k=0}^{n} y_k \varphi_k(x) = a_0 + a_1 x + \cdots + a_n x^n$. An alternative approach to Lagrange polynomial interpolation consists in computing directly the $n+1$ coefficients $\mathbf{a} = (a_0, a_1, \ldots, a_n)^T \in \mathbb{R}^{n+1}$ by enforcing the $n+1$ interpolation constraints $\Pi_n(x_i) = y_i$ for all $i = 0, \ldots, n$; i.e., $\Pi_n(x_i) = a_0 + a_1 x_i + \cdots + a_n x_i^n = y_i$ for all $i = 0, \ldots, n$. The problem boils down to solve the following linear system:

$$B\mathbf{a} = \mathbf{y} \tag{3.4}$$

where $B \in \mathbb{R}^{(n+1)\times(n+1)}$ is the Vandermonde matrix, with $B_{ij} = (x_{i-1})^{j-1}$ for $i, j = 1, \ldots, n+1$, and $\mathbf{y} = (y_0, y_1, \ldots, y_n)^T \in \mathbb{R}^{n+1}$. The linear system (3.4) admits an unique solution if and only if $\det(B) \neq 0$, i.e. if and only if the $n+1$ nodes $\{x_i\}_{i=0}^n$ are distinct. We remark that interpolating polynomials computed by solving the linear system (3.4) may suffer stability issues already for relatively "small" values of n, due to the large conditioning numbers typically associated to the matrix B.

Remark 3.5 The polynomial interpolation is in general not adequate to extrapolate information outside the interval I containing the nodes (see e.g. Example 3.8).

3.2.2 Trigonometric interpolation

We consider the *trigonometric interpolation* which uses trigonometric basis functions and it is also referred as *discrete Fourier series*; such kind of interpolation is used for periodic signals and functions. With this aim, we consider a *periodic* function $f : [0, 2\pi] \to \mathbb{C}$, i.e. such that $f(0) = f(2\pi)$; $\imath$ indicates the imaginary unit such that $\imath^2 = -1$.

Definition 3.6 Given $n+1$ nodes $\{x_j\}_{j=0}^n$ such that $x_j = jh$ for $j = 0, \dots, n$ with $h = \dfrac{2\pi}{n+1}$, the trigonometric interpolant of the periodic function $f : [0, 2\pi] \to \mathbb{C}$, say $I_t f(x)$, is:

$$I_t f(x) = \sum_{k=-(M+\mu)}^{M+\mu} \widetilde{c}_k e^{\imath k x},$$

where:

$$M = \begin{cases} n/2 & \text{if } n \text{ is even} \\ (n-1)/2 & \text{if } n \text{ is odd} \end{cases}, \qquad \mu = \begin{cases} 0 & \text{if } n \text{ is even} \\ 1 & \text{if } n \text{ is odd} \end{cases},$$

$$\widetilde{c}_k = \begin{cases} c_k & \text{for } k = -M, \dots, M \\ c_k/2 & \text{if } k = -(M+1) \text{ or } M+1 \end{cases},$$

and

$$c_k = \frac{1}{n+1} \sum_{j=0}^{n} f(x_j) e^{-\imath k j h}.$$

The coefficients $\{c_k\}_{k=0}^n \in \mathbb{C}$ and $e^{\imath k x} = \cos(kx) + \imath \sin(kx)$. If $f(x)$ is a real valued function (i.e. $f(x) \in \mathbb{R}$ for all $x \in \mathbb{R}$), then also the trigonometric interpolant is real valued (i.e. $I_t f(x) \in \mathbb{R}$ for all $x \in \mathbb{R}$); indeed, in this case, $c_{-k} = \overline{c}_k$ for $k = 0, \dots, n$.

Remark 3.6 The trigonometric interpolant $I_t f(x)$ interpolates $f(x)$ at the $n+1$ nodes $\{x_j\}_{j=0}^n$, indeed $I_t f(x_j) = f(x_j)$ for all $j = 0, \dots, n$.

The computation of the coefficients $\{c_k\}_{k=0}^n$ according to the above mentioned procedure requires $O(n^2)$ flops; instead, using the *Fast Fourier Transform* (FFT) only requires $O(n \log n)$ flops.

3.2.3 Piecewise polynomial interpolation

Piecewise polynomial interpolation is also known as *composite* interpolation and approximates a function $f(x)$ *locally* with polynomials. Piecewise polynomial interpolation is a good alternative to polynomial interpolation with equally spaced nodes to extract information inside an interval.

Definition 3.7 Let us consider $n+1$ distinct nodes $\{x_i\}_{i=0}^n$ in the interval $I = [a, b]$ such that $a = x_0 < x_1 < \cdots < x_n = b$ for which n subintervals $I_i = [x_i, x_{i+1}]$ are defined for $i = 0, \dots, n-1$; we indicate with $H := \max_{i=0,\dots,n-1} |I_i| = \max_{i=0,\dots,n-1} (x_{i+1} - x_i)$ the characteristic size of such subintervals. Given the set of data couples $\{(x_i, y_i)\}_{i=0}^n$, the *piecewise linear interpolating polynomial* $\Pi_1^H(x)$ of the data is a piecewise polynomial of degree 1 such that $\Pi_1^H(x) \in \mathbb{P}_1$ for all $x \in I_i$ and

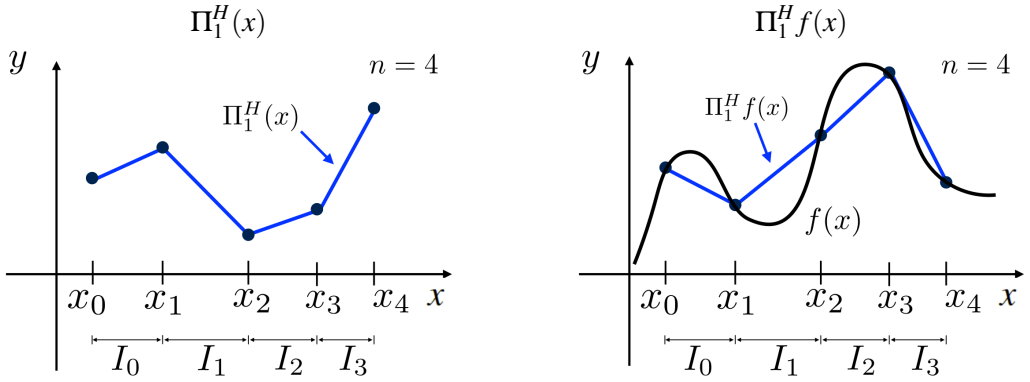
$i = 0, \ldots, n-1$ (i.e. $\Pi_1^H(x)|_{I_i} \in \mathbb{P}_1$ for all $i = 0, \ldots, n-1$), with:

$$\Pi_1^H(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} x \qquad \text{for all } i = 0, \ldots, n-1.$$

If the function $f \in C^0(I)$ is known, then the *piecewise linear interpolating polynomial* $\Pi_1^H f(x)$ of the function $f(x)$ at the nodes is $\Pi_1^H f(x)|_{I_i} \in \mathbb{P}_1$ for all $i = 0, \ldots, n-1$, with:

$$\Pi_1^H f(x) = y_i + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} x \qquad \text{for all } i = 0, \ldots, n-1.$$

♦ **Example 3.13** We report the piecewise linear interpolants of $n+1$ data couples, say $\Pi_1^H(x)$, and of a continuous function $f(x)$, say $\Pi_1^H f(x)$, in $n+1$ nodes in the interval I; specifically, $n = 4$ (i.e. $n+1 = 5$ nodes are used). The characteristic size of the subintervals $\{I_i\}_{i=0}^3$ is $H = \max_{i=0,1,2,3} |I_i|$.



♦

Definition 3.8 If the function $f \in C^0(I)$ is known, we define the *error* associated to the piecewise linear interpolating polynomial $\Pi_1^H f(x)$ as $e_1^H(f) := \max_{x \in I} |f(x) - \Pi_1^H f(x)|$.

Proposition 3.6 If $f \in C^2(I)$, then the error $e_1^H(f)$ associated to the piecewise linear interpolating polynomial $\Pi_1^H f(x)$ can be bounded by the error estimator $\widetilde{e}_1^H(f)$ as:

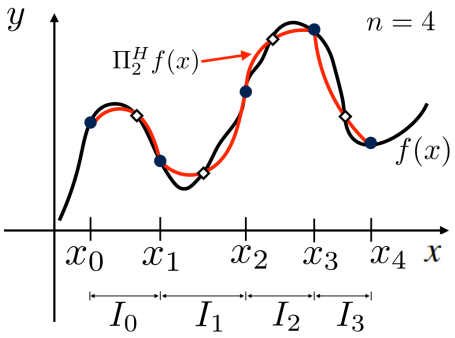
$$e_1^H(f) \leq \widetilde{e}_1^H(f) := \frac{H^2}{8} \max_{x \in I} |f''(x)|,$$

for which the error *converges* to zero with *order* 2 in H (quadratically).

In analogy with $\Pi_1^H(x)$, one can define the *piecewise quadratic polynomial* $\Pi_2^H(x)$ as $\Pi_2^H(x)|_{I_i} \in \mathbb{P}_2$ for all the subintervals I_i of I from $i = 0, \ldots, n-1$; if $f \in C^0(I)$ is known, then we use the notation $\Pi_2^H f(x)$. Similarly, one can define the *piecewise interpolating polynomial of degree* $r \geq 1$ $\Pi_r^H(x)$ as $\Pi_r^H(x)|_{I_i} \in \mathbb{P}_r$ for all $i = 0, \ldots, n-1$ (or $\Pi_r^H f(x)$ if $f \in C^0(I)$ is known).

♦ **Example 3.14** We consider the piecewise quadratic interpolation of a continuous function $f(x)$, say $\Pi_2^H f(x)$, over $n+1$ nodes in the interval I; specifically, we set $n = 4$.

The piecewise quadratic interpolant $\Pi_2^H f(x)$ interpolates $f(x)$ at the $n+1=5$ nodes and at intermediate points internal to each subinterval of I (as for example corresponding to the mid–points of the subintervals).



♦

Proposition 3.7 If $f \in C^{r+1}(I)$, then the error $e_r^H(f) := \max_{x\in I} \left| f(x) - \Pi_r^H f(x) \right|$ associated to the piecewise interpolating polynomial of degree $r \geq 1$ $\Pi_r^H f(x)$ can be bounded by the error estimator $\widetilde{e}_r^H(f)$ as:

$$e_r^H(f) \leq \widetilde{e}_r^H(f) := C_r H^{r+1} \max_{x\in I} \left| f^{(r+1)}(x) \right|,$$

with C_r a positive constant, for which the error *converges* to zero with *order* $r+1$ in H.

Remark 3.7 Piecewise interpolating polynomials $\Pi_r^H f(x)$ of any degree $r \geq 1$ are only C^0–continuous across the subintervals (internal nodes); see e.g. Example 3.14.

3.2.4 Spline functions

Spline functions, or simply *splines*, are piecewise interpolating polynomials which are smoother than standard piecewise polynomials $\Pi_r^H f(x)$ across the subintervals. Splines and their generalizations (B–splines and NURBS) are widely used in Computer Graphics and industrial applications for which high regularity of the interpolants is necessary.

Definition 3.9 A *cubic spline function*, say $s_3(x)$, is a piecewise interpolating polynomial of degree 3 which is C^2–continuous across the internal nodes of the interval I. Specifically, given $n+1$ nodes in $I = [x_0, x_n]$ with $x_0 < x_1 < \cdots < x_n$ and the subintervals $I_i = [x_i, x_{i+1}]$ for $i = 0, \ldots, n-1$, we have:

$$s_3(x)|_{I_i} \in \mathbb{P}_3 \quad \text{for all } i = 0, \ldots, n-1 \qquad \text{and} \qquad s_3''(x_i^-) = s_3''(x_i^+) \quad \text{for all } i = 1, \ldots, n-1.$$

We can write $s_3(x)|_{I_i} = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$ for all $i = 0, \ldots, n-1$, with the $4n$ coefficients $\{a_{j,i}\}$ to be determined for $j = 0,1,2,3$ and $i = 0, \ldots, n-1$. In order to obtain the coefficients of $s_3(x)$ we impose the following $4n-2$ constraints according to Definition 3.9:

$$s_3(x_i) = y_i \quad (\text{or } s_3(x_i) = f(x_i) \text{ if } f \text{ is known}) \qquad \text{for all } i = 0, \ldots, n,$$

$$s_3(x_i^-) = s_3(x_i^+), \quad s_3'(x_i^-) = s_3'(x_i^+), \quad s_3''(x_i^-) = s_3''(x_i^+) \qquad \text{for all } i = 1, \ldots, n-1.$$

In order to fully determine $s_3(x)$ one needs to enforce 2 additional constraints, whose choice determine the type of cubic spline.

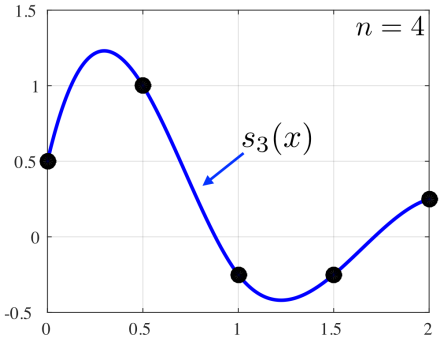
Definition 3.10 If one sets $s_3''(x_0) = s_3''(x_n) = 0$, then $s_3(x)$ is a *natural interpolating cubic spline*.

If one sets $s_3'''(x_1^-) = s_3'''(x_1^+)$ and $s_3'''(x_{n-1}^-) = s_3'''(x_{n-1}^+)$, then $s_3(x)$ is a *not–a–knot interpolating cubic spline*.

The MATLAB command `spline` considers not–a–knot interpolating cubic splines.

♦ **Example 3.15** We consider the interpolation of a set of $n+1$ data by a not–a–knot interpolating cubic spline $s_3(x)$; specifically, $n = 4$.

The cubic spline $s_3(x)$ interpolates the data at the $n+1 = 5$ nodes $\{x_i\}_{i=0}^4$ and is C^2–continuous across each internal node $\{x_i\}_{i=1}^3$.



♦

Proposition 3.8 Let us consider $n+1$ distinct nodes $\{x_i\}_{i=0}^n$ delimiting the interval $I = [x_0, x_n]$ for which n subintervals $I_i = [x_i, x_{i+1}]$ are defined for $i = 0, \ldots, n-1$ and $H := \max_{i=0,\ldots,n-1} |I_i|$ is the characteristic size of such subintervals. Then, if $f \in C^4(I)$ and $s_3(x)$ is its *natural interpolating cubic spline* at the nodes, we have the following error estimates:

$$\max_{x\in I} \left| f^{(k)}(x) - s_3^{(k)}(x) \right| \leq C_k H^{4-k} \max_{x\in I} \left| f^{(4)}(x) \right| \qquad \text{for } k = 0, 1, 2$$

and

$$\max_{x\in I\setminus\{x_1,\ldots,x_{n-1}\}} \left| f'''(x) - s_3'''(x) \right| \leq C_3 H^3 \max_{x\in I} \left| f^{(4)}(x) \right|,$$

with $C_k > 0$ positive constants, for which the *convergence order* of the error is $4-k$ in H depending on the order of derivation $k = 0, 1, 2, 3$.

3.3 Least–Squares Method

The *least–squares approximation* is ideal to extract information from a large set of data, both with and without uncertainty and noise, as well as to make predictions outside the interval in which these data are available.

Definition 3.11 Given the set of data couples $\{(x_i, y_i)\}_{i=0}^n$ (or $\{(x_i, f(x_i))\}_{i=0}^n$ if the function $f(x)$ is given) and an integer $m \geq 0$, we look for an approximating polynomial $\widetilde{f}_m(x)$ of degree m such that:

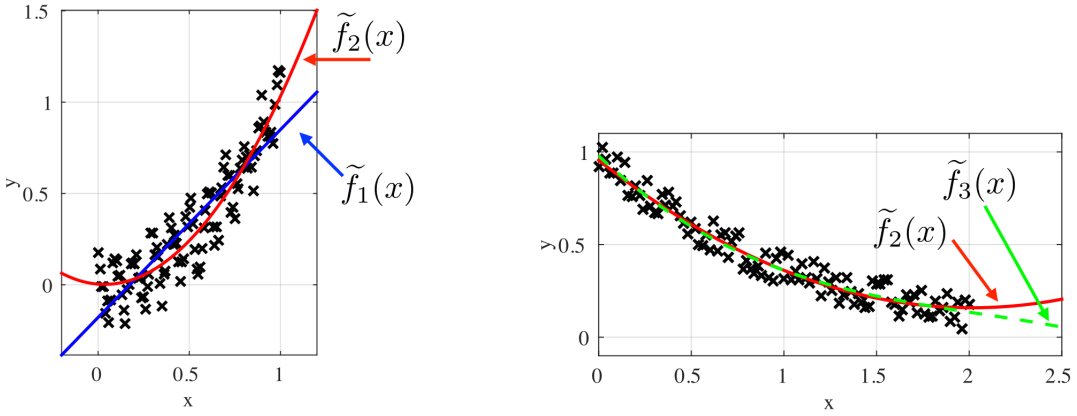
$$\sum_{i=0}^{n} \left(y_i - \widetilde{f}_m(x_i) \right)^2 \leq \sum_{i=0}^{n} \left(y_i - p_m(x_i) \right)^2 \qquad \text{for all } p_m \in \mathbb{P}_m.$$

If $\widetilde{f}_m \in \mathbb{P}_m$ exists, then it is called *least–squares approximating polynomial of degree m* of the data (or the function $f(x)$).

Remark 3.8 By convention, we assume that the nodes $\{x_i\}_{i=0}^n$ are distinct and $0 \leq m \leq n$. In a typical scenario in which the least–squares method is used, one has $0 \leq m \ll n$.

Remark 3.9 The least–squares approximating polynomial $\widetilde{f}_m(x)$ does not in general interpolate the data (or the function $f(x)$) at the nodes. Specifically, only if $m = n$ one can ensure that $\widetilde{f}_m(x_i) = y_i$ (or $\widetilde{f}_m(x_i) = f(x_i)$) for all $i = 0, \ldots, n$; indeed, in this case, $\widetilde{f}_m(x)$ coincides with the polynomial interpolant $\Pi_n(x)$ of degree n (or $\Pi_n f(x)$).

♦ **Example 3.16** We graphically represent the least–squares approximating polynomials $\widetilde{f}_m(x)$ of degrees m for two relatively large sets of data $\{(x_i, y_i)\}_{i=0}^n$, with $n = 100$. We consider $m = 1$ and 2 (left) and $m = 2$ and 3 (right).



♦

Definition 3.12 Following Definition 3.11, the least–squares approximating polynomial $\widetilde{f}_1(x)$ of degree $m = 1$ is called *regression line* or least–squares straight line.

As for the polynomial interpolation, determining the least–squares approximating polynomial $\widetilde{f}_m(x)$ of degree m consists in determining its $m+1$ coefficients $\{a_i\}_{i=0}^m$; indeed, $\widetilde{f}_m(x) = a_0 + a_1 x + \cdots + a_m x^m$. With this aim, we define the coefficients vector $\mathbf{a} = (a_0, a_1, \ldots, a_m)^T \in \mathbb{R}^{m+1}$ and the functional $\Phi : \mathbb{R}^{m+1} \to \mathbb{R}$ as:

$$\Phi(\mathbf{b}) = \sum_{i=0}^{n} [y_i - (b_0 + b_1 x_i + \cdots + b_m x_i^m)]^2,$$

which is associated to the set of data couples $\{(x_i, y_i)\}_{i=0}^n$ for the general coefficient vector $\mathbf{b} = (b_0, b_1, \ldots, b_m)^T \in \mathbb{R}^{m+1}$. Then, the *least–squares method* consists in determining the coefficients vector $\mathbf{a}$ of the polynomial $\widetilde{f}_m(x)$ of degree m such that:

$$\Phi(\mathbf{a}) = \min_{\mathbf{b} \in \mathbb{R}^{m+1}} \Phi(\mathbf{b}).$$

Since Φ is differentiable, the previous minimization problem is equivalent to solve the following differential problem:

$$\text{find } \mathbf{a} \in \mathbb{R}^{m+1} \quad : \quad \frac{\partial \Phi}{\partial b_j}(\mathbf{a}) = 0 \qquad \text{for all } j = 0, \ldots, m. \tag{3.5}$$

In turn, such differential problem reduces into solving the following linear system:

$$A\,\mathbf{a} = \mathbf{q}, \tag{3.6}$$

where $A \in \mathbb{R}^{(m+1)\times(m+1)}$ and $\mathbf{q} \in \mathbb{R}^{m+1}$ read:

$$A = \begin{bmatrix} (n+1) & \sum_{i=0}^{n} x_i & \cdots & \sum_{i=0}^{n} x_i^m \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 & \cdots & \sum_{i=0}^{n} x_i^{m+1} \\ \vdots & \vdots & & \vdots \\ \sum_{i=0}^{n} x_i^m & \sum_{i=0}^{n} x_i^{m+1} & \cdots & \sum_{i=0}^{n} x_i^{2m} \end{bmatrix} \quad \text{and} \quad \mathbf{q} = \begin{bmatrix} \sum_{i=0}^{n} y_i \\ \sum_{i=0}^{n} x_i y_i \\ \vdots \\ \sum_{i=0}^{n} x_i^m y_i \end{bmatrix},$$

respectively. By recalling the Vandermonde matrix $B \in \mathbb{R}^{(n+1)\times(m+1)}$, with $B_{ij} = (x_{i-1})^{j-1}$ for $i = 1,\ldots,n+1$ and $j = 1,\ldots,m+1$, and the data vector $\mathbf{y} = (y_0, y_1, \ldots, y_n)^T \in \mathbb{R}^{n+1}$, we observe that:

$$A = B^T B \qquad \text{and} \qquad \mathbf{q} = B^T \mathbf{y},$$

respectively.

Remark 3.10 The linear system (3.6) is a generalization of the linear system (3.4) used for the polynomial interpolation. As a matter of fact, if the nodes are distinct and $m = n$, we obtain that $\widetilde{f}_m(x) = \Pi_n(x)$ and solving the linear systems (3.4) and (3.6) yields equivalent results.

♦ **Example 3.17** We illustrate the derivation of the linear system (3.6) for $m = 1$ (i.e. $\widetilde{f}_1(x)$ is the regression line) and $n \geq m$. In this case, the functional $\Phi(\mathbf{b})$ reads:

$$\Phi(\mathbf{b}) = \sum_{i=0}^{n} \left[y_i - (b_0 + b_1 x_i)\right]^2 = \sum_{i=0}^{n} \left[y_i^2 + b_0^2 + b_1^2 x_i^2 - 2b_0 y_i - 2b_1 x_i y_i + 2b_0 b_1 x_i\right].$$

In order to formulate the problem as in Eq. (3.5), we compute the partial derivatives of Φ:

$$\frac{\partial \Phi}{\partial b_0}(\mathbf{b}) = \sum_{i=0}^{n} \left[2b_0 - 2y_i + 2b_1 x_i\right],$$

$$\frac{\partial \Phi}{\partial b_1}(\mathbf{b}) = \sum_{i=0}^{n} \left[2b_1 x_i^2 - 2x_i y_i + 2b_0 x_i\right].$$

Then, problem (3.5) can be written as the linear system (3.6) with $A \in \mathbb{R}^{2\times 2}$ and $\mathbf{q} \in \mathbb{R}^2$, being:

$$A = \begin{bmatrix} (n+1) & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \end{bmatrix} \quad \text{and} \quad \mathbf{q} = \begin{bmatrix} \sum_{i=0}^{n} y_i \\ \sum_{i=0}^{n} x_i y_i \end{bmatrix},$$

respectively. Notice that, in this case, the Vandermonde matrix $B \in \mathbb{R}^{(n+1)\times 2}$ reads:

$$B = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}.$$

♦

Corrections to Chapter 3

- Definition 3.6, it is $h = \frac{2\pi}{n}$.
- Definition 3.7, the first and second equations are, respectively

$$\Pi_1^H(x)|_{I_i} = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) \quad \text{for all } i = 0, \ldots, n-1.$$

$$\Pi_1^H f(x)|_{I_i} = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \quad \text{for all } i = 0, \ldots, n-1.$$

- Definition 3.9, the conditions are

$$s_3(x)|_{I_i} \in \mathbb{P}_3 \quad \text{for all } i = 0, \ldots, n-1, \quad \text{and} \quad s_3^{(k)}(x_i^-) = s_3^{(k)}(x_i^+) \text{ for all } i = 1, \ldots, n-1 \text{ and } k = 0, 1, 2.$$

- Proposition 3.8, in the second equation it should be

$$\max_{x \in I \setminus \{x_1, \ldots, x_{n-1}\}} \left| f'''(x) - s_3'''(x) \right| \leq C_3 H \max_{x \in I} \left| f^{(4)}(x) \right|$$

Bibliography

[1] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*, 2nd ed. Texts in Applied Mathematics 37, Springer, Berlin and Heidelberg, 2007.

[2] Alfio Quarteroni, Fausto Saleri, and Paola Gervasio. *Scientific Computing with MATLAB and Octave*, 4th ed. Texts in Computational Science and Engineering 2, Springer–Verlag, Berlin and Heidelberg, 2014.