# Chapter 1. Preliminaries

**Numerical Analysis**, a branch of Applied Mathematics situated at the border between Mathematics and Computer Science, produces methods and procedures for finding numerical solutions of various problems, with a given precision. Standard topics in a Numerical Analysis course are the approximation of problems by simpler problems, the construction of algorithms, iteration methods, error analysis, stability, asymptotic error formulas, the effects of machine arithmetic, etc.

The study will focus on issues such as:

- Problems modeled by functions that do not have an analytical expression, whose values are known only at a discrete set of points. Based on these, we want to approximate values of the function at new points, values of the derivatives or integrals of the function, etc. Such problems lead to finite and divided differences, interpolation formulas, numerical differentiation and integration schemes and many others.

- In many practical situations it is necessary to solve various types of equations or systems of equations, such as: algebraic, transcendental, differential, or integral equations, whose exact solutions cannot be found. They need to be approximated numerically.

An approximating procedure must satisfy several properties:

1. To be *convergent*, meaning the sequence of iterations (successive approximations) must converge to the exact solution, in order to produce "better and better" approximations.

2. To be *stable* (to have stability), meaning that small variations of the input data should lead to small variations in the results (the approximating solutions).

3. From its structure and properties, to be able to estimate the *error* and the *speed of convergence* of the approximating method.

Many times, the conditions that ensure stability of a numerical method coincide with the ones that guarantee its convergence.

# 1 Taylor Polynomials

We start with a very useful tool from Calculus, Taylor's theorem. This will be needed for both the development and understanding of many of the numerical methods discussed later on.

In fact, Taylor polynomials give the very first example of a numerical method, being used as a way to evaluate other functions approximately.

**Theorem 1.1.** [Taylor's Theorem] *Let $f : [a,b] \to \mathbb{R}$ be a function with $n+1$ continuous derivatives on $[a,b]$, for some $n \geq 0$ and let $x, x_0 \in [a,b]$. Then*

$$f(x) = p_n(x) + R_{n+1}(x), \tag{1.1}$$

*where*

$$p_n(x) = f(x_0) + \frac{(x-x_0)}{1!}f'(x_0) + \cdots + \frac{(x-x_0)^n}{n!}f^{(n)}(x_0) \tag{1.2}$$

*is **Taylor's polynomial of degree** $n$ **of** $f$ **at** $x_0$ and*

$$R_{n+1}(x) = \frac{1}{n!}\int_{x_0}^{x}(x-t)^n f^{(n+1)}(t)dt = \frac{(x-x_0)^{n+1}}{(n+1)!}f^{(n+1)}(\xi),\ \xi \text{ between } x \text{ and } x_0, \tag{1.3}$$

*is the **error** of the Taylor approximation.*

**Example 1.2.** Using Taylor's theorem with $x_0 = 0$, we obtain the following well-known formulas:

$$\begin{aligned}
e^x &= 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \frac{x^{n+1}}{(n+1)!}e^{\xi_x},\\
\cos x &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots + (-1)^n\frac{x^{2n}}{(2n)!} + (-1)^{n+1}\frac{x^{2n+2}}{(2n+2)!}\cos\xi_x,\\
\sin x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + (-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!} + (-1)^n\frac{x^{2n+1}}{(2n+1)!}\sin\xi_x,\\
(1+x)^a &= 1 + \binom{a}{1}x + \binom{a}{2}x^2 + \cdots + \binom{a}{n}x^n + \binom{a}{n+1}\frac{x^{n+1}}{(1+\xi_x)^{n+1-a}},
\end{aligned}$$

with

$$\binom{a}{k} = \frac{a(a-1)\cdots(a-k+1)}{k!},\ k = 1,2,3,\ldots,\ a \in \mathbb{R}.$$

An important special case of the last formula is $a = -1$, with $x$ replaced by $-x$:

$$\frac{1}{1-x} \quad = \quad 1 + x + x^2 + \cdots + x^n + \frac{x^{n+1}}{1-x},$$

where the remainder has a simpler form than before. This is easily proved by multiplying both sides by $1 - x$ and then simplifying. Rearranging the terms, we obtain the familiar formula for a partial sum of the Geometric series:

$$1 + x + x^2 + \cdots + x^n \quad = \quad \frac{1 - x^{n+1}}{1 - x}, \; x \neq 1.$$

Series representations for the functions in Example 1.2 can be obtained by letting $n \to \infty$. Recall that the series for the first three functions converge for all $x \in \mathbb{R}$, while those for the last two converge for $|x| < 1$. So, by taking Taylor polynomials of higher and higher degree, we can obtain better and better approximations of the functions above.

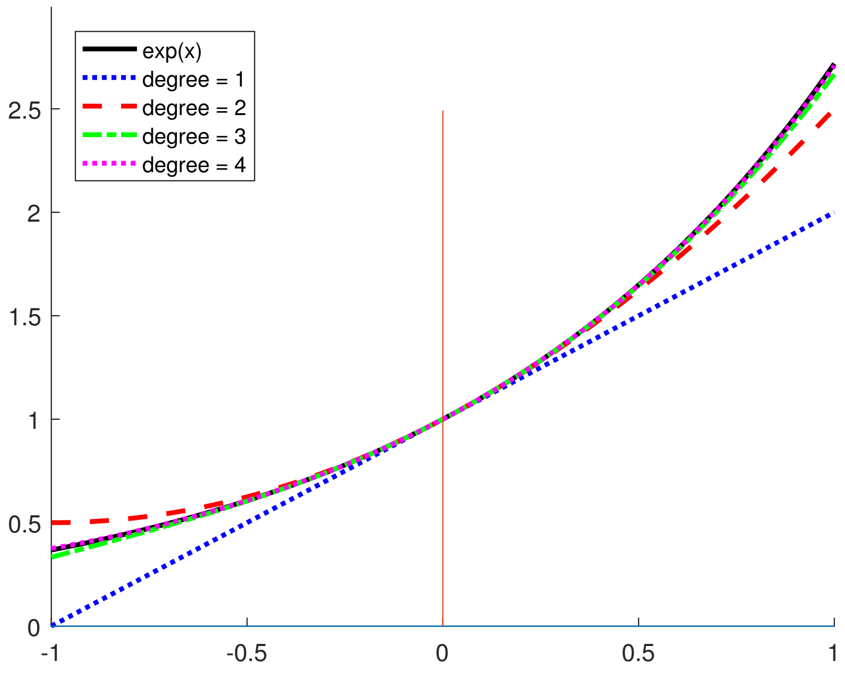

Fig. 1: Taylor approximations of $e^x$

**Example 1.3.** Consider the function $f(x) = e^x$. Figure 1 shows the approximations of $f$ with

Taylor polynomials of degree $1, 2, 3$ and $4$, for $x \in [-1, 1]$. The errors of these approximations, $\max\limits_{x\in[-1,1]} \{e^x - p_n(x)\}$, are graphed in Figure 2.

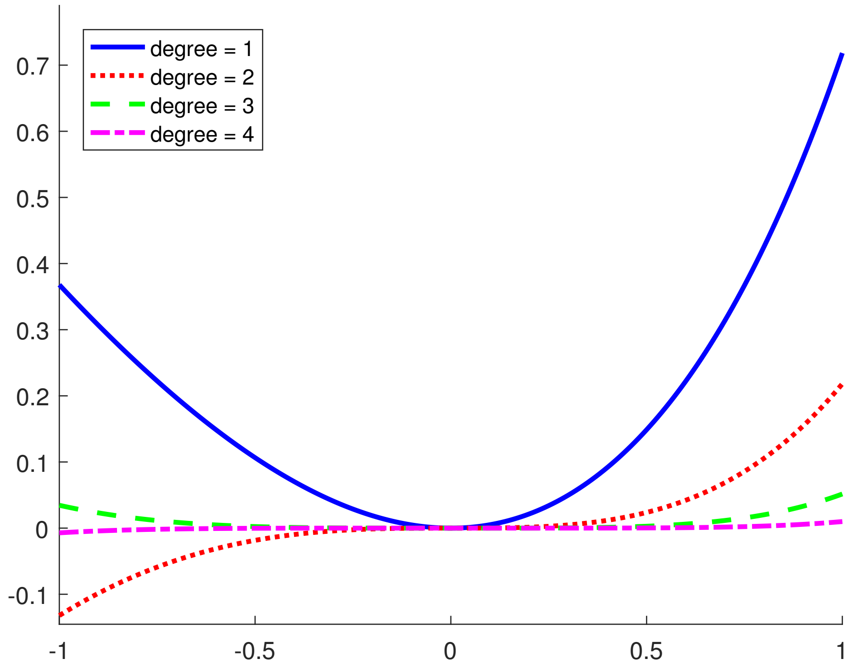

Fig. 2: Errors in Taylor approximations of $e^x$

## Taylor's formula in two dimensions

**Theorem 1.4.** *Let $f : D \subset \mathbb{R}^2 \to \mathbb{R}^2$ be a function which is $n + 1$ times continuously differentiable on $D$, for some $n \geq 0$ and let $(x, y), (x_0, y_0) \in D$. Then*

$$f(x, y) = p_n(x, y) + R_{n+1}(x, y), \tag{1.4}$$

*where*

$$\begin{aligned} p_n(x,y) &= f(x_0,y_0) + \frac{x-x_0}{1!}f'_x(x_0,y_0) + \frac{y-y_0}{1!}f'_y(x_0,y_0) \\ &+ \frac{1}{2!}\Big[(x-x_0)^2 f''_{x^2}(x_0,y_0) + 2(x-x_0)(y-y_0)f''_{xy}(x_0,y_0) + (y-y_0)^2 f''_{y^2}(x_0,y_0)\Big] \\ &+ \sum_{j=1}^{n}\frac{1}{j!}\left[(x-x_0)\frac{\partial}{\partial x} + (y-y_0)\frac{\partial}{\partial y}\right]^j f(x,y)\Bigg|_{\substack{x=x_0\\y=y_0}} \end{aligned} \tag{1.5}$$

*and*

$$R_{n+1}(x,y) = \frac{1}{(n+1)!}\left[(x-x_0)\frac{\partial}{\partial x} + (y-y_0)\frac{\partial}{\partial y}\right]^{n+1} f(x,y)\Bigg|_{\substack{x=\xi\\y=\eta}}, \tag{1.6}$$

*with $(\xi,\eta)$ a point on the line segment determined by the points $(x,y)$ and $(x_0,y_0)$.*

# 2 Errors: Sources, Propagation, Analysis

## 2.1 Types of Numerical Problems

Let us first consider the following simple examples:

**Example 2.1.** Compute

$$\int_1^3 \frac{1}{x}dx.$$

**Solution** Its exact value is

$$\int_1^3 \frac{1}{x}dx = \ln x\Big|_1^3 = \ln 3 - \ln 1 = \ln 3$$

and its approximation is

$$\int_1^3 \frac{1}{x}dx = 1.0986...$$

■

**Example 2.2.** Solve the equation

$$x^2 = 3, x > 0.$$

**Solution** The true solution is

$$x = \sqrt{3},$$

with approximating value

$$x = 1.732...$$

■

**Example 2.3.** Consider the data

| $x_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $y_i$ | 5 | 13 | 16 | 23 | 33 | 38 | 40 |

Discuss the nature of the relationship between $x$ and $y$.

**Solution** We plot the data (see Figure 3).

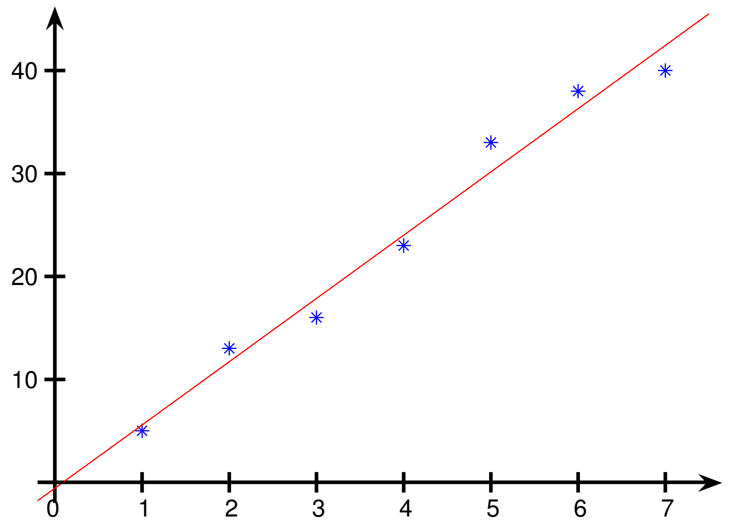

Fig. 3: Scatterplot and linear function

Notice that a straight line "best" approximates the data. So based on these values, we seek a relationship between $x$ and $y$ of the form

$$f(x) = ax + b.$$

In the next chapters, we will analyze several rigorous procedures for approximating scattered data.

■

From these examples, we see that a numerical problem is, in general, of the form

$$f(x) = y.$$

- If $x$ and $f$ are given, and we seek $y$, then this is called a **direct problem** or an **evaluation problem** (Example 2.1).
- If $y$ and $f$ are given, and we want to find $x$, then it is called an **inverse problem** (Example 2.2).
- If $x$ and $y$ are given, and $f$ must be determined, then we have an **identifying problem** (Example 2.3).

For each type of problem, we obtain an *approximating* value, which is affected by *errors*, perturbations from the true value. The study of errors and their propagation is an important task in Numerical Analysis. In practical problems, it is important that the approximations obtained have *small (negligible)* errors, that do not affect the overall precision of the numerical procedure.

## 2.2 Sources and Propagation of Errors

Let $\mathcal{A} : \mathbb{R} \to \mathcal{P}(\mathbb{R})$ be a mapping that assigns a set $\mathcal{A}(x) \subseteq \mathbb{R}$ to each real number $x \in \mathbb{R}$.

**Definition 2.4.** *Let $x \in \mathbb{R}$. The number $x^*$ is called an* ***approximation*** *of $x \in \mathbb{R}$, if $x^* \in \mathcal{A}(x)$ (notation $x \approx x^*$. The mapping $\mathcal{A}$ is called an* ***approximating procedure (method)***

In practice, $\mathcal{A}(x)$ consists of numbers that vary “little” from $x$.

Let $x \in \mathbb{R}$ and $x^* \in \mathcal{A}(x)$ be an approximation of the true (exact) value $x$.

**Definition 2.5.** *The number*

$$\Delta x = x - x^* \tag{2.1}$$

*is called the* ***error*** *of the approximation $x^*$.*

If $\Delta x > 0$, then $x^*$ is an **under-approximation**, and if $\Delta x < 0$, $x^*$ is an **over-approximation**.

For example, for the number $\pi$ = 3.141592..., the number $x^* = 3.14$ is an under-approximation, while $x^* = 3.142$ is an over-approximation.

**Definition 2.6.** *The value*

$$|\Delta x| = |x - x^*| \tag{2.2}$$

*is called* **absolute error** *and the quantity*

$$\delta x = \frac{|\Delta x|}{|x|}, x \neq 0 \tag{2.3}$$

*is called* **relative error** .

Since in practice $x$ is unknown, we use instead

$$\delta x = \frac{|\Delta x|}{|x^*|}. \tag{2.4}$$

For the relative error in $\mathbb{R}$, one can use

$$\delta x = \frac{\Delta x}{x}, \tag{2.5}$$

from which we get

$$\Delta x = x\delta x \Longrightarrow x^* - x = x\delta x,$$

or

$$x^* = (1 + \delta x)x, \tag{2.6}$$

which is widely used in applications.
For the absolute and relative error, one can also use the notations $\Delta x^*$ and $\delta x^*$.

**Sources of error**

- *Mathematical modeling of a physical problem.* A mathematical model for a physical situation is an attempt to give mathematical relationships between certain quantities of physical interest. Because of the complexity of physical reality, a variety of simplifying assumptions are used to construct a more tractable mathematical model. The resulting model has limitations on its accuracy as a consequence of these assumptions, and these limitations may or may not be troublesome, depending on the uses of the model.

- *Blunders (human errors).* In pre-computer times, chance arithmetic errors were always a serious problem. With the introduction of digital computers, the type of blunder has changed. Chance arithmetic errors (e.g., computer malfunctioning) are now relatively rare, and programming errors are currently the main difficulty. Often a program error will be repeated many times in the course of executing the program, and its existence becomes obvious because of absurd numerical output (although the source of the error may still be difficult to find). This makes good program debugging very important, even though it may not seem very rewarding immediately.

- *Uncertainty in physical data.* Most data from a physical experiment will contain error or uncertainty within it. This must affect the accuracy of any calculations based on the data, limiting the accuracy of the answers.
- *Machine errors.* This means the errors inherent in using the floating-point representation of numbers. Specifically, we mean the rounding/chopping errors and the underflow/overflow errors. The rounding/chopping errors are due to the finite length of the floating-point mantissa; and these errors occur with all of the computer arithmetic operations.
- *Mathematical truncation error.* This name refers to the error of approximation in numerically solving a mathematical problem, and it is the error generally associated with the subject of Numerical Analysis. It involves the approximation of infinite processes by finite ones, replacing noncomputable problems with computable ones, etc.

## 2.3 Propagation of Errors

We distinguish two types of problems:

**Case 1**. Given the errors in the approximations of input data, find the errors in the output. We start with the case of a function of two variables $f : D \to \mathbb{R}$, $D = \{(x, y)|x, y \in \mathbb{R}\}$. Let $(x^*, y^*)$ be the approximating values of $(x, y)$. Their absolute errors are then

$$\begin{aligned}\Delta x &= x - x^* \Rightarrow x = x^* + \Delta x,\\ \Delta y &= y - y^* \Rightarrow y = y^* + \Delta y.\end{aligned}$$

We want to compute the absolute error

$$\Delta f = f(x, y) - f(x^*, y^*) \tag{2.7}$$

and the relative error $\delta f$.

We use Taylor's formula in two variables (1.5), at $x^*$ and $y^*$. We have:

$$\begin{aligned} f(x^*+\Delta x, y^*+\Delta y) &= f(x^*,y^*) + \Delta x f'_x(x^*,y^*) + \Delta y f'_y(x^*,y^*) \\ &+ \frac{(\Delta x)^2}{2!} f''_{x^2}(x^*,y^*) + 2\frac{\Delta x \Delta y}{2!} f''_{xy}(x^*,y^*) \\ &+ \frac{(\Delta y)^2}{2!} f''_{y^2}(x^*,y^*) + \ldots, \end{aligned} \tag{2.8}$$

or

$$\begin{aligned} f(x^*+\Delta x, y^*+\Delta y) - f(x^*,y^*) &= \Delta x f'_x(x^*,y^*) + \Delta y f'_y(x^*,y^*) \\ &+ \frac{1}{2!}\left[(\Delta x)^2 f''_{x^2}(x^*,y^*) + 2\Delta x \Delta y f''_{xy}(x^*,y^*)\right. \\ &+ \left.(\Delta y)^2 f''_{y^2}(x^*,y^*)\right] + \ldots \end{aligned} \tag{2.9}$$

From here, we get

$$\begin{aligned} \Delta f &= \Delta x f'_x(x^*,y^*) + \Delta y f'_y(x^*,y^*) \\ &+ \frac{1}{2!}\left[(\Delta x)^2 f''_{x^2}(x^*,y^*) + 2\Delta x \Delta y f''_{xy}(x^*,y^*)\right. \\ &+ \left.(\Delta y)^2 f''_{y^2}(x^*,y^*)\right] + \ldots \end{aligned}$$

If $\Delta x$ and $\Delta y$ are small, then $(\Delta x)^2$, $\Delta x \Delta y$ and $(\Delta y)^2$ can be neglected.

We find the **absolute error of function** $f$ or the **maximum absolute error**:

$$\Delta f \simeq \Delta x f'_x(x^*,y^*) + \Delta y f'_y(x^*,y^*). \tag{2.10}$$

In general, for a function of $n$ variables, we have

$$\begin{aligned} \Delta f &\simeq \Delta x_1 f'_{x_1}() + \Delta x_2 f'_{x_2}() + \cdots + \Delta x_n f'_{x_n}() = \\ &= \sum_{i=1}^{n} \Delta x_i f'_{x_i}() \end{aligned} \tag{2.11}$$

This is the **propagated error**.

Then the relative error $\delta f$ is

$$\begin{aligned}\delta f = \frac{\Delta f}{f} &\simeq \sum_{i=1}^{n} \Delta x_i \frac{f'_{x_i}()}{f} = \sum_{i=1}^{n} \Delta x_i \frac{d}{dx_i} \ln f() = \\ &= \sum_{i=1}^{n} x_i \delta x_i \frac{d}{dx_i} \ln f() = \sum_{i=1}^{n} x_i \frac{d}{dx_i} \ln f() \delta x_i. \end{aligned} \tag{2.12}$$

**Case 2**. The inverse problem is to determine the precision needed in the input data that will guarantee a (given, preset) accuracy in the output data.
To do this, we use the so-called *principle of equal effects*. This assumes that all terms $f'_{x_i}\Delta x_i$ in (2.11) have the same effect, i. e.

$$f'_{x_1}\Delta x_1 = f'_{x_2}\Delta x_2 = \cdots = f'_{x_n}\Delta x_n.$$

Then (2.11) becomes

$$\Delta f \simeq n \Delta x_i f'_{x_i}(),$$

from which it follows that

$$\Delta x_i \simeq \frac{\Delta f}{n f'_{x_i}}, \tag{2.13}$$

or

$$|\Delta x_i| \simeq \frac{|\Delta f|}{n|f'_{x_i}|}. \tag{2.14}$$

Thus, by (2.12), the *relative error* is given by

$$\delta x_i \simeq \frac{|\delta f|}{n \left| x_i \frac{d}{dx_i} \ln f \right|} \tag{2.15}$$

## 3 Floating-Point Representation of Numbers. Significant Digits

**Definition 3.1.** *A number $r$ written in base $b$ (an even number) has the* ***floating-point representation*** *as*

$$r = \pm\, r_0 r_1 \dots r_p \cdot r_{p+1} \dots r_k \times b^e,$$

*where $r_0, r_1, \dots, r_k$ form the* ***mantissa (significand),*** *$e$ is the* ***exponent*** *and $b$ is the* ***base***.

In order to have uniqueness of the representation, the floating-point numbers are *normalized*, that is, we change the representation, not the value, such that $r_0 \neq 0$. The term *floating-point* *number* will be used to mean a real number that can be exactly represented in this format.

**Definition 3.2.** *The* ***significant digits*** *of a floating-point number written in base $b$ are any of the digits $1, 2, \ldots, b - 1$ in its representation that are non-zero, or located between non-zero digits, or preceded by at least one non-zero digit.*

So $0$ can be a significant digit if it does more than just indicate the floating decimal point or fills the places of unknown or omitted digits.
The leftmost significant digit is called the *most significant* digit.

For example,

- the number $7063$ has all significant digits;
- the number $0.02340$ – the last $4$ digits are significant (the zeros in front of $2$ merely indicate the decimal point, and are, therefore, *not* significant);
- $1.230 \times 10^3$ = $1230$ – zero *is* a significant digit, as it is preceded by a nonzero digit.

Consider the number $r > 0$ with the following representation in base $10$:

$$r = r_0 10^k + r_1 10^{k-1} + \cdots + r_{n-1} 10^{k-n+1} + r_n 10^{k-n} + \ldots.$$

**Definition 3.3.** *The number*

$$r^* = r_0^* 10^k + r_1^* 10^{k-1} + \cdots + r_{n-1}^* 10^{k-n+1}$$

*approximates $r$* **correctly with $n$ significant digits** *if*

$$|\Delta r^*| \leq \frac{1}{2} \times 10^{k-n+1} \tag{3.1}$$

**Example 3.4.** Find the number of significant digits with which $e^* = 2.718282$ approximates correctly the number $e = 2.71828182\ldots$.

**Solution** Let $e^*$ = 2.718282. This can be written as

$$e^* = 2 \cdot 10^0 + 7 \cdot 10^{-1} + 1 \cdot 10^{-2} + \cdots$$

So $k = 0$. Then

$$\Delta e^* = e^* - e = 0.00000018 = 0.18 \times 10^{-6} \leq \frac{1}{2} \times 10^{-6}.$$

Comparing it to (3.1), we get

$$10^{k-n+1} = 10^{-6} \Rightarrow 0 - n + 1 = -6 \Rightarrow n = 7.$$

Thus, the number $e^* = 2.718282$ approximates correctly $e = 2.71828182\ldots$ with 7 significant digits.

■

The following relation exists between the number de significant digits and the relative error:

$$\delta r^* \leq \frac{1}{r_0^* \times 10^{n-1}}, \tag{3.2}$$

where $r^*$ is the correct approximation with $n$ significant digits, with the normalized representation

$$r^* = r_0^* 10^k + r_1^* 10^{k-1} + \cdots + r_{n-1}^* 10^{k-n+1}. \tag{3.3}$$

This can be considered the *maximum relative error*.

**Example 3.5.** Let $r^*$ = $0.0875$ be a correct approximation with $3$ significant digits of a number $r$. Find the maximum relative error of this approximation.

**Solution** The number $r^*$ can be written as

$$r^* = 8 \cdot 10^{-2} + 7 \cdot 10^{-3} + 5 \cdot 10^{-4} = 8.75 \cdot 10^{-2} = 0.0875 \cdot 10^{-4},$$

from which we have $r_0^* = 8$ and $n = 3$.

The maximum relative error is

$$\frac{1}{r_0^* 10^{n-1}} = \frac{1}{8 \cdot 10^2} = \frac{1}{8} \cdot 10^{-2} = 0.125 \cdot 10^{-2} = 0.00125$$

Thus, the number $r$ can be approximated correctly by the number $r^* = 0.0875$, with 3 significant digits , with a maximum relative error of $0.125\ \%$.

■

# 4 Divided and Finite Differences

These are expressions that are helpful in writing, computing and implementing various iterative numerical procedures.

## 4.1 Divided Differences

### 4.1.1 Definition and computation

**Definition 4.1.** *Let $f : [a, b] \to \mathbb{R}$ be a differentiable function on $[a, b]$, and $x_i \in [a, b]$, $i = \overline{0, n}$, be $n + 1$ distinct nodes. The quantity*

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}. \tag{4.1}$$

*is called the* ***first-order divided difference*** *of $f$ at the nodes $x_0$ and $x_1$.*

**Remark 4.2.**
1. An alternative notation is $[x_0, x_1; f]$.
2. The first-order divided difference of a function can be thought of as a *discrete* version of the derivative.
3. If we consider $f[x_0] = f(x_0)$ the *divided difference of order* 0 at $x_0$, then $(4.1)$ can be written as

$$f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0}. \tag{4.2}$$

We define higher-order divided differences recursively using lower-order ones.

**Definition 4.3.** *The* ***divided difference of order*** *$n$ of $f$ at the distinct nodes $x_0, x_1, \dots, x_n$ is the quantity*

$$f[x_0, x_1, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}. \tag{4.3}$$

**Remark 4.4.**
1. The denominator in (4.3) is the difference between the uncommon nodes of the differences at the numerator.
2. For easy computation (and implementation) of divided differences, we generate the *table of divided differences*, illustrated below for $4$ nodes. The divided differences are obtained on the first row.

$$\begin{tikzcd}
x_0 & f[x_0] \arrow[r] & f[x_0,x_1] \arrow[r] & f[x_0,x_1,x_2] \arrow[r] & f[x_0,x_1,x_2,x_3] \\
x_1 & f[x_1] \arrow[r] \arrow[ur] & f[x_1,x_2] \arrow[r] \arrow[ur] & f[x_1,x_2,x_3] \arrow[ur] & \\
x_2 & f[x_2] \arrow[r] \arrow[ur] & f[x_2,x_3] \arrow[ur] & & \\
x_3 & f[x_3] \arrow[ur] & & &
\end{tikzcd}$$

**Example 4.5.** Let $f(x) = \sin \pi x$, and the nodes $x_0 = 0, x_1 = \frac{1}{6}, x_2 = \frac{1}{2}$. Let us construct the divided difference table.

**Solution**

$$\begin{tikzcd}
x_0 = 0 & f[x_0] = 0 \arrow[r] & f[x_0,x_1] = \dfrac{1/2-0}{1/6-0} = 3 \arrow[r] & f[x_0,x_1,x_2] = \dfrac{3/2-3}{1/2-0} = -3 \\
x_1 = 1/6 & f[x_1] = 1/2 \arrow[r] \arrow[ur] & f[x_1,x_2] = \dfrac{1-1/2}{1/2-1/6} = 3/2 \arrow[ur] & \\
x_2 = 1/2 & f[x_2] = 1 \arrow[ur] & &
\end{tikzcd}$$

■

**Divided differences with multiple nodes**

Divided differences with multiple nodes can be expressed in terms of the derivatives of the function $f$, as follows

$$f[x_0, x_0] = \lim_{x_1 \to x_0} f[x_0, x_1] = \lim_{x_1 \to x_0} \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(x_0),$$

In general, the **divided difference of order** $n$ at the node $x_0$, of multiplicity $n+1$, is defined as

$$f[x_0, x_0, \ldots, x_0] \;=\; \frac{f^{(n)}(x_0)}{n!}, \tag{4.4}$$

and further, for mixed nodes (some simple, some multiple), we use definition 4.3.

**Example 4.6.** Let us see the divided difference table for 3 double nodes.

**Solution** We have the nodes $x_0, x_1, x_2$ and the values $f(x_i), f'(x_i),\ i = 0, 1, 2$ . We define the sequence of nodes $z_0, z_1, \dots, z_5$ by

$$z_{2i} = z_{2i+1} = x_i, i = 0, 1, 2.$$

We build the divided difference table relative to the nodes $z_i, i = \overline{0, 5}$.
Since $z_{2i} = z_{2i+1} = x_i$ for every $i = 0, 1, 2$, $f[z_{2i}, z_{2i+1}] = f[x_i, x_i]$ is a divided difference with a double node and it is equal to $f'(x_i)$; therefore we will use $f'(x_0), f'(x_1), f'(x_2)$ instead of first order divided differences $f[z_0, z_1], f[z_2, z_3], f[z_4, z_5]$.

$$\begin{tikzcd}
z_0 = x_0 & f[z_0] \arrow[r] & f[z_0, z_1] = f'(x_0) \arrow[r] & f[z_0, z_1, z_2] & \dots \\
z_1 = x_0 & f[z_1] \arrow[r] \arrow[ur] & f[z_1, z_2] = \dfrac{f(z_2) - f(z_1)}{z_2 - z_1} \arrow[r] \arrow[ur] & f[z_1, z_2, z_3] & \dots \\
z_2 = x_1 & f[z_2] \arrow[r] \arrow[ur] & f[z_2, z_3] = f'(x_1) \arrow[r] \arrow[ur] & f[z_2, z_3, z_4] & \dots \\
z_3 = x_1 & f[z_3] \arrow[r] \arrow[ur] & f[z_3, z_4] = \dfrac{f(z_4) - f(z_3)}{z_4 - z_3} \arrow[r] \arrow[ur] & f[z_3, z_4, z_5] & \dots \\
z_4 = x_2 & f[z_4] \arrow[r] \arrow[ur] & f[z_4, z_5] = f'(x_2) \arrow[ur] & & \\
z_5 = x_2 & f[z_5] \arrow[ur] & & &
\end{tikzcd}$$

■

### 4.1.2 Properties of divided differences

**Theorem 4.7.** *Divided differences have a number of special properties that can simplify work with them:*
a)

$$f[x_0, x_1, \dots, x_n] = \sum_{i=0}^{n} \frac{f(x_i)}{u'(x_i)} = \sum_{i=0}^{n} \frac{f(x_i)}{u_i(x_i)}, \tag{4.5}$$

*where* $u(x) = (x - x_0)(x - x_1)\dots(x - x_n)$ *and* $u_i(x) = \dfrac{u(x)}{x - x_i}$.

b) *For any permutation* $\{i_0, i_1, \dots, i_n\}$ *of the integers* $\{0, 1, \dots, n\}$,

$$f[x_{i_0}, x_{i_1}, \dots, x_{i_n}] = f[x_0, x_1, \dots, x_n]. \tag{4.6}$$

c)

$$f[x_0, x_1, \dots, x_n] = \frac{(Wf)(x_0, x_1, \dots, x_n)}{V(x_0, x_1, \dots, x_n)}, \tag{4.7}$$

*where*

$$Wf(x_0, x_1, \dots, x_n) = \begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} & f(x_0) \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} & f(x_1) \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} & f(x_n) \end{vmatrix} \tag{4.8}$$

*and* $V(x_0, x_1, \dots, x_n)$ *is the Vandermonde determinant.*

d) *Let* $e_k(x) = x^k, k \geq 0$. *Then,*

$$e_k[x_0, x_1, \dots, x_n] = \begin{cases} 0, & k < n \\ 1, & k = n \end{cases}.$$

*For a polynomial of degree* $k$, $P_k = a_0 + a_1 x + \dots + a_k x^k$,

$$P_k[x_0, x_1, \dots, x_n] = \begin{cases} 0, & k < n \\ a_n, & k = n \end{cases}. \tag{4.9}$$

e) *If* $f \in C^n[a, b]$, *where* $[a, b]$ *is the smallest interval containing the distinct nodes* $\{x_0, \dots, x_n\}$, *then there exists* $\xi_n \in (a, b)$ *such that*

$$f[x_0, x_1, \dots, x_n] = \frac{1}{n!} f^{(n)}(\xi_n), \tag{4.10}$$

*(the mean-value formula for divided differences).*

*Proof.* (Selected)

a) First off, let us notice that

$$u'(x) \;=\; \sum_{i=0}^{n} u_i(x)$$

and $u_i(x_j) = 0, i \neq j$, from where it follows that

$$u'(x_i) \;=\; u_i(x_i),\; i = 0, 1, \ldots, n.$$

We prove (4.5) by induction on $n$. Let us look at several cases.

Although it's a convention, let us start with $\boldsymbol{n = 0}$. In this case,

$$\begin{aligned} u(x) &= x - x_0, \\ u_0(x) &= 1,\; u_0(x_0) \;=\; 1, \\ u'(x) &= 1,\; u'(x_0) \;= 1 \end{aligned}$$

and

$$f[x_0] \;= f(x_0) \;=\; \frac{f(x_0)}{u_0(x_0)} \;= \sum_{i=0}^{0} \frac{f(x_i)}{u_i(x_i)},$$

so (4.5) is trivially true.

$\boldsymbol{n = 1}$. We have

$$\begin{aligned} u(x) &= (x - x_0)(x - x_1), \\ u_0(x) &= x - x_1,\; u_0(x_0) \;=\; x_0 - x_1, \\ u_1(x) &= x - x_0,\; u_1(x_1) \;=\; x_1 - x_0, \\ u'(x) &= (x - x_1) + (x - x_0), \\ u'(x_0) &= x_0 - x_1 \;=\; u_0(x_0), \\ u'(x_1) &= x_1 - x_0 \;=\; u_1(x_1). \end{aligned}$$

The first-order divided difference is

$$\begin{aligned} f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} \;=\; \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ &= \frac{f(x_0)}{x_0 - x_1} + \frac{f(x_1)}{x_1 - x_0} \;=\; \frac{f(x_0)}{u_0(x_0)} + \frac{f(x_1)}{u_1(x_1)}. \end{aligned}$$

To better understand the induction step, let us consider one more case, $n = 2$.

$\boldsymbol{n = 2}$. Now, $u(x) = (x - x_0)(x - x_1)(x - x_2)$. The second-order divided difference can be written as

$$\begin{aligned}
f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\
&= \frac{1}{x_2 - x_0}\left[\frac{f(x_1)}{x_1 - x_2} + \frac{f(x_2)}{x_2 - x_1} - \frac{f(x_0)}{x_0 - x_1} - \frac{f(x_1)}{x_1 - x_0}\right] \\
&= \frac{f(x_0)}{u_0(x_0)} + \frac{f(x_2)}{u_2(x_2)} + \frac{f(x_1)}{x_2 - x_0}\left[\frac{1}{x_1 - x_2} - \frac{1}{x_1 - x_0}\right] \\
&= \frac{f(x_0)}{u_0(x_0)} + \frac{f(x_2)}{u_2(x_2)} + \frac{f(x_1)}{x_2 - x_0} \cdot \frac{x_1 - x_0 - x_1 + x_2}{u_1(x_1)} \\
&= \sum_{i=0}^{2} \frac{f(x_i)}{u_i(x_i)}.
\end{aligned}$$

Assume (4.5) is true for $\boldsymbol{n - 1}$. For the case $\boldsymbol{n}$, we have

$$u(x) = (x - x_0)(x - x_1)\dots(x - x_{n-1})(x - x_n),$$

and to simplify the writing, for $u_i(x)$ we use the notation

$$u_i(x) = (x - x_0)\dots \cdot / \cdot \dots (x - x_n).$$

Then,

$$\begin{aligned}
f[x_0, \dots, x_n] &= \frac{f[x_1, \dots, x_n] - f[x_0, \dots, x_{n-1}]}{x_n - x_0} \\
&= \frac{1}{x_n - x_0}\left[\sum_{i=1}^{n-1} \frac{f(x_i)}{(x_i - x_1)\dots \cdot / \cdot \dots (x_i - x_n)} + \frac{f(x_n)}{(x_n - x_1)\dots(x_n - x_{n-1})}\right. \\
&\qquad \left. - \frac{f(x_0)}{(x_0 - x_1)\dots(x_0 - x_{n-1})} - \sum_{i=1}^{n-1} \frac{f(x_i)}{(x_i - x_0)\dots \cdot / \cdot \dots (x_i - x_{n-1})}\right] \\
&= \frac{f(x_0)}{u_0(x_0)} + \frac{f(x_n)}{u_n(x_n)} + \frac{1}{x_n - x_0}\sum_{i=1}^{n-1} \frac{f(x_i)}{u_i(x_i)} \cdot (x_i - x_0 - x_i + x_n),
\end{aligned}$$

hence,

$$f[x_0,\dots,x_n] \;=\; \sum_{i=0}^{n}\frac{f(x_i)}{u_i(x_i)}.$$

b) This follows directly from part a), since the right-hand-side of (4.5) is symmetric with respect to the indexes $\{1,2,\dots,n\}$.
c) Let us recall that for Vandermonde determinants, the following are true:

$$V(x_0,x_1,\dots,x_n) \;=\; \begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{vmatrix} \;=\; \prod_{0\le j<i\le n}(x_i-x_j)$$

and

$$V(x_0,\dots,/,\dots,x_n) \;=\; \frac{V(x_0,x_1,\dots,x_n)}{(x_i-x_0)\dots/\dots(x_i-x_n)} \;=\; (-1)^{n+i}\frac{V(x_0,x_1,\dots,x_n)}{u_i(x_i)}.$$

The idea for the proof of (4.7) is to expand $(Wf)$ over the elements of the last column, to get

$$\begin{aligned}(Wf)(x_0,x_1,\dots,x_n) \;&=\; V(x_0,x_1,\dots,x_n)\sum_{i=0}^{n}\frac{f(x_i)}{u_i(x_i)} \\ &=\; V(x_0,x_1,\dots,x_n)f[x_0,\dots,x_n],\end{aligned}$$

by part a).
d) For $f(x)=e_k(x)=x^k$, we have

$$(We_k)(x_0,x_1,\dots,x_n) \;=\; \begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} & x_0^k \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} & x_1^k \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} & x_n^k \end{vmatrix} \;=\; \begin{cases} 0, & k<n \\ V(x_0,x_1,\dots,x_n), & k=n \end{cases},$$

so, by part c),

$$e_k[x_0,x_1,\dots,x_n] \;=\; \begin{cases} 0, & k<n \\ 1, & k=n \end{cases}.$$

Then, for $P_k = a_0 + a_1 x + \cdots + a_k x^k$,

$$P_k[x_0, x_1, \ldots, x_n] = (a_0 + a_1 x + \cdots + a_k x^k)[x_0, x_1, \ldots, x_n] = \begin{cases} 0, & k < n \\ a_n, & k = n \end{cases}.$$

□

**Remark 4.8.** As a consequence of part e), if $f \in C^n[a, b]$ and $\alpha \in [a, b]$, then

$$\lim_{x_0, \ldots, x_n \to \alpha} f[x_0, x_1, \ldots, x_n] = \lim_{\xi_n \to \alpha} \frac{f^{(n)}(\xi_n)}{n!} = \frac{1}{n!} f^{(n)}(\alpha),$$

the computational formula for divided differences with multiple nodes.

## 4.2 Finite Differences

**Definition 4.9.** *Consider the equidistant nodes $x_i = x_0 + ih, i = 0, 1, \ldots, n,\ h > 0$.*
*The quantity*

$$\Delta^1 f(x_i) = f(x_{i+1}) - f(x_i) = f_{i+1} - f_i \tag{4.11}$$

*is called the **first-order forward difference** of $f$ with step $h$ at $x_i$, and*

$$\Delta^k f(x_i) = \Delta^{k-1} f(x_{i+1}) - \Delta^{k-1} f(x_i) = \Delta^{k-1} f_{i+1} - \Delta^{k-1} f_i \tag{4.12}$$

*is the $k$**th-order forward difference** of $f$ with step $h$, at $x_i$.*

**Remark 4.10.**
1. As a convention, we use $\Delta^0 f(x_i) = f(x_i) = f_i$.
2. For easy computation (and implementation) of forward differences, we construct a *table of forward differences*, similar to the one used for divided differences, illustrated below for $4$ nodes.

$$\begin{tikzcd}
x_0 & f_0 \arrow[r] & \Delta f_0 \arrow[r] & \Delta^2 f_0 \arrow[r] & \Delta^3 f_0 \\
x_1 & f_1 \arrow[r] \arrow[ur] & \Delta f_1 \arrow[r] \arrow[ur] & \Delta^2 f_1 \arrow[ur] & \\
x_2 & f_2 \arrow[r] \arrow[ur] & \Delta f_2 \arrow[ur] & & \\
x_3 & f_3 \arrow[ur] & & &
\end{tikzcd}$$

In a similar way, we define the **backward difference** $\nabla$ by

$$\begin{aligned}
\nabla^0 f_i &= f_i, \\
\nabla^1 f_i &= f_i - f_{i-1}, \\
\nabla^k f_i &= \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1},
\end{aligned} \tag{4.13}$$

and they can also be easily computed in a table.

$$\begin{tikzcd}
x_0 & f_0 \arrow[dr] & & & \\
x_1 & f_1 \arrow[r] \arrow[dr] & \nabla f_1 \arrow[dr] & & \\
x_2 & f_2 \arrow[r] \arrow[dr] & \nabla f_2 \arrow[r] \arrow[dr] & \nabla^2 f_2 \arrow[dr] & \\
x_3 & f_3 \arrow[r] & \nabla f_3 \arrow[r] & \nabla^2 f_3 \arrow[r] & \nabla^3 f_3
\end{tikzcd}$$

**Remark 4.11.** These differences are referred to collectively as *finite differences*. Usually, if nothing is specified, by "finite" differences we mean "forward" differences.

Denote by

$$X = \{x_i \mid x_i = x_0 + ih,\ i = \overline{0,n}, x_0, h \in \mathbb{R}\}$$

and for $f : X \to \mathbb{R}$, by $f_i = f(x_i)$.

Let us write a few finite differences and notice a pattern.

$$\begin{aligned}
\Delta^1 f(x_i) &= f_{i+1} - f_i, \\
\Delta^2 f(x_i) &= \Delta^1 f_{i+1} - \Delta^1 f_i = f_{i+2} - f_{i+1} - \big(f_{i+1} - f_i\big) = f_{i+2} - 2f_{i+1} + f_i, \\
\Delta^3 f(x_i) &= \Delta^2 f_{i+1} - \Delta^2 f_i = f_{i+3} - 2f_{i+2} + f_{i+1} - \big(f_{i+2} - 2f_{i+1} + f_i\big) \\
&= f_{i+3} - 3f_{i+2} + 3f_{i+1} - f_i.
\end{aligned}$$

It can easily be proved (by induction) that

$$\Delta^n f(x_i) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f_{n-k+i},$$

or, equivalently, by the symmetry of combinations,

$$\Delta^n f(x_i) = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} f_{k+i}.$$

In particular, we have

$$\Delta^n f(x_0) = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} f_k. \tag{4.14}$$

Finite and divided differences for equally spaced nodes are closely related.

**Proposition 4.12.** *Let* $f : X \to \mathbb{R}$. *Then*

$$f[a, a+h, \ldots, a+nh] = \frac{1}{n!h^n} \Delta^n f(a). \tag{4.15}$$

# Chapter 2. Function Approximation

Approximation of functions is one of the most important tasks in Numerical Analysis.

Most functions encountered in mathematical problems and applications cannot be evaluated exactly, even though we usually handle them as if they where completely known quantities. The simplest and most important of these are $\sqrt{x}, e^x, \log x$, and the trigonometric functions; and there are many other functions that occur commonly in physics, engineering, and other disciplines. In evaluating functions, by hand or using a computer, we are essentially limited to the elementary arithmetic operations $+, -, \times$ and $\div$. Combining these operations means that we can evaluate polynomials and rational functions, which are polynomials divided by polynomials. All other functions must be evaluated by using approximations based on polynomials or rational functions, including piecewise variants of them (e.g., spline functions). Although rational functions generally give slightly more efficient approximations, polynomials are adequate for most problems and their theory is much easier to work with.

**Interpolation** is the process of finding and evaluating a function whose graph goes through a set of given points. The points may arise as measurements in a physical problem, or they may be obtained from a known function. The interpolating function is usually chosen from a restricted class of functions and *polynomials* are the most commonly used class.

Interpolation is an important tool in producing computable approximations to commonly used functions. Moreover, to numerically integrate or differentiate a function, we often replace the function with a simpler approximating expression, and it is then integrated or differentiated. These simpler expressions are almost always obtained by interpolation. Also, some of the most widely used numerical methods for solving differential equations are obtained from interpolating approximations. Finally, interpolation is widely used in computer graphics, to produce smooth curves and surfaces when the geometric object of interest is given at only a discrete set of data points.

## 1 Polynomial Interpolation

**Interpolation problem.** Given $n+1$ distinct points – called *nodes (or knots)* – $x_i \in [a,b], i = \overline{0,n}$ and the values $f(x_i) = y_i$ of an unknown function $f : [a,b] \to \mathbb{R}$ , find a polynomial $P(x)$ of minimum degree, satisfying

$$P(x_i) \quad = \quad f(x_i),\ i = \overline{0,n}, \tag{1.1}$$

called *interpolation conditions*. This polynomial approximates function $f$.

## 1.1 Lagrange Interpolation

### Linear interpolation

We start with a simple case: consider two interpolation nodes, $(x_0, y_0), (x_1, y_1), x_0 \neq x_1$.
We know that there is a unique *line* passing through these points. That means we can find a polynomial of degree $1$ that interpolates the data. Let us find it.

The slope of the line is

$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

and its equation is

$$y - y_0 = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0).$$

We find the linear interpolation polynomial as

$$\begin{aligned}
P_1(x) &= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) \\
&= \left(1 - \frac{x - x_0}{x_1 - x_0}\right) y_0 + \frac{x - x_0}{x_1 - x_0} y_1 \\
&= \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1.
\end{aligned}$$

**Example 1.1.** Consider the function $f(x) = \sqrt{x}$ and the nodes $x_0 = 1, x_1 = 4$, i.e. the data $(1, 1), (4, 2)$.

**Solution** The linear interpolation polynomial is

$$P_1(x) = \frac{1}{3}x + \frac{2}{3}.$$

The graphs of $f$ and $P_1$ on the interval $[0, 15]$ are shown in Figure 1.

■

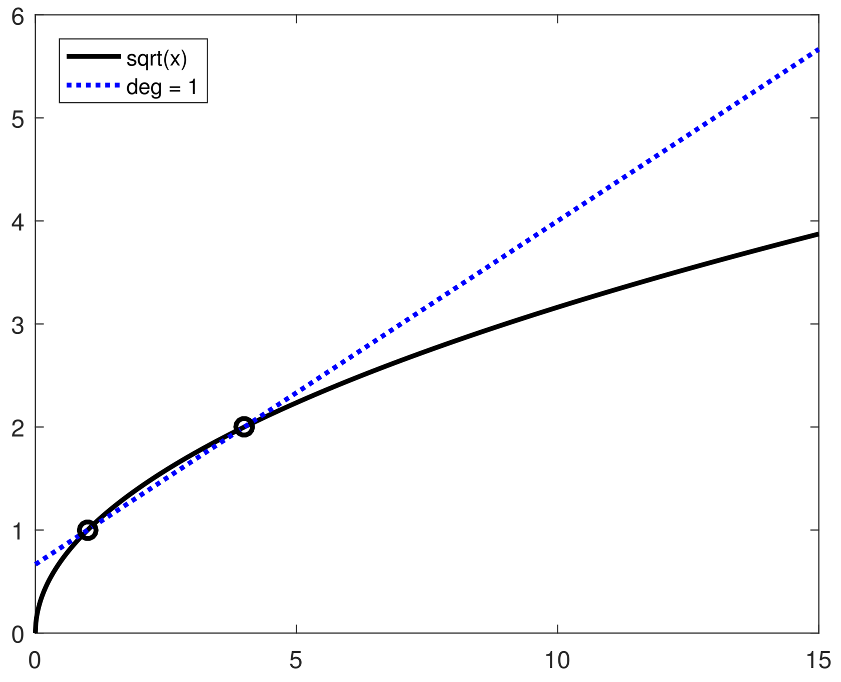

Fig. 1: Linear interpolation of function $\sqrt{x}$

**Quadratic interpolation**

We go further and consider 3 distinct nodes $(x_0, y_0), (x_1, y_1), (x_2, y_2)$. It can easily be checked that the quadratic polynomial

$$P_2(x) \quad = \quad \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2$$

interpolates these data.

**Example 1.2.** In Example 1.1 we add the node $(9, 3)$. The graphs of $f$ and the two interpolation polynomials $P_1$ and $P_2$ are plotted in Figure 2.

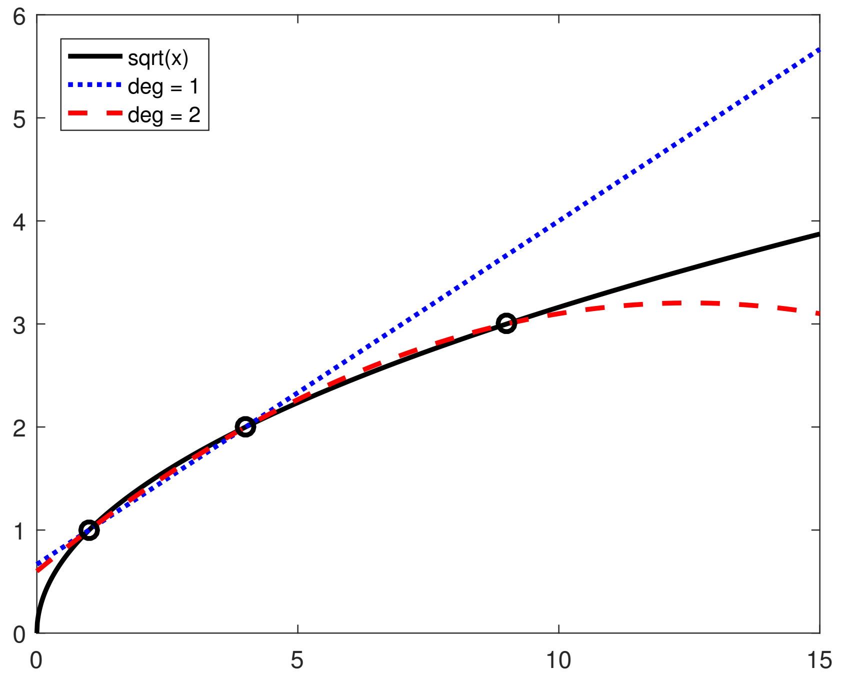

Fig. 2: Linear and quadratic interpolation of function $\sqrt{x}$

## General case

Consider the interval $[a, b] \subset \mathbb{R}$, a function $f : [a, b] \to \mathbb{R}$ and a set of $n + 1$ distinct nodes $\{x_0, x_1, \ldots, x_n\} \subset [a, b]$.
Recall the notations

$$\begin{aligned} u(x) &= \prod_{j=0}^{n}(x - x_j), \\ u_j(x) &= \frac{u(x)}{x - x_j},\ j = 0, 1, \ldots, n. \end{aligned} \tag{1.2}$$

**Theorem 1.3.** *There is a unique polynomial $L_n f$ of degree at most $n$, satisfying the interpolation conditions* (1.1)*. This polynomial can be written as*

$$L_n f(x) = \sum_{i=0}^{n} l_i(x) f(x_i), \tag{1.3}$$

*where*

$$l_i(x) \;=\; \prod_{\substack{j=0\\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j} \;=\; \frac{u_i(x)}{u_i(x_i)} \;=\; \frac{u_i(x)}{u'(x_i)}. \tag{1.4}$$

*$L_n f$ is called the **Lagrange interpolation polynomial** of $f$ at the nodes $x_0, x_1, \ldots, x_n$. The functions $l_i(x), i = \overline{0,n}$ are called **Lagrange fundamental (basis) polynomials** associated with these points.*

*Proof.* It can easily be checked that $l_i$ is a polynomial of degree at most $n$ and that

$$l_i(x_j) \;= \delta_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}.$$

Hence, the polynomial $L_n f$ defined in(1.3) is also a polynomial of degree at most $n$ and it satisfies conditions (1.1).

To prove uniqueness, assume there exists another polynomial $P_n^*$ (of degree at most $n$) satisfying conditions (1.1) and consider

$$Q_n \;=\; L_n - P_n^*.$$

By (1.1), $Q_n(x_i) = 0, i = 0, \ldots, n$, which means $Q_n$, a polynomial of degree at most $n$, has $n+1$ distinct roots. By the Fundamental Theorem of Algebra, $Q_n$ must be identically zero, thus proving the uniqueness of $L_n$.

□

## Error and convergence

We want to use the approximation

$$f(x) \;\approx\; L_n f(x),\; x \in [a,b].$$

To this end, we must assess (bound) the error (the remainder)

$$(R_n f)(x) \;=\; f(x) - (L_n f)(x),\; x \in [a,b]. \tag{1.5}$$

**Theorem 1.4.** *Let $[a,b] \subset \mathbb{R}$, $f : [a,b] \to \mathbb{R}$ a function of class $C^{n+1}[a,b]$ and consider the distinct nodes $\{x_0, x_1, \ldots, x_n\} \subset [a,b]$. Then there exists $\xi \in (a,b)$ such that*

$$(R_n f)(x) \;=\; \frac{u(x)}{(n+1)!} f^{(n+1)}(\xi). \tag{1.6}$$

**Example 1.5.** For linear and quadratic interpolation, the remainders are given by

$$\begin{aligned}(R_1f)(x) &= \frac{(x-x_0)(x-x_1)}{2}f''(\xi),\\ (R_2f)(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)}{6}f'''(\xi).\end{aligned}$$

For $f(x)=\sqrt{x}=x^{1/2}$, the derivatives are

$$\begin{aligned}f'(x) &= \frac{1}{2}x^{-1/2} = \frac{1}{2}\cdot\frac{1}{\sqrt{x}},\\ f''(x) &= \frac{1}{2}\left(-\frac{1}{2}\right)x^{-3/2} = -\frac{1}{4}\cdot\frac{1}{x\sqrt{x}},\\ f'''(x) &= -\frac{1}{4}\left(-\frac{3}{2}\right)x^{-5/2} = \frac{3}{8}\cdot\frac{1}{x^2\sqrt{x}}.\end{aligned}$$

So, for the remainders, we have

$$\begin{aligned}|(R_1f)(x)| &= \frac{|(x-x_0)(x-x_1)|}{8}\cdot\frac{1}{\xi\sqrt{\xi}},\\ |(R_2f)(x)| &= \frac{3}{8}\frac{|(x-x_0)(x-x_1)(x-x_2)|}{6}\cdot\frac{1}{\xi^2\sqrt{\xi}} = \frac{|(x-x_0)(x-x_1)(x-x_2)|}{16}\cdot\frac{1}{\xi^2\sqrt{\xi}}.\end{aligned}$$

**Remark 1.6.** In general, an upper bound of the interpolation error is given by

$$|(R_nf)(x)| \le \frac{|u(x)|}{(n+1)!}M_{n+1}(f), \tag{1.7}$$

where

$$M_{n+1}(f) = \sup_{t\in[a,b]}|f^{(n+1)}(t)|.$$

The term $|u(x)|$ can be minimized by a suitable choice of the nodes.
Assume the interval is $[-1,1]$ (then, for a general interval $[a,b]$, we use the linear change of variables $x=\frac{b-a}{2}t+\frac{b+a}{2},\ t\in[-1,1],\ x\in[a,b]$).

An optimal choice of nodes are the roots of the **Chebyshev polynomial of the first kind**:

$$T_m(x) = \cos(m\arccos x). \tag{1.8}$$

With the change of variables $x = \cos t, t \in [0, \pi]$, we get

$$\begin{aligned} T_m(x) &= \cos(mt) = \frac{1}{2}\left(e^{imt} + e^{-imt}\right) \\ &= \frac{1}{2}\left[(\cos t + i \sin t)^m + (\cos t - i \sin t)^m\right] \\ &= \frac{1}{2}\left[\left(x + i\sqrt{1-x^2}\right)^m + \left(x - i\sqrt{1-x^2}\right)^m\right]. \end{aligned}$$

The odd powers of the radical will be canceled, resulting in a polynomial of degree $m$ in $x$, with leading coefficient $2^{m-1}$.

**Remark 1.7.** Chebyshev polynomials of the first kind have some remarkable properties:
1. Polynomials of degree $0, 1, 2$ and $3$ are easily computable using trigonometric identities. They are

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x^2 - 1, \\ T_3(x) &= 4x^3 - 3x. \end{aligned}$$

2. Higher degree polynomials can be obtained from the recurrence relation

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x),\ k = 1, 2, \ldots.$$

For example, the next polynomial is

$$T_4(x) = 8x^4 - 8x^2 + 1.$$

3. $\{T_n(x)\}_{n\in\mathbb{N}}$ is a sequence of *orthogonal* polynomials on $(-1, 1)$ with respect to the weight function $w(x) = \dfrac{1}{\sqrt{1-x^2}}$, i.e.

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}\, dx = 0,\ n \neq m.$$

4. To minimize the term $|u(x)|$ in (1.7) on the interval $[-1,1]$, we choose

$$u(x) = \widetilde{T}_{n+1}(x) = \frac{1}{2^n} T_{n+1}(x),$$

i.e. the nodes

$$x_k \;=\; \cos\frac{2k-1}{2n}\pi,\; k = 0,1,\ldots,n. \tag{1.9}$$

In this case, we have

$$||R_n f|| \;\leq\; \frac{1}{2^n(n+1)!}||f^{(n+1)}||.$$

For the general case, on the interval $[a,b]$, we take

$$u(x) = \widetilde{T}_{n+1}(x;a,b) = \frac{(b-a)^{n+1}}{2^{2n+1}} \cos\left((n+1)\arccos\frac{2x-a-b}{b-a}\right)$$

and for the remainder, we have

$$||R_n f|| \;\leq\; \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!}||f^{(n+1)}||.$$

Regarding the convergence of the Lagrange polynomial $L_n f$ to $f$, this *does not* always happen. The polynomial does converge, if, for instance, $f \in C^\infty[a,b]$, with $|f^k(x)| \leq M_k,\ \forall x \in [a,b], k = 0,1,2,\ldots$ and satisfies

$$\lim_{k\to\infty} \frac{(b-a)^k}{k!} M_k \;=\; 0.$$

**Example 1.8** (**Runge's Example**). Consider the function

$$f(x) \;=\; \frac{1}{1+x^2},\; x \in [-5,5]$$

and the equally spaced nodes

$$x_k^{(n)} \;=\; -5 + 10\,\frac{k}{n},\; k = \overline{0,n}.$$

It can be shown that

$$\lim_{n\to\infty} |f(x) - L_n f(x)| \;=\; \begin{cases} 0, & \text{if} \quad |x| < 3.633\ldots \\ \infty, & \text{if} \quad |x| > 3.633\ldots \end{cases}$$

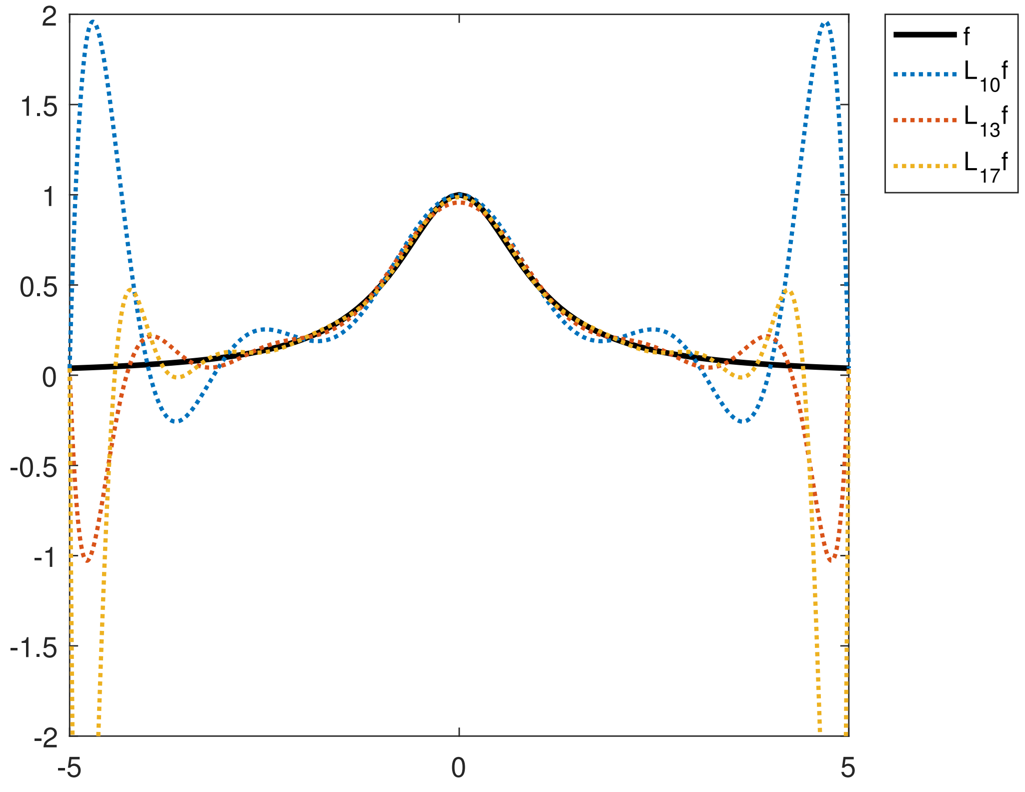

Fig. 3: Runge's Example, $n = 10, 13, 17$

## 1.2 Efficient Computation of Interpolation Polynomials

Recall the Lagrange interpolation polynomial of a function $f$, at the distinct nodes $(x_i, f_i), f_i = f(x_i),\ i = \overline{0, n}$:

$$L_n f(x) \;=\; \sum_{i=0}^{n} l_i(x) f_i, \tag{1.1}$$

where

$$\begin{aligned} l_i(x) \;&=\; \prod_{\substack{j=0\\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j} \;=\; \frac{u_i(x)}{u_i(x_i)} \;=\; \frac{u_i(x)}{u'(x_i)}, \\ u(x) \;&=\; \prod_{j=0}^{n} (x - x_j),\ \ u_j(x) \;=\; \frac{u(x)}{x - x_j},\ j = 0, 1, \dots, n. \end{aligned}$$

This formula is well-suited for many theoretical uses of interpolation, but it is less desirable for practical computations. Among its shortcomings: each evaluation of $L_n f(x)$ requires $O(n^2)$ flops (additions and multiplications); adding a new node $(x_{n+1}, f_{n+1})$ requires a new computation from scratch and knowing $L_n f(x)$ does not lead to a less expensive way to evaluate $L_{n+1} f(x)$. For these reasons, we need alternative and more easily computable formulations and expressions for interpolation polynomials.

### 1.2.1 Barycentric interpolation

The Lagrange formula (1.1) can be rewritten in such a way that it can be evaluated and updated in $O(n)$ flops.

$$L_n f(x) \;=\; \sum_{i=0}^{n} \frac{u_i(x)}{u'(x_i)} f_i \;=\sum_{i=0}^{n} \frac{u(x)}{(x - x_i) u'(x_i)} f_i \;=\; u(x) \sum_{i=0}^{n} \frac{\dfrac{1}{u'(x_i)}}{x - x_i} f_i.$$

Let

$$w_i \;=\; \frac{1}{u'(x_i)} \;=\; \frac{1}{\displaystyle\prod_{\substack{j=0\\ j\neq i}}^{n} (x_i - x_j)},\ i = 0, 1, \dots, n. \tag{1.2}$$

These are called **barycentric weights**. With these, the Lagrange interpolation polynomial can be written as

$$L_n f(x) \;=\; u(x)\sum_{i=0}^{n} \frac{w_i}{x-x_i} f_i. \tag{1.3}$$

Formula (1.3) is called the *first barycentric formula*.

Now, Lagrange interpolation is a formula requiring $O(n^2)$ operations for calculating some quantities independent of $x$, the weights $w_i$, followed by $O(n)$ flops for evaluating $L_n f(x)$, once these numbers are known. Incorporating a new node $x_{n+1}$ entails two calculations:
- dividing each $w_i, i = 0, \ldots, n$ by $x_i - x_{n+1}$, for a cost of $2n + 2$ flops,
- computing a new weight $w_{n+1}$ using formula (1.2), for another $n + 1$ flops.

Formula (1.3) can be improved even further. Notice that for the constant function $f \equiv 1$, the Lagrange polynomial is $f$ itself

$$L_n f \;\equiv\; 1,$$

by the uniqueness of the interpolation polynomial. Substituting in (1.3), we find

$$1 \;=\; u(x)\sum_{i=0}^{n} \frac{w_i}{x-x_i}$$

and further

$$u(x) \;=\; \frac{1}{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x-x_i}}.$$

Then the Lagrange polynomial can be written as

$$L_n f(x) \;=\; \frac{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x-x_i} f_i}{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x-x_i}}, \tag{1.4}$$

called the *second barycentric formula*.

### 1.2.2 Newton-type methods

The next procedures gives an alternative form for the interpolation polynomial, as well as for the remainder.

#### Newton's divided difference formula

To better understand (and write) the transition from $n$ to $n+1$ nodes, we slightly change the notations. For the monic polynomial of the nodes ("monic" means the leading coefficient is $1$), previously denoted by "$u(x)$", we introduce a new notation, one that also emphasizes the *number of nodes* that it refers to. So, let

$$\begin{aligned}\psi_n(x) &= (x-x_0)\dots(x-x_{n-1})(x-x_n),\\ \psi_{n-1}(x) &= (x-x_0)\dots(x-x_{n-1}).\end{aligned}$$

Then we have

$$\begin{aligned}\psi_n(x) &= (x-x_n)\psi_{n-1}(x),\\ \psi_n'(x) &= \psi_{n-1}(x)+(x-x_n)\psi_{n-1}'(x).\end{aligned}$$

Hence,

$$\begin{aligned}\psi_n'(x_i) &= (x_i-x_n)\psi_{n-1}'(x_i),\ i=0,\dots,n-1,\\ \psi_n'(x_n) &= \psi_{n-1}(x_n).\end{aligned}\tag{1.5}$$

With these new notations, we can write

$$\begin{aligned}L_{n-1}f(x) &= \sum_{i=0}^{n-1}\frac{\psi_{n-1}(x)}{(x-x_i)\psi_{n-1}'(x_i)}f_i,\\ L_nf(x) &= \sum_{i=0}^{n}\frac{\psi_n(x)}{(x-x_i)\psi_n'(x_i)}f_i.\end{aligned}\tag{1.6}$$

Let us also recall one of the properties of divided differences (Theorem 4.7 a), in Lecture 2):

$$f[x_0,\dots,x_n] = \sum_{i=0}^{n}\frac{1}{\psi_n'(x_i)}f_i.\tag{1.7}$$

We want to derive a simple recursive formula from $L_{n-1}f$ to $L_nf$, when adding a new node $x_n$. Let

$$Q(x) \;=\; L_nf(x) - L_{n-1}f(x). \tag{1.8}$$

Obviously, $Q$ is a polynomial of degree $n$ and $Q(x_i) = 0, i = 0, \ldots, n-1$, so its $n$ roots are precisely the nodes $x_0, \ldots, x_{n-1}$. Then $Q$ is of the form

$$\begin{aligned} Q(x) &= a_n(x - x_0)\ldots(x - x_{n-1}) \;=\; a_n\psi_{n-1}(x), \\ Q(x_n) &= a_n\psi_{n-1}(x_n), \end{aligned} \tag{1.9}$$

for some constant $a_n \in \mathbb{R}$ that we want to find.

On the other hand, the polynomial $L_nf$ also interpolates $f$ at the node $x_n$, so $L_nf(x_n) = f(x_n) = f_n$ and, thus,

$$Q(x_n) \;=\; f_n - L_{n-1}f(x_n). \tag{1.10}$$

By (1.9)–(1.10), it follows that

$$Q(x) \;=\; \frac{f_n - L_{n-1}f(x_n)}{\psi_{n-1}(x_n)}.$$

Now using (1.5)–(1.7), we get:

$$\begin{aligned} a_n &= \frac{f_n}{\psi_{n-1}(x_n)} - \frac{1}{\psi_{n-1}(x_n)}\sum_{i=0}^{n-1}\frac{\psi_{n-1}(x_n)}{(x_n - x_i)\psi'_{n-1}(x_i)}f_i \\ &= \frac{f_n}{\psi_{n-1}(x_n)} + \sum_{i=0}^{n-1}\frac{f_i}{(x_i - x_n)\psi'_{n-1}(x_i)} \\ &= \frac{f_n}{\psi'_n(x_n)} + \sum_{i=0}^{n-1}\frac{f_i}{\psi'_n(x_i)} \\ &= \sum_{i=0}^{n}\frac{f_i}{\psi'_n(x_i)} \;=\; f[x_0, \ldots, x_n]. \end{aligned}$$

Thus,

$$Q(x) \;=\; f[x_0, \ldots, x_n]\psi_{n-1}(x).$$

So, by (1.8)–(1.10), we have the following recurrence relation for the Lagrange polynomial:

$$L_n f(x) = L_{n-1} f(x) + f[x_0, \dots, x_n]\psi_{n-1}(x),\ n \geq 1. \tag{1.11}$$

Iteratively, we get

$$\begin{aligned}
L_0 f(x) &= f(x_0), \\
L_1 f(x) &= f(x_0) + f[x_0, x_1](x - x_0), \\
L_2 f(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1), \\
\dots & \\
L_n f(x) &= f(x_0) + f[x_0, x_1](x - x_0) + \dots + f[x_0, \dots, x_n](x - x_0) \dots (x - x_{n-1}),
\end{aligned} \tag{1.12}$$

The expression on the right-hand-side of (1.12) is called **Newton's divided difference form** of the interpolation polynomial, or **Newton's interpolation polynomial** and it is denoted by $N_n f(x)$. To be clear, by the uniqueness of the interpolation polynomial at $n + 1$ distinct nodes, the two polynomials *coincide*, $L_n f(x) = N_n f(x)$, they are just expressed (written) in different forms.

If we denote by

$$D_i = f[x_0, \dots, x_i],\ i \geq 0,$$

Newton's polynomial $N_n f$ can be written in the *nested form*

$$\begin{aligned}
N_n f(x) &= D_0 + (x - x_0)D_1 + (x - x_0)(x - x_1)D_2 + \dots + (x - x_0)\dots(x - x_{n-1})D_n \\
&= D_0 + (x - x_0)\Bigg[D_1 + (x - x_1)\Bigg[D_2 + \dots \\
&+ (x - x_{n-2})\Big[D_{n-1} + (x - x_{n-1})D_n\Big] \dots \Bigg]\Bigg].
\end{aligned} \tag{1.13}$$

Writing it this way, we can see that the evaluation of $N_n f(x)$ requires only $n$ multiplications and $n$ additions (once the divided differences have been computed), so this is a more computationally efficient formula for the interpolation polynomial.

Next, we also want to express the remainder in a new form. Let $[a, b]$ denote the smallest interval containing the distinct nodes $\{x_0, \dots, x_n\}$ and let $x \in [a, b]$ be fixed. We write recursively:

$$\begin{aligned}
f\left[x, x_0\right] &= \frac{f(x)-f(x_0)}{x-x_0} \\
f\left[x, x_0, x_1\right] &= \frac{f\left[x, x_0\right]-f\left[x_0, x_1\right]}{x-x_1} \\
f\left[x, x_0, x_1, x_2\right] &= \frac{f\left[x, x_0, x_1\right]-f\left[x_0, x_1, x_2\right]}{x-x_2} \\
\ldots & \quad \ldots \\
f\left[x, x_0, \ldots, x_{n-1}\right] &= \frac{f\left[x, x_0, \ldots, x_{n-2}\right]-f\left[x_0, x_1, \ldots, x_{n-1}\right]}{x-x_{n-1}} \\
f\left[x, x_0, x_1, \ldots, x_n\right] &= \frac{f\left[x, x_0, \ldots, x_{n-1}\right]-f\left[x_0, x_1, \ldots, x_n\right]}{x-x_n}.
\end{aligned} \tag{1.14}$$

Multiplying the first equation in (1.14) by $(x-x_0)$, the second by $(x-x_0)(x-x_1)$, the third by $(x-x_0)(x-x_1)(x-x_2)$, ..., the next to last by $(x-x_0)(x-x_1)\ldots(x-x_{n-1})$ and the last one by $(x-x_0)(x-x_1)\ldots(x-x_n)$ and adding them term by term, we obtain

$$\begin{aligned}
f(x) &= f(x_0)+f[x_0, x_1](x-x_0)+f[x_0, x_1, x_2](x-x_0)(x-x_1)+\ldots \\
&+ f[x_0, \ldots, x_n](x-x_0)\ldots(x-x_{n-1})+f[x, x_0, \ldots, x_n]\psi_n(x) \\
&= N_n f(x)+f[x, x_0, \ldots, x_n]\psi_n(x),
\end{aligned}$$

from which we have

$$R_n f(x) = f[x, x_0, \ldots, x_n](x-x_0)\ldots(x-x_n). \tag{1.15}$$

By the mean value formula for divided differences (Theorem 4.7 e), in Lecture 2), we find the previous formula for the remainder:

$$R_n f(x) = \frac{(x-x_0)\ldots(x-x_n)}{(n+1)!} f^{(n+1)}(\xi),\ \xi \in (a, b).$$

**Example 1.1.** Given the data below, find $N_1(0.15)$ and $N_2(0.15)$, the linear and quadratic interpolates evaluated at $x = 0.15$.

| $n$ | $x_n$ | $f(x_n)$ |
|---|---|---|
| 0 | 0.1 | 0.2 |
| 1 | 0.2 | 0.24 |
| 2 | 0.3 | 0.3 |

**Solution** First, we compute the divided differences:

$$\begin{tikzcd}
x_0 = 0.1 \quad f[x_0] = 0.2 \arrow[r] & f[x_0, x_1] = \dfrac{0.24 - 0.2}{0.2 - 0.1} = 0.4 \arrow[r] & f[x_0, x_1, x_2] = \dfrac{0.6 - 0.4}{0.3 - 0.1} = 1 \\
x_1 = 0.2 \quad f[x_1] = 0.24 \arrow[r] \arrow[ur] & f[x_1, x_2] = \dfrac{0.3 - 0.24}{0.3 - 0.2} = 0.6 \arrow[ur] & \\
x_2 = 0.3 \quad f[x_2] = 0.3 \arrow[ur] & &
\end{tikzcd}$$

The linear interpolate is then

$$N_1 f(x) \;=\; f(x_0) + f[x_0, x_1](x - x_0) \;=\; 0.2 + 0.4(x - 0.1) \;=\; 0.4x + 0.16,$$

so we have the approximation

$$f(0.15) \;\approx\; N_1(0.15) \;=\; 0.22.$$

Using all three nodes, we find the quadratic interpolate

$$\begin{aligned}
N_2 f(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
&= 0.2 + 0.4(x - 0.1) + 1 \cdot (x - 0.1)(x - 0.2) \\
&= x^2 + 0.1x + 0.18,
\end{aligned}$$

which yields the approximation

$$f(0.15) \;\approx\; N_2(0.15) \;=\; 0.2175.$$

∎

**Newton's forward and backward difference formula**

In the case where the interpolating nodes $x_i$ are not equally spaced, we use Newton's divided difference formula presented above; however, when the nodes are equidistant, we can construct simpler and less expensive algorithms, using finite differences. Historically, these algorithms were of great importance in interpolating functions whose values were given in tables, but the availability of more powerful computers diminished their relevance. However, with new processors (fpu's), they have made a comeback.

Assume the values of a function $f$ are known at the $h$-step equidistant nodes

$$x_i = x_0 + ih, i = 0, 1, \ldots$$

Recall the forward differences of the function $f$

$$\begin{aligned} \Delta^1 f(x_i) &= f(x_i + h) - f(x_i) = f_{i+1} - f_i, \\ \Delta^k f(x_i) &= \Delta^{k-1} f(x_i + h) - \Delta^{k-1} f(x_i) = \Delta^{k-1} f_{i+1} - \Delta^{k-1} f_i \end{aligned}$$

and the property (Proposition 4.12 in Lecture 2)

$$f[x_0, x_0 + h, \ldots, x_0 + ih] = \frac{1}{i!h^i} \Delta^i f_0.$$

The Newton form of the $n$th degree polynomial $L_n f$ of $f$ at the nodes $x_i = x_0 + ih, i = 0, 1, \ldots, n$, can be simplified. Denote by $s = (x - x_0)/h$. Then

$$\begin{aligned} (x - x_0) \ldots (x - x_{i-1}) f[x_0, \ldots, x_0 + ih] &= \big(sh\big) \cdot \big((s-1)h\big) \ldots \big((s - i + 1)h\big) \frac{1}{i!h^i} \Delta^i f_0 \\ &= \frac{s(s-1)\ldots(s-i+1)}{i!} \Delta^i f_0. \end{aligned}$$

Using the notation

$$\binom{s}{k} = \frac{s(s-1)\cdots(s-k+1)}{k!}, \; s \in \mathbb{R}, k \in \mathbb{N}$$

(the generalized binomial coefficient), we find *Newton's forward difference formula.*

$$L_n f(x) = f_0 + \binom{s}{1} \Delta f_0 + \binom{s}{2} \Delta^2 f_0 + \cdots + \binom{s}{n} \Delta^n f_0, \tag{1.16}$$

with $s = (x - x_0)/h$.

The error after $n$ iterations, for $x = x_0 + sh$, is given by

$$f(x) - L_n f(x) = h^{n+1} \binom{s}{n+1} f^{(n+1)}(\xi_x), \tag{1.17}$$

where $\xi_x$ lies in the smallest interval containing $x_0, \dots, x_n$ and $x$.

Similarly, using backward differences $\nabla$

$$\begin{aligned} \nabla^0 f_i &= f_i, \\ \nabla^1 f_i &= f_i - f_{i-1}, \\ \nabla^k f_i &= \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1} \end{aligned}$$

and the change of variables $s = (x - x_n)/h$, we obtain

$$L_n f(x) = f_n + \frac{s}{1!}\nabla f_n + \frac{s(s+1)}{2!}\nabla^2 f_n + \cdots + \frac{s(s+1)\dots(s+n-1)}{n!}\nabla^n f_n,$$

which can be written as

$$L_n f(x) = f_n + \binom{s}{1}\nabla f_n + \binom{s+1}{2}\nabla^2 f_n + \cdots + \binom{s+n-1}{n}\nabla^n f_n \tag{1.18}$$

This is called *Newton's backward difference formula*.

In this case, the interpolation error is

$$f(x) - L_n f(x) = h^{n+1} \binom{s+n}{n+1} f^{(n+1)}(\xi_x), \tag{1.19}$$

where $\xi_x$ lies in the smallest interval containing $x_0, \dots, x_n$ and $x$.

**Example 1.2.** Consider again the data in Example 1.1.

| $n$ | $x_n$ | $f_n$ |
|---|---|---|
| 0 | 0.1 | 0.2 |
| 1 | 0.2 | 0.24 |
| 2 | 0.3 | 0.3 |

Let us find $L_2 f(0.15)$ using finite differences.

**Solution** By (1.16), we have

$$\begin{aligned} L_2 f(x) &= f_0 + \begin{pmatrix} s \\ 1 \end{pmatrix} \Delta f_0 + \begin{pmatrix} s \\ 2 \end{pmatrix} \Delta^2 f_0 \\ &= f_0 + \frac{s}{1!}\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 \\ &= f_0 + \frac{x-x_0}{h}\Delta f_0 + \frac{(x-x_0)(x-x_0-h)}{2h^2}\Delta^2 f_0, \end{aligned}$$

where $s = (x - x_0)/h,\ h = 0.1$.
We compute the forward differences:

$$\begin{tikzcd}
x_0 = 0.1 & f_0 = 0.2 \arrow[r] & \Delta f_0 = 0.24 - 0.2 = 0.04 \arrow[r] & \Delta^2 f_0 = 0.06 - 0.04 = 0.02 \\
x_1 = 0.2 & f_1 = 0.24 \arrow[r] \arrow[ur] & \Delta f_1 = 0.3 - 0.24 = 0.06 \arrow[ur] & \\
x_2 = 0.3 & f_2 = 0.3 \arrow[ur] & &
\end{tikzcd}$$

So,

$$\begin{aligned} L_2 f(x) &= 0.2 + \frac{x-0.1}{0.1}\cdot 0.04 + \frac{(x-0.1)(x-0.2)}{0.02}\cdot 0.02 \\ &= x^2 + 0.1x + 0.18 \end{aligned}$$

and

$$L_2 f(0.15) \;=\; 0.2175.$$

Using backward differences, by (1.18), we get

$$\begin{aligned} L_2 f(x) &= f_2 + \begin{pmatrix} s \\ 1 \end{pmatrix} \nabla f_2 + \begin{pmatrix} s+1 \\ 2 \end{pmatrix} \nabla^2 f_2 \\ &= f_2 + \frac{s}{1!}\nabla f_2 + \frac{(s+1)s}{2!}\nabla^2 f_2 \\ &= f_2 + \frac{x-x_2}{h}\nabla f_2 + \frac{(x-x_2+h)(x-x_2)}{2h^2}\nabla^2 f_2 \end{aligned}$$

with $s = (x - x_2)/h,\ h = 0.1$.

The backward differences are found in the table

$$\begin{tikzcd}
x_0 = 0.1 & f_0 = 0.2 \arrow[dr] & & \\
x_1 = 0.2 & f_1 = 0.24 \arrow[r] \arrow[dr] & \nabla f_1 = 0.24 - 0.2 = 0.04 \arrow[dr] & \\
x_2 = 0.3 & f_2 = 0.3 \arrow[r] & \nabla f_2 = 0.3 - 0.24 = 0.06 \arrow[r] & \nabla^2 f_2 = 0.06 - 0.04 = 0.02
\end{tikzcd}$$

Hence,

$$\begin{aligned}
L_2 f(x) &= 0.3 + \frac{x - 0.3}{0.1} \cdot 0.06 + \frac{(x-0.2)(x-0.3)}{0.02} \cdot 0.02 \\
&= x^2 + 0.1x + 0.18
\end{aligned}$$

and

$$L_2 f(0.15) = 0.2175.$$

■

**Remark 1.3.** Interpolation algorithms can be classified according to the "step" of the grid (the distance between two consecutive nodes, when sorted in increasing order). There are *variable step* methods (the Lagrange form with fundamental polynomials, the barycentric formulas, Newton's divided difference formula) and *constant step* algorithms (Newton's forward and backward formulas). For variable step methods the precision is the same at any intermediate value in the interval covered by the data $(x_i, f_i)$. So these methods do not have so-called *preferential precision zones*. In contrast, Newton's forward formula is particularly useful (i.e. it has higher precision) for interpolating the values of $f(x)$ near the beginning of the set of values (closer to the first node, $(x_0, f_0)$), whereas the backward formula is preferred when the value of $f(x)$ is required near the end of the table (in the vicinity of the last node, $(x_n, f_n)$).

### 1.2.3 Aitken-type methods

These are variable step iterative methods and they highlight another important aspect: in many cases, the degree required to attain a certain desired accuracy in polynomial interpolation is *not known*. It can be obtained from the remainder, but that assumes knowledge (or at least knowing a bound) of $||f^{(n+1)}||_\infty$.

The idea behind these methods is to write an interpolation polynomial of degree $n$, iteratively, in terms of two interpolation polynomials of degree $n-1$, that only use a part of the $n+1$ nodes. Let us illustrate the idea for a simple case. For two nodes, $x_0$ and $x_1$, the polynomial of degree 1 interpolating these data, can be written successively (using Lagrange basis polynomials) as

$$\begin{aligned} P_{01}(x) &= l_0(x)f_0 + l_1(x)f_1 \\ &= \frac{x-x_1}{x_0-x_1}f_0 + \frac{x-x_0}{x_1-x_0}f_1 \\ &= \frac{(x-x_0)f_1 - (x-x_1)f_0}{x_1-x_0} \\ &= \frac{(x-x_0)P_1(x) - (x-x_1)P_0(x)}{x_1-x_0}, \end{aligned}$$

where $P_0$ denotes the polynomial that interpolates $f$ at the node $x_0$ (a polynomial of degree $0$, hence, a constant, $f_0$), $P_1$, the polynomial of degree $0$ that interpolates $f$ at the node $x_1$ (identically equal to $f_1$), and $P_{01}$ the polynomial of degree $1$ that interpolates $f$ at the nodes $x_0, x_1$. Similarly, if we add another node, $x_2$, we can define

$$P_{12}(x) = \frac{(x-x_1)P_2(x) - (x-x_2)P_1(x)}{x_2-x_1},$$

which is the polynomial of degree $1$ that interpolates $f$ at the nodes $x_1, x_2$. We proceed further and define

$$P_{012}(x) = \frac{(x-x_0)P_{12}(x) - (x-x_2)P_{01}(x)}{x_2-x_0}. \tag{1.20}$$

Let us compute its values at the nodes.

$$\begin{aligned} P_{012}(x_0) &= \frac{0 - (x_0-x_2)P_{01}(x_0)}{x_2-x_0} = P_{01}(x_0) = f_0, \\ P_{012}(x_1) &= \frac{(x_1-x_0)P_{12}(x_1) - (x_1-x_2)P_{01}(x_1)}{x_2-x_0} = \frac{(x_1-x_0)f_1 - (x_1-x_2)f_1}{x_2-x_0} = f_1, \\ P_{012}(x_2) &= \frac{(x_2-x_0)P_{12}(x_2) - 0}{x_2-x_0} = P_{12}(x_2) = f_2. \end{aligned}$$

Since $P_{012}$ is a polynomial of degree $2$, by the uniqueness of the Lagrange interpolation polynomial, it follows that $P_{012} = L_2 f$.

In a similar fashion, we can construct recursively the polynomials

$$\begin{aligned}
P_{123}(x) &= \frac{(x-x_1)P_{23}(x)-(x-x_3)P_{12}(x)}{x_3-x_1}, \\
P_{0123}(x) &= \frac{(x-x_0)P_{123}(x)-(x-x_3)P_{012}(x)}{x_3-x_0} \\
\dots
\end{aligned}$$

**Proposition 1.4.** *Let $x_0, \dots, x_k$ be distinct nodes and let $f_i, i = 0, \dots, k$, be the values of a function $f$ at the nodes. Then the Lagrange polynomial interpolating $f$ at these nodes is given by*

$$\begin{aligned}
P_{01\dots k}(x) &= \frac{1}{x_k-x_0}\begin{vmatrix} x-x_0 & P_{01\dots k-1}(x) \\ x-x_k & P_{12\dots k}(x) \end{vmatrix} \\
&= \frac{(x-x_0)P_{12\dots k}(x)-(x-x_k)P_{01\dots k-1}(x)}{x_k-x_0}. \qquad (1.21)
\end{aligned}$$

*Proof.* Obviously, by its construction, the polynomial in (1.21) has degree $k$. Its values at the nodes are

$$\begin{aligned}
P_{01\dots k}(x_0) &= \frac{-(x_0-x_k)P_{01\dots k-1}(x_0)}{x_k-x_0} = P_{01\dots k-1}(x_0) = f_0, \\
P_{01\dots k}(x_j) &= \frac{(x_j-x_0)P_{12\dots k}(x_j)-(x_j-x_k)P_{01\dots k-1}(x_j)}{x_k-x_0} = f_j,\ j = \overline{1, k-1}, \\
P_{01\dots k}(x_k) &= \frac{(x_k-x_0)P_{12\dots k}(x_k)}{x_k-x_0} = P_{12\dots k}(x_k) = f_k.
\end{aligned}$$

Hence, by the uniqueness of the Lagrange interpolation polynomial, it follows that $P_{01\dots k} = L_k f$.

□

Thus, we established a recurrence relation between a Lagrange interpolation polynomial of degree $k$ and two Lagrange interpolation polynomials of degree $k-1$. The computations can be organized in a table, illustrated below for $4$ nodes.

$$\begin{array}{lllll}
x_0 & P_0 & & & \\
x_1 & P_1 & P_{01} & & \\
x_2 & P_2 & P_{12} & P_{012} & \\
x_3 & P_3 & P_{23} & P_{123} & P_{0123}
\end{array}$$

Now, if, for instance, $P_{0123}$ does not provide a desired approximation precision, we can consider a new node and add a new line to the table:

$$x_4 \quad P_4 \quad P_{34} \quad P_{234} \quad P_{1234} \quad P_{01234}$$

and we can compare neighboring elements on a row, column or diagonal to check if the desired accuracy has been achieved.

The method described above is called *Neville's method*.

The notations can be simplified. We denote now the polynomials above by $\tilde{P}$ (instead of $P$) and define the new polynomials $P$ as follows:

$$P_{i,j} \;=\; \tilde{P}_{i-j,i-j+1,\dots,i-1,i},\; j=i,i-1,\dots 0,$$

i.e., recursively,

$$\begin{aligned} &P_{i,0} := f(x_i), \quad i=\overline{0,n},\\ &P_{i,j} := \frac{(x-x_{i-j})P_{i,j-1}-(x-x_i)P_{i-1,j-1}}{x_i-x_{i-j}}\\ &= \frac{1}{x_i-x_{i-j}}\begin{vmatrix} x-x_{i-j} & P_{i-1,j-1}\\ x-x_i & P_{i,j-1}\end{vmatrix}, \quad i\ge j>0. \end{aligned} \tag{1.22}$$

We get a new table

$$\begin{array}{lllll} x_0 & P_{00} & & & \\ x_1 & P_{10} & P_{11} & & \\ x_2 & P_{20} & P_{21} & P_{22} & \\ x_3 & P_{30} & P_{31} & P_{32} & P_{33} \end{array}$$

and the Lagrange polynomial will be the one on the diagonal $L_n f = P_{nn}$.

If the interpolation converges, then the sequence $\{P_{ii}\}_{i\ge 0}$ also converges and we can use the stopping criterion

$$|P_{ii}-P_{i-1,i-1}| \;<\; \varepsilon.$$

*Aitken's method* is similar to Neville's method. We construct the table

$$\begin{array}{lllll} x_0 & P_{00} & & & \\ x_1 & P_{10} & P_{11} & & \\ x_2 & P_{20} & P_{21} & P_{22} & \\ x_3 & P_{30} & P_{31} & P_{32} & P_{33} \end{array}$$

defining recursively

$$
\begin{aligned}
&P_{i,0} := f(x_i), \quad i = \overline{0,n}, \\
&P_{i,j+1} := \frac{1}{x_i - x_j} \begin{vmatrix} x - x_j & P_{j,j} \\ x - x_i & P_{i,j} \end{vmatrix} = \frac{(x-x_j)P_{i,j} - (x-x_i)P_{j,j}}{x_i - x_j}, \quad i > j \geq 0.
\end{aligned}
\tag{1.23}
$$

**Example 1.5.** Approximate $\sqrt{2}$ interpolating the function $f(x) = 2^x$ at the nodes $-1, 0, 1$, and then at the nodes $-1, 0, 1, 2$.

**Solution**
With Neville's method, we have the table

$$
\begin{array}{llll}
x_0 = -1 & P_{00} = 1/2 & & \\
x_1 = 0 & P_{10} = 1 & P_{11} = 5/4 & \\
x_2 = 1 & P_{20} = 2 & P_{21} = 3/2 & P_{22} = 23/16,
\end{array}
$$

where, for $x = 1/2$,

$$
\begin{aligned}
P_{11} &= \frac{(x-x_0)P_{10} - (x-x_1)P_{00}}{x_1 - x_0} = \frac{(1/2-(-1))\cdot 1 - (1/2-0)\cdot 1/2}{0-(-1)} = 5/4, \\
P_{21} &= \frac{(x-x_1)P_{20} - (x-x_2)P_{10}}{x_2 - x_1} = \frac{(1/2-0)\cdot 2 - (1/2-1)\cdot 1}{1-0} = 3/2, \\
P_{22} &= \frac{(x-x_0)P_{21} - (x-x_2)P_{11}}{x_2 - x_0} = \frac{(1/2-(-1))\cdot 3/2 - (1/2-1)\cdot 5/4}{1-(-1)} = 23/16.
\end{aligned}
$$

Thus, with linear interpolation, we get the approximation

$$\sqrt{2} \approx 23/16 = 1.4375$$

and

$$|P_{22} - P_{11}| = 3/16 = 0.1875.$$

We add a new node $x_3 = 2$ and a new line to the table, to get

$$
\begin{array}{lllll}
x_0 = -1 & P_{00} = 1/2 & & & \\
x_1 = 0 & P_{10} = 1 & P_{11} = 5/4 & & \\
x_2 = 1 & P_{20} = 2 & P_{21} = 3/2 & P_{22} = 23/16 & \\
x_3 = 2 & P_{30} = 4 & P_{31} = 1 & P_{32} = 11/8 & P_{33} = 45/32,
\end{array}
$$

with

$$\begin{aligned}
P_{31} &= \frac{(x-x_2)P_{30}-(x-x_3)P_{20}}{x_3-x_2} = \frac{(1/2-1)\cdot 4-(1/2-2)\cdot 2}{2-1} = 1,\\
P_{32} &= \frac{(x-x_1)P_{31}-(x-x_3)P_{21}}{x_3-x_1} = \frac{(1/2-0)\cdot 1-(1/2-2)\cdot 3/2}{2-0} = 11/8\\
P_{33} &= \frac{(x-x_0)P_{32}-(x-x_3)P_{22}}{x_3-x_0} = \frac{(1/2-(-1))\cdot 11/8-(1/2-2)\cdot 23/16}{2-(-1)} = 45/32.
\end{aligned}$$

The new approximation (using quadratic interpolation) is

$$\sqrt{2} \approx 45/32 = 1.4063, \text{ with } |P_{33}-P_{22}| = 1/32 = 0.0313.$$

Let us note that the exact value of $\sqrt{2}$ rounded to $4$ correct decimals is $1.4142$, so the actual errors of the two approximations are

$$|\sqrt{2}-P_{22}| = 0.0233 \text{ şi } |\sqrt{2}-P_{33}| = 0.0079.$$

With Aitken's algorithm, (1.23), we construct the table

$$\begin{array}{lllll}
x_0=-1 & P_{00}=1/2 & & & \\
x_1=0 & P_{10}=1 & P_{11}=5/4 & & \\
x_2=1 & P_{20}=2 & P_{21}=13/8 & P_{22}=23/16 & \\
x_3=2 & P_{30}=4 & P_{31}=9/4 & P_{32}=3/2 & P_{33}=45/32,
\end{array}$$

where

$$\begin{aligned}
P_{21} &= \frac{(x-x_0)P_{20}-(x-x_2)P_{00}}{x_2-x_0} = \frac{(1/2-(-1))\cdot 2-(1/2-1)\cdot 1/2}{1-0} = 13/8,\\
P_{22} &= \frac{(x-x_1)P_{21}-(x-x_2)P_{11}}{x_2-x_1} = \frac{(1/2-0)\cdot 13/8-(1/2-1)\cdot 5/4}{1-0} = 23/16\\
P_{31} &= \frac{(x-x_0)P_{30}-(x-x_3)P_{00}}{x_3-x_0} = \frac{(1/2-(-1))\cdot 4-(1/2-2)\cdot 1/2}{2-(-1)} = 9/4,\\
P_{32} &= \frac{(x-x_1)P_{31}-(x-x_3)P_{11}}{x_3-x_1} = \frac{(1/2-0)\cdot 9/4-(1/2-2)\cdot 5/4}{2-0} = 3/2,\\
P_{33} &= \frac{(x-x_2)P_{32}-(x-x_3)P_{22}}{x_3-x_2} = \frac{(1/2-1)\cdot 3/2-(1/2-2)\cdot 23/16}{2-1} = 45/32.
\end{aligned}$$

■

## 1.3 Hermite Interpolation

Consider the following situation: For a moving object, we know the distances traveled $d_0, d_1, \dots, d_m$, at times $t_0, t_1, \dots, t_m$, and we want a polynomial approximation of the distance function $d = d(t)$ on the entire interval containing the points $t_0, \dots, t_m$. Obviously, this is a Lagrange interpolation problem and we already know how to find the interpolation polynomial.

Now, assume that, in addition, we also know the values of the *velocities* $v_i$ of the object at times $t_i, i = \overline{0, m}$. We would expect that this additional information helps us find an *even better* approximation of the function $d$. However, from what we know about Lagrange interpolation, there is *no way* to include this data into our approximation. Since the velocity is the derivative with respect to time of the distance traveled, this means that we also have information about the *derivatives* of the function we want to interpolate. This is a **Hermite interpolation** problem. The ideas and computational formulas are similar to the ones we used to determine the Lagrange interpolation polynomial.

### 1.3.1 Interpolation with double nodes

For a variety of applications, as the one described above, it is convenient to consider polynomials $P(x)$ that interpolate a function $f(x)$ and in addition have the derivative polynomial $P'(x)$ also interpolate the derivative function $f'(x)$.

**Hermite interpolation problem with double nodes.** Given $m+1$ distinct nodes $x_i, i = \overline{0, m}$ and the values $f(x_i), f'(x_i)$ of an unknown function $f$ and its derivative, find a polynomial $P(x)$ of minimum degree, satisfying the interpolation conditions

$$\begin{aligned} P(x_i) &= f(x_i), \\ P'(x_i) &= f'(x_i),\ i = \overline{0, m}. \end{aligned} \tag{1.1}$$

Since for each node there are two values (of the function and of its derivative) given, we call them *double* nodes.

There are $2m+2$ conditions in (1.1), so we seek a polynomial of degree (at most) $n = 2m+1$. We determine this polynomial in a similar way to the construction of the Lagrange polynomial. Recall the notations:

$$\begin{aligned} \psi_m(x) &= (x - x_0)\dots(x - x_{m-1})(x - x_m), \\ l_i(x) &= \frac{(x - x_0)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_m)}{(x_i - x_0)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_m)} = \frac{\psi_m(x)}{(x - x_i)\psi_m'(x_i)}, \end{aligned} \tag{1.2}$$

for $i = 0, 1, \ldots, m$.

**Theorem 1.1.** *There is a unique polynomial $H_n f$ of degree at most $n$, satisfying the interpolation conditions* (1.1)*. This polynomial can be written as*

$$H_n f(x) \;=\; \sum_{i=0}^{m} \Big[h_{i0}(x) f(x_i) + h_{i1}(x) f'(x_i)\Big], \tag{1.3}$$

*where*

$$\begin{aligned} h_{i0}(x) &= \big[1 - 2l_i'(x_i)(x - x_i)\big]\big[l_i(x)\big]^2, \\ h_{i1}(x) &= (x - x_i)\big[l_i(x)\big]^2,\ i = 0, \ldots, m. \end{aligned} \tag{1.4}$$

*$H_n f$ is called the **Hermite interpolation polynomial** of $f$ at the double nodes $x_0, x_1, \ldots, x_m$. The functions $h_{i0}(x), h_{i1}(x),\ i = \overline{0,m}$ are called **Hermite fundamental polynomials** associated with these points.*

*Proof.* The degree of polynomials $l_i$ from (1.2) is $m$, so the degree of $h_{i0}, h_{i1}$ and $H_n f$ is $2m+1 = n$. The derivatives of the Hermite fundamental polynomials are

$$\begin{aligned} h_{i0}'(x) &= -2l_i'(x_i)\big(l_i(x)\big)^2 + 2\big(1 - 2l_i'(x_i)(x - x_i)\big)l_i'(x)l_i(x), \\ h_{i1}'(x) &= \big(l_i(x)\big)^2 + 2(x - x_i)l_i'(x)l_i(x). \end{aligned}$$

Notice that $l_i(x),\ i = \overline{0,m}$ are the Lagrange fundamental polynomials, thus,

$$l_i(x_j) \;=\; \delta_{ij} \;=\; \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}.$$

Then,

$$\begin{aligned} h_{i0}(x_j) &= 0,\ j \neq i, \\ h_{i0}(x_i) &= 1 \cdot \big(l_i(x_i)\big)^2 \;=\; 1, \\ h_{i1}(x_j) &= 0,\ j \neq i, \\ h_{i1}(x_i) &= 0. \end{aligned}$$

The values of the derivatives at the nodes are

$$\begin{aligned} h'_{i0}(x_j) &= 0,\ j \neq i, \\ h'_{i0}(x_i) &= -2l'_i(x_i) + 2l'_i(x_i)l_i(x_i) \ = \ 0, \\ h'_{i1}(x_j) &= 0,\ j \neq i, \\ h'_{i1}(x_i) &= 1 + 0 \ = \ 1. \end{aligned}$$

It follows that

$$\begin{aligned} (H_n f)(x_i) &= f(x_i), \\ (H_n f)'(x_i) &= f'(x_i),\ i = \overline{0, m}, \end{aligned}$$

hence, the polynomial $H_n f$ given in (1.3) satisfies the interpolation conditions (1.1).

To prove uniqueness, assume there exists another polynomial $G_n$ (of degree at most $n = 2m+1$) satisfying relations (1.1) and consider

$$Q_n \ = \ H_n - G_n.$$

From the interpolation conditions, it follows that

$$Q_n(x_i) \ = \ Q'_n(x_i) \ = \ 0,\ i = 0, \dots, m.$$

So, $Q_n$, a polynomial of degree at most $2m + 1$, has $m + 1$ *double* roots. By the Fundamental Theorem of Algebra, $Q_n$ must be identically zero, thus proving the uniqueness of $H_n$.

□

**Example 1.2.** One of the most widely used form of Hermite interpolation is the cubic Hermite polynomial, which solves the interpolation problem with two double nodes $a < b$,

$$\begin{aligned} P(a) &= f(a),\ P(b) \ = \ f(b), \\ P'(a) &= f'(a),\ P'(b) \ = \ f'(b). \end{aligned} \tag{1.5}$$

**Solution.** Letting $x_0 = a,\ x_1 = b$, with our previous notations and formulas, we have

$$\begin{aligned}
\psi_1(x) &= (x-a)(x-b), \\
l_0(x) &= \frac{x-b}{a-b},\ l_0'(x) \ = \ \frac{1}{a-b}, \\
l_1(x) &= \frac{x-a}{b-a},\ l_1'(x) \ = \ \frac{1}{b-a}.
\end{aligned}$$

The Hermite fundamental polynomials are given by

$$\begin{aligned}
h_{00}(x) &= \big(1 - 2l_0'(a)(x-a)\big)\big(l_0(x)\big)^2 \ = \ \left[1 + 2\frac{x-a}{b-a}\right]\left[\frac{b-x}{b-a}\right]^2, \\
h_{10}(x) &= \big(1 - 2l_1'(b)(x-b)\big)\big(l_1(x)\big)^2 \ = \ \left[1 + 2\frac{b-x}{b-a}\right]\left[\frac{x-a}{b-a}\right]^2, \\
h_{01}(x) &= (x-a)\big(l_0(x)\big)^2 \ = \ \frac{(x-a)(b-x)^2}{(b-a)^2}, \\
h_{11}(x) &= (x-b)\big(l_1(x)\big)^2 \ = \ -\frac{(x-a)^2(b-x)}{(b-a)^2}.
\end{aligned}$$

So the cubic Hermite polynomial is

$$\begin{aligned}
H_3f(x) &= \left[1 + 2\frac{x-a}{b-a}\right]\left[\frac{b-x}{b-a}\right]^2 \cdot f(a) \ + \ \left[1 + 2\frac{b-x}{b-a}\right]\left[\frac{x-a}{b-a}\right]^2 \cdot f(b) \\
&+ \ \frac{(x-a)(b-x)^2}{(b-a)^2} \cdot f'(a) \ - \ \frac{(x-a)^2(b-x)}{(b-a)^2} \cdot f'(b).
\end{aligned}$$

■

### 1.3.2 Newton's divided difference form

Just as in the case of Lagrange interpolation, Newton's divided differences provide a more easily computable form of the Hermite interpolation polynomial.

Consider $2m+2$ distinct nodes $z_0, z_1, \ldots, z_{2m}, z_{2m+1}$ and the Newton polynomial interpolating a function $f$ at these nodes.

$$N_{2m+1}(x) \ = \ f(z_0) + f[z_0, z_1](x - z_0) + \cdots + f[z_0, \ldots, z_{2m+1}](x - z_0)\ldots(x - z_{2m}),$$

with the error given by

$$R_{2m+1}(x) \;=\; f(x)-N_{2m+1}(x) \;=\; f[x,z_0,\dots,z_{2m+1}](x-z_0)\dots(x-z_{2m+1}).$$

We take the limits in the two relations above

$$z_0,z_1\to x_0,\quad z_2,z_3\to x_1,\ \dots,\ z_{2i},z_{2i+1}\to x_i,\ \dots\ z_{2m},z_{2m+1}\to x_m.$$

Denoting by $n=2m+1$, we get

$$\begin{aligned} N_n(x) &= f(x_0)+f[x_0,x_0](x-x_0)+f[x_0,x_0,x_1](x-x_0)^2 \\ &+ f[x_0,x_0,x_1,x_1](x-x_0)^2(x-x_1)+\dots \\ &+ f[x_0,x_0,\dots,x_m,x_m](x-x_0)^2\dots(x-x_{m-1})^2(x-x_m) \end{aligned} \tag{1.6}$$

and for the remainder,

$$f(x)-N_n(x) \;=\; f[x,x_0,x_0,\dots,x_m,x_m](x-x_0)^2\dots(x-x_m)^2. \tag{1.7}$$

**Proposition 1.3.** *Let $[a,b]\subset\mathbb{R}$ be the smallest interval containing the distinct nodes $x_0,\dots,x_m$ and $f:[a,b]\to\mathbb{R}$ be a function of class $C^{2m+2}[a,b]$. Then, for the two polynomials in* (1.3) *and* (1.6)*, we have*

$$H_nf(x) \;=\; N_n(x),\forall x\in[a,b], \tag{1.8}$$

*with the interpolation error*

$$R_n(x) \;=\; f(x)-H_nf(x) \;=\; \big[\psi_m(x)\big]^2\frac{f^{(n+1)}(\xi_x)}{(n+1)!},\ \xi_x\in(a,b). \tag{1.9}$$

*Proof.* By the way it was constructed (in (1.6)), obviously the polynomial $N_n$ has degree at most $n$. Then, by the uniqueness of the Hermite interpolation polynomial, it suffices to show that $N_n$ satisfies the interpolation conditions (1.1).
From (1.7), it follows that

$$f(x_i)-N_n(x_i) \;=\; 0,\ i=0,\dots,m.$$

Also, by the same relation, we have for the derivatives,

$$\begin{aligned} f'(x) - N_n'(x) &= (x-x_0)^2 \dots (x-x_m)^2 \frac{\partial}{\partial x} f[x, x_0, x_0, \dots, x_m, x_m] \\ &+ 2f[x, x_0, x_0, \dots, x_m, x_m] \sum_{i=0}^{m} \left[ (x-x_i) \prod_{\substack{j=0 \\ j\neq i}}^{m} (x-x_j)^2 \right], \end{aligned}$$

hence,

$$f'(x_i) - N_n'(x_i) = 0, \; i = 0, \dots, m.$$

Thus,

$$H_n f(x) = N_n(x), \forall x \in [a,b]$$

and the error formula (1.9) follows directly from (1.7) and the mean-value formula for divided differences.

$\square$

**Example 1.4.** Let us find the polynomial and the remainder for the Hermite interpolation problem with two double nodes $a < b$, from Example 1.2.

**Solution.** We have

$$\begin{aligned} H_3 f(x) &= f(a) + f[a,a](x-a) + f[a,a,b](x-a)^2 \\ &+ f[a,a,b,b](x-a)^2(x-b). \end{aligned}$$

The divided difference table for two double nodes is

$$\begin{tikzcd}
z_0 = a & f(a) \arrow[r] & f[a,a] = f'(a) \arrow[r] & f[a,a,b] \arrow[r] & f[a,a,b,b] \\
z_1 = a & f(a) \arrow[r] \arrow[ur] & f[a,b] = \dfrac{f(b)-f(a)}{b-a} \arrow[r] \arrow[ur] & f[a,b,b] \arrow[ur] & \\
z_2 = b & f(b) \arrow[r] \arrow[ur] & f[b,b] = f'(b) \arrow[ur] & & \\
z_3 = b & f(b), \arrow[ur] & & &
\end{tikzcd}$$

where

$$\begin{aligned}
f[a,a,b] &= \frac{f[a,b]-f'(a)}{b-a}, \\
f[a,b,b] &= \frac{f'(b)-f[a,b]}{b-a}, \\
f[a,a,b,b] &= \frac{f[a,b,b]-f[a,a,b]}{b-a} = \frac{f'(b)-2f[a,b]+f'(a)}{(b-a)^2}.
\end{aligned}$$

The interpolation error is given by

$$\begin{aligned}
f(x)-H_3f(x) &= (x-a)^2(x-b)^2 f[x,a,a,b,b] \\
&= \frac{(x-a)^2(x-b)^2}{24} f^{(4)}(\xi_x),
\end{aligned}$$

with $\xi_x$ belonging to the smallest interval that contains the points $a, b$ and $x$.
We can find a bound for the error. Considering that on $[a,b]$, the maximum of the function $|(x-a)(x-b)|$ occurs at the midpoint of the interval, $\frac{a+b}{2}$, and that the maximum value is $\frac{(b-a)^2}{4}$, we have

$$\max_{x\in[a,b]} \big|f(x)-H_3f(x)\big| \leq \frac{(b-a)^4}{384} \max_{t\in[a,b]} \big|f^{(4)}(t)\big|.$$

■

**Example 1.5** (Continuation of Example 1.1 in Lecture 2)**.** Consider the function $f:[0.5,5]\to\mathbb{R},\ f(x)=\sqrt{x}$ and the nodes $a=1, b=4$. Let us compare Lagrange and Hermite approximations.

**Solution.** For the *simple* nodes $a=1, b=4$, we have the interpolation conditions

$$\begin{aligned}
L_1f(a) &= f(a) = 1, \\
L_1f(b) &= f(b) = 2,
\end{aligned}$$

satified by the Lagrange polynomial of degree $1$

$$L_1f(x) = \frac{1}{3}x + \frac{2}{3}.$$

If the nodes are *double*, the interpolation conditions are

$$\begin{aligned}
H_3 f(a) &= f(a) = 1, \\
H_3 f(b) &= f(b) = 2, \\
H_3 f'(a) &= f'(a) = 1/(2\sqrt{1}) = 1/2, \\
H_3 f'(b) &= f'(b) = 1/(2\sqrt{4}) = 1/4.
\end{aligned}$$

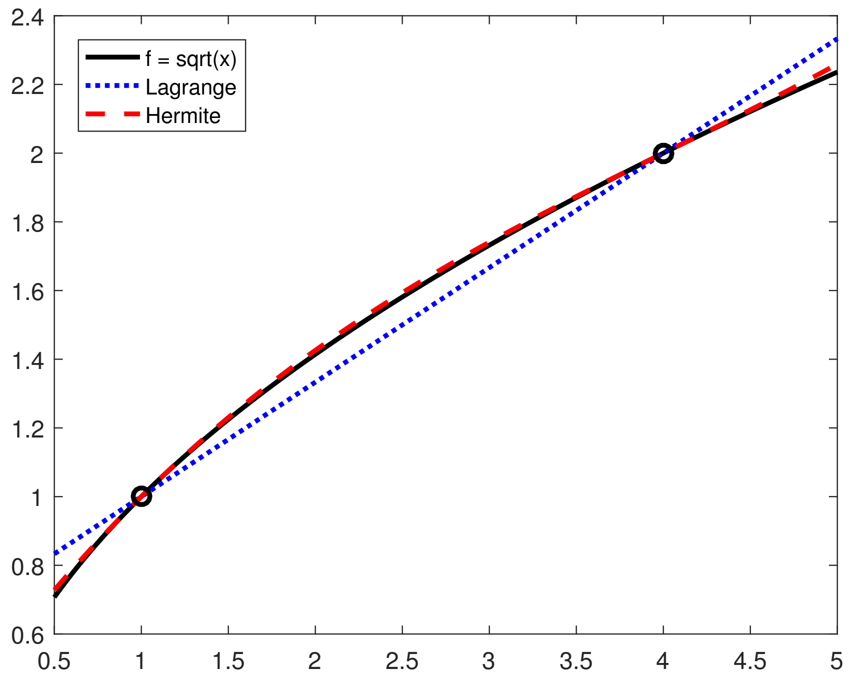

Fig. 1: Lagrange and Hermite interpolation with 2 nodes of the function $\sqrt{x}$

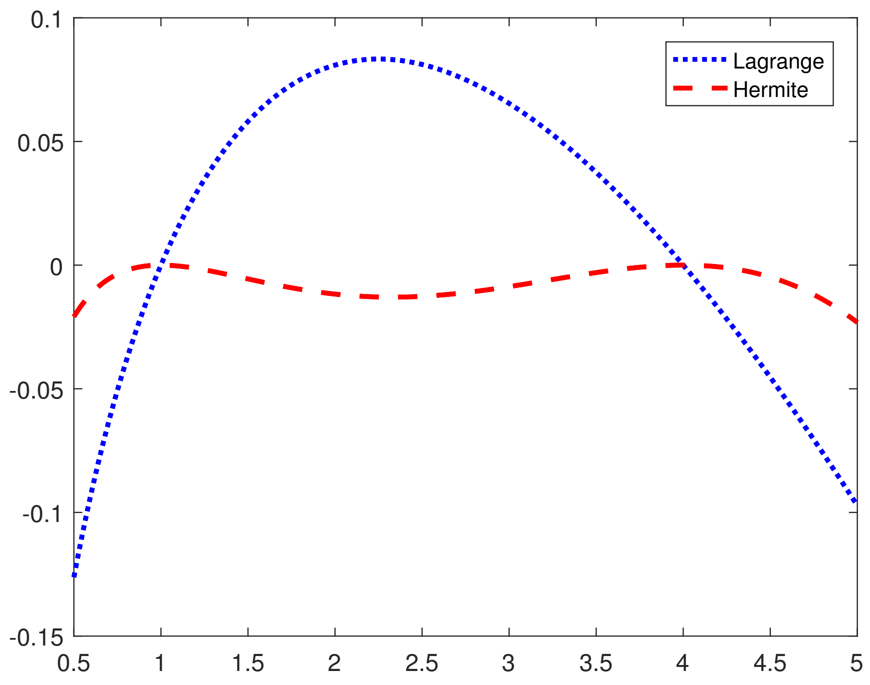

Fig. 2: Error of Lagrange and Hermite interpolation with 2 nodes of the function $\sqrt{x}$

The divided difference table is

$$\begin{tikzcd}
z_0 = 1 \quad f(1) = 1 \arrow[r] & f'(1) = 1/2 \arrow[r] & f[1,1,4] = -1/18 \arrow[r] & f[1,1,4,4] = 1/108 \\
z_1 = 1 \quad f(1) = 1 \arrow[r] \arrow[ur] & f[1,4] = 1/3 \arrow[r] \arrow[ur] & f[1,4,4] = -1/36 \arrow[ur] & \\
z_2 = 4 \quad f(4) = 2 \arrow[r] \arrow[ur] & f'(4) = 1/4 \arrow[ur] & & \\
z_3 = 4 \quad f(4) = 2, \arrow[ur] & & &
\end{tikzcd}$$

The corresponding cubic Hermite interpolation polynomial is given by

$$H_3 f(x) \;=\; 1 + \frac{1}{2}(x-1) - \frac{1}{18}(x-1)^2 + \frac{1}{108}(x-1)^2(x-4).$$

The graphs of $f$ and the two interpolation polynomials, $L_1, H_3$, on the interval $[0.5, 5]$, are shown in Figure 1. The interpolation errors are plotted in Figure 2.

■

### 1.3.3 General case

**Hermite interpolation problem.** Given $m+1$ distinct nodes $x_i \in [a,b], i = \overline{0,m}$,

$$\begin{aligned} x_0 &\text{ of multiplicity } r_0+1,\\ x_1 &\text{ of multiplicity } r_1+1,\\ &\dots\\ x_i &\text{ of multiplicity } r_i+1,\\ &\dots\\ x_m &\text{ of multiplicity } r_m+1,\end{aligned}$$

and the values $f^{(j)}(x_i),\ i = 0,1,\dots,m,\ j = 0,\dots,r_i$, of an unknown function $f : [a,b] \to \mathbb{R}$ whose derivatives of order up to $r_i$ exist at $x_i, i = \overline{0,m}$, find a polynomial $P(x)$ of minimum degree, satisfying the interpolation conditions

$$P^{(j)}(x_i) \;=\; f^{(j)}(x_i),\ i = \overline{0,m},\ j = \overline{0,r_i}. \tag{1.10}$$

Above, there are

$$n+1 \overset{\text{not}}{=} \sum_{i=0}^{m}(r_i+1)$$

conditions, so the polynomial satisfying these relations will have degree at most $n$.

**Theorem 1.6.** *There is a unique polynomial $H_n f$ of degree at most $n$, satisfying the interpolation conditions* (1.10)*. This polynomial is called the* ***Hermite interpolation polynomial*** *of the function $f$, relative to the nodes $x_0, x_1, \dots, x_m$ and the integers $r_0, r_1, \dots, r_m$, and it can be written as*

$$H_n f(x) \;=\; \sum_{i=0}^{m}\sum_{j=0}^{r_i} h_{ij}(x) f^{(j)}(x_i). \tag{1.11}$$

**Remark 1.7.**
1. The functions $h_{ij}(x),\ i = \overline{0,m}, j = \overline{0,r_i}$, are called **Hermite fundamental polynomials**.
2. If we denote by

$$\begin{aligned} u(x) &= \prod_{i=0}^{m}(x-x_i)^{r_i+1},\\ u_i(x) &= \prod_{\substack{j=0\\ j\neq i}}^{m}(x-x_j)^{r_j+1} = \frac{u(x)}{(x-x_i)^{r_i+1}},\end{aligned} \tag{1.12}$$

then the fundamental polynomials $h_{ij}(x)$ din (1.11) can be written as

$$h_{ij}(x) \quad = \quad \frac{(x-x_i)^j}{j!}\Bigg[\sum_{k=0}^{r_i-j}\frac{(x-x_i)^k}{k!}\Big[\frac{1}{u_i(x)}\Big]^{(k)}_{x=x_i}\Bigg]u_i(x). \qquad (1.13)$$

2. A more computable form can be found using Newton divided differences. Re-indexing the nodes according to their multiplicity,

$$\begin{aligned} z_0 &= x_0,\ \dots,\ z_{r_0} = x_0, \\ z_{r_0+1} &= x_1,\ \dots,\ z_{(r_0+1)+r_1} = x_1, \\ z_{(r_0+1)+(r_1+1)} &= x_2,\ \dots,\ z_{(r_0+1)+(r_1+1)+r_2} = x_2, \\ &\dots \\ z_{n-r_m} &= x_m,\ \dots,\ z_n = x_m, \end{aligned}$$

the Hermite polynomial can be written in Newton's form as

$$\begin{aligned} N_nf(x) &= f(z_0)+f[z_0,z_1](x-z_0)+\dots \\ &+ f[z_0,\dots,z_n](x-z_0)\dots(x-z_{n-1}), \end{aligned} \qquad (1.14)$$

with interpolation error

$$\begin{aligned} R_n(x) &= f(x)-N_n(x) = f[x,z_0,\dots,z_n](x-z_0)\dots(x-z_n) \\ &= \frac{u(x)}{(n+1)!}f^{(n+1)}(\xi_x),\ \xi_x\in(a,b). \end{aligned} \qquad (1.15)$$

**Special cases**

**1.** If all $r_i = 0, i = \overline{0,m}$, the nodes are simple and we have the Lagrange interpolation formula.

**2.** If we consider one single node, $x_0$, of multiplicity $n+1$, the Hermite interpolation formula is reduced to Taylor's formula.

$$\begin{aligned} f(x) &= f(x_0)+\frac{x-x_0}{1!}f'(x_0)+\frac{(x-x_0)^2}{2!}f''(x_0)+\dots \\ &+ \frac{(x-x_0)^n}{n!}f^{(n)}(x)+R_n(f), \end{aligned} \qquad (1.16)$$

with
$$R_n(x) = \frac{(x-x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\xi_x). \tag{1.17}$$

**3.** Consider the simple node $x_0$ and the double node $x_1$.

We have
$$\begin{aligned} &\text{multiplicity of the node } x_0: \quad r_0+1=1, \\ &\text{multiplicity of the node } x_1: \quad r_1+1=2, \end{aligned}$$
so $n+1=3$ and the polynomial has degree $n=2$. Thus, it is of the form
$$H_2f(x) \ = \ ax^2+bx+c.$$

Coefficients $a, b$ and $c$ are found from the interpolation conditions:
$$\begin{cases} H_2f(x_0) &= f(x_0) \\ H_2f(x_1) &= f(x_1) \\ H_2'f(x_1) &= f'(x_1) \end{cases},$$
i.e., from the linear system
$$\begin{cases} x_0^2a + x_0b + c = f(x_0) \\ x_1^2a + x_1b + c = f(x_1) \\ 2x_1a + b = f'(x_1) \end{cases}. \tag{1.18}$$

The matrix of this system,
$$V \ = \ \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 0 & 1 & 2x_1 \end{bmatrix},$$
is called a *generalized Vandermonde matrix.* It is invertible and the elements of its inverse are the coefficients of $h_{00}, h_{10}$ and $h_{11}$ from the expression in (1.11) for the Hermite polynomial
$$H_2f(x) \ = \ h_{00}f(x_0) + h_{10}f(x_1) + h_{11}f'(x_1).$$

The solution of the system (1.18) is given by

$$\begin{aligned} a &= \frac{f[x_1,x_1]-f[x_0,x_1]}{x_1-x_0} = f[x_0,x_1,x_1], \\ b &= f[x_0,x_1]-(x_1+x_0)f[x_0,x_1,x_1], \\ c &= f(x_0)-x_0f[x_0,x_1]+x_0x_1f[x_0,x_1,x_1], \end{aligned}$$

from which we get the Newton form of the Hermite polynomial

$$H_2f(x) = f(x_0)+(x-x_0)f[x_0,x_1]+(x-x_0)(x-x_1)f[x_0,x_1,x_1]. \qquad (1.19)$$

The formula for the remainder follows from (1.15):

$$R_2(x) = \frac{f^{(3)}(\xi_x)}{3!}(x-x_0)(x-x_1)^2. \qquad (1.20)$$

By symmetry, if the node $x_0$ is double and $x_1$ is simple, the corresponding Hermite polynomial and its error are given by

$$\begin{aligned} H_2(x) &= f(x_1)+(x-x_1)f[x_0,x_1]+(x-x_0)(x-x_1)f[x_0,x_0,x_1], \\ R_2(x) &= \frac{f^{(3)}(\xi_x)}{3!}(x-x_0)^2(x-x_1). \end{aligned}$$

**4.** Consider two nodes, $x_0 = a$, of multiplicity $m+1$ and $x_1 = b$, of multiplicity $n+1$.

The Hermite polynomial has degree

$$(m+1)+(n+1)-1 = m+n+1.$$

With the notations from Remark 1.7, we have

$$\begin{aligned} u(x) &= (x-a)^{m+1}(x-b)^{n+1}, \\ u_0(x) &= (x-b)^{n+1}, \\ u_1(x) &= (x-a)^{m+1}. \end{aligned}$$

the Hermite polynomial is of the form

$$H_{m+n+1}f(x) = \sum_{j=0}^{m} h_{0j}(x) f^{(j)}(a) + \sum_{i=0}^{n} h_{1i}(x) f^{(i)}(b) \tag{1.21}$$

and the fundamental polynomials are given by

$$\begin{aligned}
h_{0j}(x) &= \frac{(x-a)^j}{j!}\left[\sum_{k=0}^{m-j} \frac{(x-a)^k}{k!}\left[\frac{1}{(x-b)^{n+1}}\right]_{x=a}^{(k)}\right](x-b)^{n+1},\\
h_{1i}(x) &= \frac{(x-b)^i}{i!}\left[\sum_{k=0}^{n-i} \frac{(x-b)^k}{k!}\left[\frac{1}{(x-a)^{m+1}}\right]_{x=b}^{(k)}\right](x-a)^{m+1}.
\end{aligned}$$

In Newton's form (1.14),

$$\begin{aligned}
H_{m+n+1}f(x) &= f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(m)}(a)}{m!}(x-a)^m \\
&+ f[\underbrace{a,\ldots,a}_{m+1},b](x-a)^{m+1} + f[\underbrace{a,\ldots,a}_{m+1},b,b](x-a)^{m+1}(x-b) \\
&+ \cdots + f[\underbrace{a,\ldots,a}_{m+1},\underbrace{b,\ldots,b}_{n+1}](x-a)^{m+1}(x-b)^n,
\end{aligned}$$

with remainder

$$\begin{aligned}
R_{m+n+1} &= f[x,\underbrace{a,\ldots,a}_{m+1},\underbrace{b,\ldots,b}_{n+1}](x-a)^{m+1}(x-b)^{n+1} \\
&= \frac{f^{(m+n+2)}(\xi_x)}{(m+n+2)!}(x-a)^{m+1}(x-b)^{n+1},\ \xi_x \in (a,b).
\end{aligned}$$

## 1.4 Birkhoff Interpolation

Consider the following situation: We have a moving object and the times $t_0, t_1, \ldots, t_m$. For some of these nodes, we know the *distances* traveled $d_i = d(t_i), i \in I \subset \{0, 1, \ldots, m\}$, for others, the *velocities* $v_j = d'(t_j), j \in \tilde{I} \subset \{0, 1, \ldots, m\}$ and for others, the *accelerations* $a_k = d''(t_k), k \in I^* \subset \{0, 1, \ldots, m\}$. Having all these data, can we find a polynomial approximation of the distance function $d = d(t)$ on the entire interval containing the points $t_0, \ldots, t_m$?

Obviously, this is *not* a Lagrange interpolation problem, because we do not have the values of the function at all the nodes. We *cannot* find a Hermite polynomial, either, because at some nodes, only the value of the derivative (or the second derivative) is given (without the values of the function). This is a **Birkhoff interpolation** problem, also known as *lacunary Hermite interpolation* (because not *all* the functional or derivative values for all points are provided) and it is more general than Hermite interpolation.

### 1.4.1 Birkhoff interpolation polynomial

**Birkhoff interpolation problem.** Let $x_k \in [a, b], k = \overline{0, m}$, be $m + 1$ distinct nodes, $r_k \in \mathbb{N}$ and $I_k \subseteq \{0, , \ldots, r_k\},\ k = 0, \ldots, m$. Consider the function $f : [a, b] \to \mathbb{R}$ whose derivatives $f^{(j)}(x_k), k = 0, \ldots, m, j \in I_k$ exist. Find a polynomial $P(x)$ of minimum degree, satisfying the interpolation conditions

$$P^{(j)}(x_k) \quad = \quad f^{(j)}(x_k),\ k = \overline{0, m},\ j \in I_k. \tag{1.1}$$

Denote by $n + 1 = |I_0| + \cdots + |I_m|$, where $|I_k|$ is the cardinality (number of elements) of $I_k$. There are $n + 1$ interpolation conditions in (1.1), so we seek a polynomial of degree at most $n$,

$$P_n(x) \quad = \quad a_0 + a_1 x + \cdots + a_n x^n,$$

whose coefficients are found from the linear system generated by the interpolation conditions (1.1). If the determinant of this system is not equal to zero, then the Birkhoff interpolation problem has a unique solution.

**Remark 1.1.** If $I_k = \{0, 1, \ldots, r_k\}$, for every $k = 0, \ldots, m$, then the Birkhoff interpolation problem is reduced to Hermite interpolation (which, in turn, is reduced to Lagrange interpolation when $r_k = 0, k = 0, \ldots, m$). Hence, Birkhoff interpolation is more general.

Unlike Lagrange and Hermite interpolation, the Birkhoff interpolation problem (1.1) *does not* always have a solution. When such a polynomial, denoted by $B_n f$, exists, it has the form

$$B_n f(x) = \sum_{k=0}^{m} \sum_{j \in I_k} b_{kj}(x) f^{(j)}(x_k). \tag{1.2}$$

The terms $b_{kj}(x)$ are called **Birkhoff fundamental polynomials** and they satisfy the relations:

$$\begin{aligned} b_{kj}^{(p)}(x_\nu) &= 0,\ \nu \neq k,\ p \in I_\nu, \\ b_{kj}^{(p)}(x_k) &= \delta_{jp},\ p \in I_k,\ \text{for}\ j \in I_k \text{ and } \nu, k = 0, 1, \ldots, m, \end{aligned} \tag{1.3}$$

where

$$\delta_{jp} = \begin{cases} 0, & j \neq p \\ 1, & j = p \end{cases}$$

Kronecker's symbol.

**Remark 1.2.** Because some of the functional (or derivative) values are missing, finding mathematical expressions for the Birkhoff fundamental polynomials $b_{kj}, k = 0, \ldots, m; j \in I_k$, is, in general, difficult. They can be determined (when possible) directly from din conditions (1.1).

**Example 1.3.** Let $f \in C^2[0,1]$ and consider the nodes $x_0 = 0,\ x_1 = 1$, for which the values $f(0) = 1$ and $f'(1) = 2$ are given. Find the Birkhoff polynomial that interpolates these data.

**Solution.**
We have $m = 1$, two nodes, with $I_0 = \{0\}, I_1 = \{1\}$, so $n = 1 + 1 - 1 = 1$. We want a polynomial of degree $1$,

$$P(x) = a_0 + a_1 x,$$

satisfying the conditions

$$\begin{aligned} P(0) &= f(0), \\ P'(1) &= f'(1). \end{aligned}$$

From here, we have the linear system

$$\begin{cases} a_0 \phantom{+ a_1} &= f(0) \\ \phantom{a_0 +} a_1 &= f'(1) \end{cases}.$$

The determinant of this system is

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1 \neq 0,$$

so this problem has a unique solution

$$\begin{cases} a_0 &= f(0) \\ a_1 &= f'(1) \end{cases},$$

i.e. the polynomial we seek is

$$P(x) = f(0) + f'(1)x = 1 + 2x.$$

On the other hand, by (1.2), the Birkhoff polynomial is of the form

$$B_1 f(x) = b_{00}(x)f(0) + b_{11}(x)f'(1).$$

Let us find the fundamental polynomials $b_{00}(x)$ şi $b_{11}(x)$. Both have degree $1$, hence,

$$\begin{aligned} b_{00}(x) &= ax + b, \\ b_{11}(x) &= cx + d. \end{aligned}$$

By conditions (1.3), for $b_{00,}$ we have

$$\begin{cases} b_{00}(x_0) &= 1 \\ b'_{00}(x_1) &= 0 \end{cases} \iff \begin{cases} b_{00}(0) &= 1 \\ b'_{00}(1) &= 0 \end{cases} \iff \begin{cases} b &= 1 \\ a &= 0 \end{cases},$$

thus,

$$b_{00}(x) = 1.$$

Similarly, for $b_{11}$, we have

$$\begin{cases} b_{11}(x_0) &= 0 \\ b'_{11}(x_1) &= 1 \end{cases} \iff \begin{cases} b_{11}(0) &= 0 \\ b'_{11}(1) &= 1 \end{cases} \iff \begin{cases} d &= 0 \\ c &= 1 \end{cases},$$

so we get

$$b_{11}(x) \;=\; x.$$

Thus,

$$B_1 f(x) \;=\; f(0) + x f'(1) \;=\; 1 + 2x.$$

■

**Example 1.4.** Find a polynomial of smallest degree (if it exists) satisfying the conditions

$$P(-1) \;=\; P(1) \;=\; 0,\; P'(0) \;=\; 1.$$

**Solution.**
Here, we have $m = 2$, so 3 nodes, for which $I_0 = \{0\}, I_1 = \{1\}, I_2 = \{0\}$. Hence, we seek a polynomial of degree $n = 1 + 1 + 1 - 1 = 2$. This is of the form

$$P(x) \;=\; a_0 + a_1 x + a_2 x^2$$

and must satisfy the relations

$$\begin{aligned} P(-1) &= 0, \\ P'(0) &= 1, \\ P(1) &= 0. \end{aligned}$$

We obtain the linear system

$$\begin{cases} a_0 - a_1 + a_2 = 0 \\ \qquad a_1 \qquad\;\; = 1 \\ a_0 + a_1 + a_2 = 0 \end{cases}.$$

Subtracting the first equation from the third, we get $a_1 = 0$, which contradicts the second equation. The system is incompatible and, thus, this interpolation problem *does not have* a solution.

■

**Example 1.5. [Abel-Goncearov interpolation]** Let $f \in C^{n+1}[0, nh]$, with $h > 0,\; n \in \mathbb{N}$. Find a polynomial of smallest degree satisfying the relations

$$\begin{aligned} P(0) &= f(0), \\ P'(h) &= f'(h), \\ &\dots \\ P^{(n)}(nh) &= f^{(n)}(nh). \end{aligned}$$

This problem has a unique solution for every $h > 0,\ n \in \mathbb{N}$.

### 1.4.2 Peano's theorem and the error for Birkhoff interpolation

To find an error formula for Birkhoff interpolation (when the Birkhoff polynomial exists), we need an important result from linear operator theory.

Let us recall some notions and properties:

• Let $n \in \mathbb{N}^*$. We define the space

$$H^n[a,b] = \{f : [a,b] \to \mathbb{R} \mid f \in C^{n-1}[a,b], f^{(n-1)} \text{ absolutely continuous on } [a,b]\}.$$

• A function $f : [a,b] \to \mathbb{R}$ is *absolutely continuous* on $[a,b]$, if, for instance, it has a derivative $f'$ almost everywhere, the derivative is Lebesgue integrable, and

$$f(x) = f(a) + \int_a^x f'(t)dt,\ \forall x \in [a,b].$$

• $H^n[a,b]$ is linear space.
• Any function $f \in H^n[a,b]$ has a Taylor-type representation, with the remainder in integral form

$$f(x) = \sum_{k=0}^{n-1} \frac{(x-a)^k}{k!} f^{(k)}(a) + \int_a^x \frac{(x-t)^{n-1}}{(n-1)!} f^{(n)}(t)\, dt.$$

• The function

$$z_+ = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

is called the *positive part* of $z$, and $z_+^n$ is called a *truncated power*.

• For $m \in \mathbb{N}$, $\mathbb{P}_m$ denotes the space of all polynomials of degree at most $m$. Obviously, $\mathbb{P}_m \subset C^\infty[a,b], \forall m \geq 0$.
• The *kernel* of a linear map $L : V \to W$ between two vector spaces $V$ and $W$, is the set of all vectors in $V$ that are mapped to zero:

$$\ker L \;=\; \{v \in V \mid L(v) = \mathbf{0}_{\boldsymbol{W}}\},$$

where $\mathbf{0}_{\boldsymbol{W}}$ is null vector in $W$.

The next theorem is paramount in Numerical Analysis. It gives a representation of real linear functionals defined on $H^n[a,b]$. This result provides means for expressing the errors in many approximating procedures.

**Theorem 1.6. [Peano]**
*Let $L : H^n[a,b] \to \mathbb{R}$ be a linear functional that commutes with the definite integral operator. If* $\ker L = \mathbb{P}_{n-1}$*, then*

$$Lf \;=\; \int\limits_a^b K_n(t) f^{(n)}(t)dt, \tag{1.4}$$

*where*

$$K_n(t) \;=\; \frac{1}{(n-1)!} L\Big[(\cdot - t)_+^{n-1}\Big] \tag{1.5}$$

*is called the* ***Peano kernel****. The notation above means that the operator $L$ is applied to $f$ with respect to the variable $(\cdot)$.*

*Proof.* As $f \in H^n[a,b]$, we know that it has a Taylor-type representation, with the remainder in integral form

$$f(x) \;=\; (T_{n-1}f)(x) + (R_{n-1}f)(x), \tag{1.6}$$

where

$$(T_{n-1}f)(x) \;=\; \sum_{k=0}^{n-1} \frac{(x-a)^k}{k!} f^{(k)}(a) \tag{1.7}$$

and

$$(R_{n-1}f)(x) \;=\; \int_a^x \frac{(x-t)^{n-1}}{(n-1)!} f^{(n)}(t)\,dt \;=\; \frac{1}{(n-1)!}\int_a^x (x-t)^{n-1} f^{(n)}(t)\,dt. \tag{1.8}$$

So,

$$(Lf)(x) \;=\; \big(L(T_{n-1}f)\big)(x) + \big(L(R_{n-1}f)\big)(x). \tag{1.9}$$

Since $\ker L = \mathbb{P}_{n-1}$ and $(T_{n-1}f) \in \mathbb{P}_{n-1}$, we have

$$\big(L(T_{n-1}f)\big)(x) \;=\; 0, \forall x \in [a,b]. \tag{1.10}$$

Now, the positive part of $(x-t)$, for $x, t \in [a,b]$ is

$$(x-t)_+ \;=\; \begin{cases} x-t, & t \le x \\ 0, & t > x \end{cases}.$$

Thus, we can write

$$\begin{aligned}\int_a^b (x-t)_+^{n-1} f^{(n)}(t)\,dt \;&=\; \int_a^x (x-t)_+^{n-1} f^{(n)}(t)\,dt + \int_x^b (x-t)_+^{n-1} f^{(n)}(t)\,dt \\ &=\; \int_a^x (x-t)^{n-1} f^{(n)}(t)\,dt.\end{aligned}$$

Substituting this into (1.8), we get

$$(R_{n-1}f)(x) \;=\; \frac{1}{(n-1)!}\int_a^b (x-t)_+^{n-1} f^{(n)}(t)\,dt. \tag{1.11}$$

Since $L$ commutes with the definite integral operator, we obtain

$$\begin{aligned}\big(L(R_{n-1}f)\big)(x) &= \frac{1}{(n-1)!}L\Big(\int_a^b (x-t)_+^{n-1} f^{(n)}(t)\,dt\Big)\\ &= \frac{1}{(n-1)!}\int_a^b L\Big((x-t)_+^{n-1}\Big) f^{(n)}(t)\,dt \qquad (1.12)\\ &= \int_a^b \big(K_n(t)\big)(x) f^{(n)}(t)\,dt\end{aligned}$$

By relations (1.9), (1.10) and (1.12), it follows that

$$Lf = \int_a^b K_n(t) f^{(n)}(t)dt.$$

$\square$

**Corollary 1.7.** If the kernel $K$ has constant sign on $[a,b]$ and $f^{(n)}$ is continuous on $[a,b]$, then there exists $\xi \in [a,b]$ such that

$$Lf = \frac{1}{n!} f^{(n)}(\xi) Le_n, \tag{1.13}$$

where $e_k(x) = x^k,\ k \in \mathbb{N}$.

*Proof.* If the kernel $K$ has constant sign on $[a,b]$, we can apply the mean value theorem in (1.4):

$$Lf = f^{(n)}(\xi)\int_a^b K_n(t)\,dt. \tag{1.14}$$

Notice that the kernel $K$ *does not* depend on $f$ and the relation above is true *regardless* of the function $f$. Then, taking $f = e_n$, we get

$$\begin{aligned} Le_n &= e_n^{(n)}(\xi) \int_a^b K_n(t)\,dt \\ &= n! \int_a^b K_n(t)\,dt, \end{aligned}$$

from which we get

$$\int_a^b K_n(t)\,dt = \frac{1}{n!} Le_n.$$

Using this in (1.14), we obtain (1.13).

$\square$

This corollary is the one that is mostly used in applications (to assess the approximation error).

**Example 1.8.** Let us find a formula for the rest of the Birkhoff polynomial in Example 1.3.

**Solution.** We found the Birkhoff polynomial

$$B_1 f(x) = f(0) + f'(1)x,\ x \in [0,1],$$

so, we have

$$f(x) = B_1 f(x) + R_1 f(x).$$

We apply Peano's theorem to the rest operator,

$$Lf = R_1 f = f - B_1 f.$$

We have

$$\begin{aligned} R_1 e_0(x) &= e_0(x) - B_1 e_0(x) = e_0(x) - \big(e_0(0) + e_0'(1)x\big) = 1 - (1+0) = 0, \\ R_1 e_1(x) &= e_1(x) - B_1 e_1(x) = e_1(x) - \big(e_1(0) + e_1'(1)x\big) = x - (0 + 1 \cdot x) = 0, \\ R_1 e_2(x) &= e_2(x) - B_1 e_2(x) = e_2(x) - \big(e_2(0) + e_2'(1)x\big) = x^2 - 2x \neq 0, \end{aligned}$$

(the first two were obvious, since $B_1$ is polynomial of degree $1$, but it is a good computational exercise). Thus,

$$R_1 f(x) \;=\; \int_0^1 K_2(x,t) f''(t)\,dt,$$

with

$$\begin{aligned} K_2(x,t) &= R_1\left(\frac{(x-t)_+^1}{1!}\right) \\ &= \underbrace{(x-t)_+}_{f(x)} - \Big(\underbrace{(0-t)_+}_{f(0)} + \underbrace{1\cdot x}_{f'(1)x}\Big) \\ &= (x-t)_+ - (-t)_+ - x. \end{aligned}$$

Since $x, t \in [0,1]$, we have $(-t)_+ = 0$. Fix an arbitrary $x \in [0,1]$.
If $0 \le t \le x$, then $(x-t)_+ \;=\; x-t$ and

$$K_2(x,t) \;=\; x-t-x \;=\; -t \;\le\; 0.$$

If $x \le t \le 1$, then $(x-t)_+ \;=\; 0$ and

$$K_2(x,t) \;=\; 0-x \;=\; -x \;\le\; 0.$$

So, either way, $K_2$ has constant sign on $[0,1]$. By Corollary 1.7, it follows that

$$\begin{aligned} R_1 f(x) &= \frac{1}{2!} f''(\xi) R_1 e_2(x) \\ &= \frac{x^2-2x}{2!} f''(\xi),\; \xi \in [0,1]. \end{aligned}$$

As $|x(x-2)| \le 1$, for $x \in [0,1]$, we have the following estimate for the interpolation error:

$$|R_1 f(x)| \;\le\; \frac{1}{2}||f''||_\infty,\; \forall x \in [0,1].$$

■

**Example 1.9.** Let $f \in C^3[0,2]$ be a function for which the values $f'(0) = 1$, $f(1) = 2$ and $f'(2) = 1$ are known. Approximate $f(1/2)$ using Birkhoff interpolation and estimate the error.

**Solution.**

We have $m+1=3$ nodes, with $I_0=\{1\}, I_1=\{0\}, I_2=\{1\}$, so $n=1+1+1-1=2$. The Birkhoff polynomial of degree (at most) 2,

$$\begin{aligned} P(x) &= ax^2+bx+c, \\ P'(x) &= 2ax+b, \end{aligned}$$

must satisfy the relations

$$\begin{aligned} P'(0) &= b = f'(0), \\ P(1) &= a+b+c = f(1) \\ P'(2) &= 4a+b = f'(2). \end{aligned}$$

The determinant of the corresponding linear system is

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 4 & 1 & 0 \end{vmatrix} = -\begin{vmatrix} 1 & 1 \\ 4 & 0 \end{vmatrix} = 4 \neq 0,$$

hence, there exists a unique Birkhoff interpolation polynomial, of the form

$$B_2f(x) = b_{01}(x)f'(0)+b_{10}(x)f(1)+b_{21}(x)f'(2).$$

The fundamental polynomials (of degree at most 2) must satisfy the conditions

$$\left\{\begin{aligned} b'_{01}(0) &= 1 \\ b_{01}(1) &= 0 \\ b'_{01}(2) &= 0 \end{aligned}\right., \quad \left\{\begin{aligned} b'_{10}(0) &= 0 \\ b_{10}(1) &= 1 \\ b'_{10}(2) &= 0 \end{aligned}\right. \quad \text{and} \quad \left\{\begin{aligned} b'_{21}(0) &= 0 \\ b_{21}(1) &= 0 \\ b'_{21}(2) &= 1 \end{aligned}\right..$$

From these, we find the interpolation polynomial

$$\begin{aligned} B_2f(x) &= \frac{1}{4}(x-1)(3-x)f'(0)+f(1)+\frac{1}{4}(x^2-1)f'(2) \\ \Bigg( &= \frac{1}{4}(x-1)(3-x)+2+\frac{1}{4}(x^2-1) = x+1\Bigg), \end{aligned}$$

with derivative

$$\begin{aligned}(B_2f)'(x) &= \frac{1}{2}(2-x)f'(0) + \frac{1}{2}xf'(2)\\ \Big( &= \frac{1}{2}(2-x) + \frac{1}{2}x \;=\; 1\Big).\end{aligned}$$

The remainder is computed as

$$R_2f(x) = f(x) - B_2f(x) \;=\; f(x) - \left[\frac{1}{4}(x-1)(3-x)f'(0) + f(1) + \frac{1}{4}(x^2-1)f'(2)\right].$$

We have

$$\begin{aligned}R_2e_2(x) &= x^2 - \left[0 + 1 + \frac{1}{4}(x^2-1)\cdot 4\right] \;=\; x^2 - 1 - (x^2-1) \;=\; 0,\\ R_2e_3(x) &= x^3 - \left[0 + 1 + \frac{1}{4}(x^2-1)\cdot 12\right] \;=\; (x-1)(x^2-2x-2) \;\neq\; 0.\end{aligned}$$

(again, the first relation *needed not* be checked, but it is important to see that it *does hold*, even though, for this particular data, the polynomial $B_2f$ has actually degree *less than* 2).
The remainder is given by

$$R_2f(x) \;=\; \int_0^2 K_3(x,t)f'''(t)\,dt.$$

where

$$\begin{aligned}K_3(x,t) &= R_2\left(\frac{(x-t)_+^2}{2!}\right)\\ &= \frac{(x-t)_+^2}{2} - \left[\frac{1}{4}(x-1)(3-x)(0-t)_+ + \frac{(1-t)_+^2}{2} + \frac{1}{4}(x^2-1)(2-t)_+\right].\end{aligned}$$

We compute

$$\begin{aligned}K_3(1/2,t) &= \frac{(1/2-t)_+^2}{2}\\ &- \left[\frac{1}{4}(1/2-1)(3-1/2)(0-t)_+ + \frac{(1-t)_+^2}{2} + \frac{1}{4}((1/2)^2-1)(2-t)_+\right]\\ &= \frac{1}{2}\left(\frac{1}{2}-t\right)_+^2 - \frac{1}{2}(1-t)_+^2 \;+ \frac{3}{16}(2-t).\end{aligned}$$

We have the following cases:

1. $0 \le t \le \dfrac{1}{2}$,

$$K_3(1/2,t) \;=\; \frac{1}{8}(1-2t)^2 - \frac{1}{2}(1-t)^2 + \frac{3}{16}(2-t) \;=\; \frac{5}{16}t \;\ge\; 0.$$

2. $\dfrac{1}{2} \le t \le 1$,

$$K_3(1/2,t) \;=\; -\frac{1}{2}(1-t)^2 + \frac{3}{16}(2-t) \;=\; -\frac{1}{16}(8t^2 - 13t + 2) \;\ge\; 0,$$

   because the roots of the quadratic polynomial above, $\dfrac{13 \pm \sqrt{105}}{16} \;=\; \{0.1721, 1.4529\}$, lie *outside* the interval $\left[\dfrac{1}{2}, 1\right]$.

3. $1 \le t \le 2$,

$$K_3(1/2,t) \;=\; \frac{3}{16}(2-t) \;\ge\; 0.$$

So, $K_2$ has constant on $[0,2]$ and, thus,

$$R_2 f(1/2) \;=\; \frac{1}{3!}f'''(\xi)R_2 e_3(1/2) \;=\; \frac{1}{6}\cdot\frac{11}{8}f'''(\xi) \;=\; \frac{11}{48}f'''(\xi),\; \xi \in [0,2].$$

In the end, we have the approximation

$$f(1/2) \;\approx\; B_2 f(1/2) \;=\; 3/2,$$

with the error

$$|R_2 f(1/2)| \;\le\; \frac{11}{48}||f'''||_\infty.$$

■

# 2 Spline Interpolation

Polynomial interpolation has a major setback: the difference between the values of the function $f$ and the values of the interpolation polynomial outside the nodes' interval can be quite large. Choosing more nodes and finding a higher degree polynomial does not solve this problem, but increases the computational cost. So, even though polynomials are smooth and easy to work with functions, they are not always the best choice for approximating functions.

From these considerations came the idea of changing polynomials to *piecewise polynomials* that satisfy some continuity conditions (of the interpolation function and some of its derivatives). Such functions are called *splines*.

Historically, spline functions can be traced all the way back to ancient mathematics. The term “spline” was first used by I. J. Schoenberg in $1946$, but a thorough spline function theory started developing in $1964$, as their good approximating properties became more evident. They can be used in a large variety of ways in approximation theory, computer graphics, data fitting, numerical integration and differentiation, and the numerical solution of integral, differential, and partial differential equations.

Over time, there have been several world renowned research groups in spline theory, scattered all over the world. One such group, with remarkable contributions, was a Romanian research group (based especially in Cluj).

The basic idea of approximating a function on an interval $[a, b]$ with spline functions, is to use different polynomials (of lower degree) on different parts of the interval. The reason for this is the fact that on a sufficiently small interval, functions can be approximated arbitrarily well by polynomials of low degree, even degree $1$, or $0$.

**Definition 2.1.** *Let $\Delta$ be a grid of the interval $[a, b]$,*

$$\Delta \ : \ a = x_1 < x_2 < \cdots < x_{n-1} < x_n = b.$$

*The set*

$$\mathbb{S}_m^k(\Delta) \ = \ \{s \mid s \in C^k[a,b],\ s|_{[x_i,x_i+1]} \in \mathbb{P}_m,\ i = 1,2,\ldots,n-1\} \tag{2.1}$$

*is called the* ***the space of polynomial spline functions*** *of degree $m$ and class $k$ on $\Delta$.*

These are piecewise polynomial functions, of degree $\leq m$, continuous at $x_1, \ldots, x_{n-1}$, together with all their derivatives of order up to $k$. In general, we assume $0 \le k < m$. For $k = m$,

$$S_m^m \;=\; \mathbb{P}_m.$$

If $k = -1$, we allow discontinuities at the grid points.

## 2.1 Linear Splines

For $m = 1$ and $k = 0$, we have *linear spline functions*. We determine a function $s_1 \in \mathbb{S}_1^0(\Delta)$ such that

$$s_1(x_i) \;=\; f(x_i) \;=\; f_i,\; i = 1, 2, \dots, n.$$

That means that on the interval $[x_i, x_{i+1}]$, the function $s_1$ is the interpolation polynomial of degree $1$

$$s_1(f;x) \;=\; f_i + f[x_i, x_{i+1}](x - x_i) \;=\; f_i + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i). \tag{2.2}$$

The graph of this function is shown in Figure 1.
The error is given by

$$|f(x) - s_1(f;x)| \;\le\; \frac{|(x - x_i)(x - x_{i+1})|}{2!} \max_{x \in [x_i, x_{i+1}]} |f''(x)| \le \frac{h_i^2}{8} \max_{x \in [x_i, x_{i+1}]} |f''(x)|, \tag{2.3}$$

where we denoted by $h_i = x_{i+1} - x_i$.
Hence, if $|\Delta|$ denotes

$$|\Delta| \;=\; \max_{i = \overline{1, n-1}} h_i,$$

we have

$$||f(\cdot) - s_1(f, \cdot)||_\infty \;\le\; \frac{|\Delta|^2}{8} ||f''||_\infty. \tag{2.4}$$

Obviously, $\mathbb{S}_1^0(\Delta)$ is a vector space. To find its dimension, we count the number of degrees of freedom and the number of constraints. There are $n - 1$ subintervals and $2$ coefficients to be determined (i.e. $2$ degrees of freedom) on each, for a total of $2(n-1)$. We have continuity conditions at each interior node, so $n - 2$ constraints. Thus, in the end we have

$$\dim \mathbb{S}_1^0(\Delta) \;=\; 2(n-1) - (n-2) \;=\; n.$$

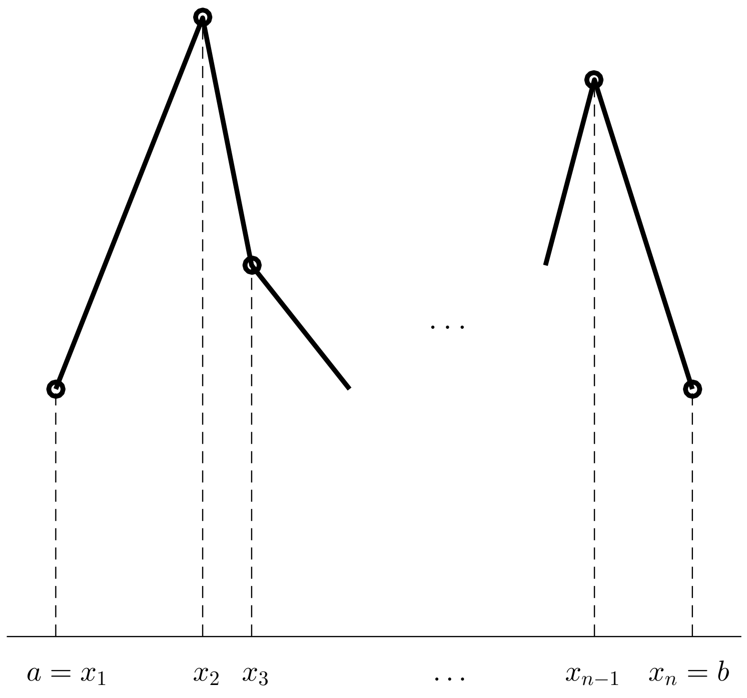

Fig. 1: Linear splines

A basis for this space is given by the so-called *B-spline functions*. Taking $x_0 = x_1 = a,\ x_{n+1} = x_n = b$, for $i = \overline{1,n}$, we define

$$B_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & \text{If } x_{i-1} \le x \le x_i \\ \dfrac{x_{i+1} - x}{x_{i+1} - x_i}, & \text{If } x_i \le x \le x_{i+1} \\ 0, & \text{in rest.} \end{cases} \tag{2.5}$$

Note that the first equation for $i = 1$, and the second equation for $i = n$, are to be ignored. The functions $B_i$ are sometimes referred to as "hat functions" (Chinese hats), but note that the first and the last hat are cut in half. Their graphs are depicted in Figure 2. They are linearly independent and have the property

$$B_i(x_j) = \delta_{ij}.$$

Any function $s \in \mathbb{S}_1^0(\Delta)$ can be written uniquely as

$$s(x) = \sum_{i=1}^{n} c_i B_i(x).$$

$B$-spline functions play the same role as fundamental Lagrange polynomials $l_i$.

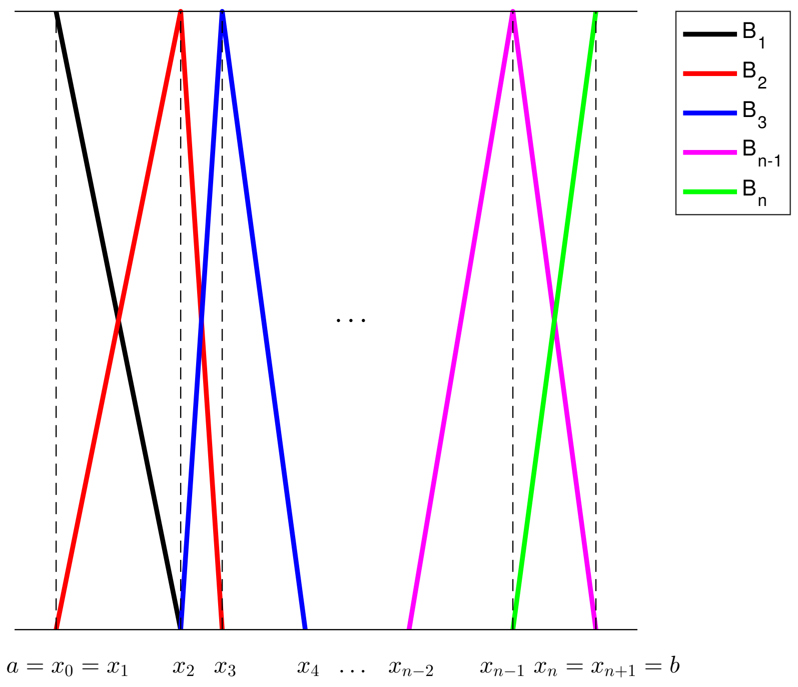

Fig. 2: Linear B-splines

A linear spline agrees with the data, but it has the disadvantage of not having a smooth graph. Most data will represent a smooth curved graph, one without the corners of a linear spline. Consequently, we usually want to construct a smooth curve that interpolates the given data points, but one that follows the shape of the linear spline.

## 2.2 Cubic Splines

*Cubic splines* are the most widely used. In general, cubic splines are fairly smooth functions that are convenient to work with, and they have come to be widely used in the past several decades in computer graphics and in many areas of Applied Mathematics and Statistics. There are several types of cubic spline functions, depending on the smoothness conditions they satisfy.

**Interpolation with cubic splines** $s \in \mathbb{S}_3^1(\Delta)$

We impose the continuity of the first order derivative of $s_3(f;\cdot)$ by prescribing the values of the first derivative at each node $x_i, i = 1, 2, \dots, n$. Given $n$ arbitrary numbers $m_1, m_2, \dots, m_n$, we seek a function $s_3(f;\cdot)$ that satisfies the conditions

$$\begin{aligned} s_3|_{[x_i,x_i+1]} &= p_i(x) \in \mathbb{P}_3,\ i = 1, 2, \dots, n-1, \\ s_3(f;x_i) &= f_i,\ i = 1, 2, \dots, n, \\ s_3'(f;x_i) &= m_i,\ i = 1, 2, \dots, n. \end{aligned} \tag{2.6}$$

This means that on each subinterval $[x_i, x_{i+1}]$, $s_3(f;\cdot)$ is the unique solution of the Hermite interpolation problem

$$\begin{aligned} p_i(x_i) &= f_i,\ \ p_i(x_{i+1}) \ = \ f_{i+1}, \\ p_i'(x_i) &= m_i,\ \ p_i'(x_{i+1}) \ = \ m_{i+1},\ i = \overline{1, n-1}. \end{aligned} \tag{2.7}$$

The divided differences are computed from the table

$$\begin{tikzcd}
x_i & f_i \arrow[r] & m_i \arrow[r] & \frac{f[x_i,x_{i+1}]-m_i}{h_i} \arrow[r] & \frac{m_{i+1}-2f[x_i,x_{i+1}]+m_i}{h_i^2} \\
x_i & f_i \arrow[r] \arrow[ur] & f[x_i,x_{i+1}] \arrow[r] \arrow[ur] & \frac{m_{i+1}-f[x_i,x_{i+1}]}{h_i} \arrow[ur] & \\
x_{i+1} & f_{i+1} \arrow[r] \arrow[ur] & m_{i+1} \arrow[ur] & & \\
x_{i+1} & f_{i+1}, \arrow[ur] & & &
\end{tikzcd}$$

Using the Newton form of the Hermite polynomial, we have

$$\begin{aligned} p_i(x) &= f_i + m_i(x - x_i) + \frac{f[x_i, x_{i+1}] - m_i}{h_i}(x - x_i)^2 \\ &+ \frac{m_{i+1} - 2f[x_i, x_{i+1}] + m_i}{h_i^2}(x - x_i)^2(x - x_{i+1}). \end{aligned}$$

Alternatively, we can write it in Taylor's form around $x_i$. Considering that $x - x_{i+1} = x - x_i - h_i$, for $x \in [x_i, x_{i+1}]$, we get

$$p_i(x) = c_{i,0} + c_{i,1}(x - x_i) + c_{i,2}(x - x_i)^2 + c_{i,3}(x - x_i)^3, \tag{2.8}$$

with

$$\begin{aligned} c_{i,0} &= f_i, \\ c_{i,1} &= m_i, \\ c_{i,2} &= \frac{f[x_i,x_{i+1}]-m_i}{h_i} - c_{i,3}h_i \;=\; \frac{3f[x_i,x_{i+1}]-2m_i-m_{i+1}}{h_i}, \\ c_{i,3} &= \frac{m_{i+1}-2f[x_i,x_{i+1}]+m_i}{h_i^2}. \end{aligned} \tag{2.9}$$

Hence, to compute $s_3(f;x)$ at a point $x \in [a,b]$ that is not a node, we first identify the interval $[x_i, x_{i+1}]$ that contains $x$, then compute the coefficients in (2.9) and evaluate the spline using (2.8).

Next, we discuss some possible choices for the parameters $m_1, m_2, \ldots, m_n$.

**Piecewise cubic Hermite interpolation**

Assuming that the derivatives $f'(x_i),\ i = 1, \ldots, n$, are known, we choose $m_i = f'(x_i)$. This way, we obtain a strictly local scheme, where the polynomial on each subinterval $[x_i, x_{i+1}]$ is completely determined by the interpolation data at node points inside, independently of the other pieces. The error in this case (see Example 1.4, in Lecture 4) is

$$||f(\cdot) - s_3(f,\cdot)||_\infty \;\le\; \frac{1}{384}\,|\Delta|^4\,||f^{(4)}||_\infty. \tag{2.10}$$

For equally spaced nodes, we have

$$|\Delta| \;=\; (b-a)/(n-1)$$

and, therefore,

$$||f(\cdot) - s_3(f,\cdot)||_\infty \;=\; O(n^{-4}),\ n \to \infty. \tag{2.11}$$

**Interpolation with cubic splines $s \in \mathbb{S}_3^2(\Delta)$**

To have $s_3(f;\cdot) \in S_3^2(\Delta)$, we require continuity of the second derivatives at the nodes, i.e.

$$p''_{i-1}(x_i) \;=\; p''_i(x_i),\ i = 2, \ldots, n-1,$$

which, for the Taylor coefficients in (2.8), means

$$2c_{i-1,2} \;+\; 6c_{i-1,3}h_{i-1} \;=\; 2c_{i,2},\ i = 2, \ldots, n-1. \tag{2.12}$$

Substituting in (2.9), we obtain the linear system

$$h_i m_{i-1} \;+\; 2(h_{i-1}+h_i)m_i \;+\; h_{i-1}m_{i+1} \;=\; b_i,\ i=2,\ldots,n-1, \tag{2.13}$$

where

$$b_i \;=\; 3\Big(h_i f[x_{i-1},x_i] + h_{i-1} f[x_i,x_{i+1}]\Big). \tag{2.14}$$

Thus, we have a system of $n-2$ linear equations with $n$ unknowns, $m_1, m_2, \ldots, m_n$. Once $m_1$ and $m_n$ are chosen, the system is tridiagonal and can be solved efficiently by several methods.

Next, we discuss possible choices for $m_1$ and $m_n$.

1. **Complete (clamped) splines.** We take

$$m_1 = f'(a),\ m_n = f'(b).$$

For this type of spline, it can be shown that, if $f \in C^4[a,b]$, then

$$||f^{(r)}(\cdot) - s_3^{(r)}(f,\cdot)||_\infty \;\le\; C_r|\Delta|^{4-r}\ ||f^{(4)}||_\infty,\ r=0,1,2,3, \tag{2.15}$$

where

$$C_0 = \frac{5}{384},\ C_1 = \frac{1}{24},\ C_2 = \frac{3}{8},$$

and $C_3$ depends on the ratio $|\Delta|/\min\limits_i h_i$.

2. **Endpoint second derivative splines.** We require

$$s_3''(f,a) \;=\; f''(a),\ s_3''(f,b) \;=\; f''(b).$$

These lead to two more equations,

$$\begin{aligned} 2m_1 \;+\; m_2 \qquad\qquad &= 3f[x_1,x_2] - \frac{1}{2}f''(a)h_1, \\ m_{n-1} \;+\; 2m_n &= 3f[x_{n-1},x_n] - \frac{1}{2}f''(b)h_{n-1}. \end{aligned} \tag{2.16}$$

We place the first equation at the beginning of the system (2.13) and the second at the end of it, thus preserving the tridiagonal structure of the system.

3. **Natural cubic splines.** Imposing

$$s_3''(f;a) = s_3''(f;b) = 0,$$

we get the same two equations as above, with $f''(a) = f''(b) = 0$:

$$\begin{array}{ccccccc} 2m_1 & + & m_2 & & & = & 3f[x_1,x_2], \\ & & & m_{n-1} & + \; 2m_n & = & 3f[x_{n-1},x_n]. \end{array} \tag{2.17}$$

The advantage of this type of spline is that it requires only the function values of $f$ – no derivatives – but the price paid is a decrease in the accuracy to $O(|\Delta|^2)$ near the endpoints (unless indeed $f''(a) = f''(b) = 0$).

4. **"Not-a-knot" (deBoor) splines.** Here we impose the conditions that the first two pieces and the last two, coincide, i.e.

$$p_1(x) \equiv p_2(x), \; p_{n-2}(x) \equiv p_{n-1}(x).$$

This means that the first interior node, $x_2$, and the last one, $x_{n-1}$, are both inactive (hence, the name). We get two more equations expressing the continuity of $s_3'''(f;x)$ at $x = x_2$ and $x = x_{n-1}$. This comes down to the equality of the leading coefficients $c_{1,3} = c_{2,3}$ and $c_{n-2,3} = c_{n-1,3}$. Thus, we get

$$\begin{array}{lclclcl} h_2^2\, m_1 & + & \big(h_2^2 - h_1^2\big) m_2 & - & h_1^2\, m_3 & = & \beta_1, \\ h_{n-1}^2\, m_{n-2} & + & \big(h_{n-1}^2 - h_{n-2}^2\big) m_{n-1} & - & h_{n-2}^2\, m_n & = & \beta_2, \end{array} \tag{2.18}$$

where

$$\begin{aligned} \beta_1 &= 2\Big(h_2^2 f[x_1,x_2] - h_1^2 f[x_2,x_3]\Big), \\ \beta_2 &= 2\Big(h_{n-1}^2 f[x_{n-2},x_{n-1}] - h_{n-2}^2 f[x_{n-1},x_n]\Big). \end{aligned}$$

Again, we place the first equation at the beginning of the system (2.13) and the second at the end of it. Even so, the resulting system is *no longer* tridiagonal, but it can be transformed into a tridiagonal one, by combining equations 1 and 2, and $n-1$ and $n$, respectively. Consequently, the first and the last equations become

$$\begin{array}{lclclcl} h_2\, m_1 & + & \big(h_2+h_1\big)m_2 & & & = & \gamma_1, \\ & & & \big(h_{n-1}-h_{n-2}\big)m_{n-1} & + \quad h_{n-2}\, m_n & = & \gamma_2, \end{array} \tag{2.19}$$

where

$$\begin{aligned} \gamma_1 &= \frac{1}{h_2+h_1}\left[f[x_1,x_2]h_2\Big(h_1+2\big(h_1+h_2\big)\Big)+h_1^2 f[x_2,x_3]\right], \\ \gamma_2 &= \frac{1}{h_{n-1}+h_{n-2}}\left[h_{n-1}^2 f[x_{n-2},x_{n-1}]+\Big(2\big(h_{n-1}+h_{n-2}\big)+h_{n-1}\Big)h_{n-2} f[x_{n-1},x_n]\right]. \end{aligned}$$

**Finding cubic splines using the second derivatives**

Computational formulas for finding cubic splines $s \in \mathbb{S}_3^2(\Delta)$ can be derived (in a similar way) when the arbitrary numbers $M_1, M_2, \ldots, M_n$ are given and forced to satisfy the conditions

$$\begin{aligned} s_3\big|_{[x_i,x_i+1]} &= p_i(x) \in \mathbb{P}_3,\ i=1,2,\ldots,n-1, \\ s_3(f;x_i) &= f_i,\ i=1,2,\ldots,n, \\ s_3''(f;x_i) &= M_i,\ i=1,2,\ldots,n. \end{aligned} \tag{2.20}$$

Since $s_3$ is a cubic polynomial, its second derivative is linear. Hence, on $[x_i, x_{i+1}]$, we have

$$s_3''(f;x) = ax+b,$$

satisfying the conditions

$$s_3''(f;x_i) = M_i,\ s_3''(f;x_{i+1}) = M_{i+1},\ \ i=1,2,\ldots,n-1.$$

The values $a$ and $b$ are determined from the system

$$\left\{ \begin{array}{lclcl} ax_i & + & b & = & M_i \\ ax_{i+1} & + & b & = & M_{i+1} \end{array} \right. .$$

Integrating successively, then imposing (2.20) and the continuity conditions at the nodes, $s_3'(f;x_i) = s_3'(f;x_{i+1}),\ i=\overline{1,n-1}$, we get the linear system

$$h_{i-1}\, M_{i-1} \ + \ 2\,(h_{i-1}+h_i)\, M_i \ + \ h_i\, M_{i+1} \ = \ 6\,(f[x_i,x_{i+1}]-f[x_{i-1},x_i])\,, \tag{2.21}$$

for $i = \overline{2, n-1}$.
The two extra conditions needed for a closed system can be imposed, e.g., on $M_1$ and $M_n$. If $M_1 = M_n = 0$, we get the natural cubic spline.
Other conditions can be enforced, such as the continuity of $s_3'''(f; x)$ at $x = x_2$ and $x = x_{n-1}$, which lead to deBoor cubic splines.
If the first and last equations are

$$\begin{array}{ccccc} 2M_1 & + & M_2 & & & = & 6\left(f[x_1, x_2] - f_1'\right), \\ & & & M_{n-1} & + \; 2M_n & = & 6\left(f_n' - f[x_{n-1}, x_n]\right), \end{array} \tag{2.22}$$

where $f_1' = f'(a), f_n' = f'(b)$, then the resulting function is the complete cubic spline.

**Example 2.2.** Find the natural cubic spline that interpolates the data

| $x_i$ | 1 | 2 | 4 | 5 |
|---|---|---|---|---|
| $f_i$ | 3 | 5 | 9 | 10 |

**Solution.**
We have $n = 4$ nodes and $h_1 = 1, h_2 = 2, h_3 = 1$.
From (2.13)–(2.17), the linear system for the unknowns $m_i$, also called *slopes*, is

$$\begin{cases} 2m_1 + m_2 = 6 \\ 2m_1 + 6m_2 + m_3 = 18 \\ 2m_2 + 6m_3 + 2m_4 = 12 \\ m_3 + 2m_4 = 3 \end{cases}$$

with solution

$$m_1 = \frac{87}{46},\ m_2 = \frac{51}{23},\ m_3 = \frac{21}{23},\ m_4 = \frac{24}{23}.$$

The system (from (2.21) together with the conditions $M_1 = M_4 = 0$) for the *moments* $M_i$ becomes

$$\begin{cases} M_1 = 0 \\ M_1 + 6M_2 + 2M_3 = 0 \\ 2M_2 + 6M_3 + M_4 = -6 \\ M_4 = 0 \end{cases}$$

whose solution is

$$M_1 = 0,\ M_2 = \frac{3}{8},\ M_3 = -\frac{9}{8},\ M_4 = 0.$$

Hence, both ways, we get the natural cubic spline function

$$s_3(x) = \begin{cases} \dfrac{x^3}{16} - \dfrac{3x^2}{16} + \dfrac{17x}{8} + 1, & x \in [1,2] \\[2ex] -\dfrac{x^3}{8} + \dfrac{15x^2}{16} - \dfrac{x}{8} + \dfrac{5}{2}, & x \in [2,4] \\[2ex] -\dfrac{3x^3}{16} - \dfrac{45x^2}{16} + \dfrac{119x}{8} - \dfrac{35}{2}, & x \in [4,5] \end{cases} .$$

■

**Minimality properties of cubic spline interpolants**

Natural and complete splines have interesting optimality properties. Henceforth, we denote them by $s_{nat}(f;\cdot)$ and $s_{compl}(f;\cdot)$, respectively.

**Theorem 2.3.** *Let $g \in C^2[a,b]$ be any function that interpolates $f$ on $\Delta$. Then*

$$\int_a^b |s''_{nat}(f;x)|^2 dx \ \le \ \int_a^b |g''(x)|^2 dx, \tag{2.23}$$

*with equality if and only if $g(\cdot) \ = \ s_{nat}(f;\cdot)$.*

For the next minimality result, we slightly change the subdivision $\Delta$. Consider the grid

$$\Delta' \ : \ a = x_0 = x_1 < x_2 < \cdots < x_{n-1} < x_n = x_{n+1} = b, \tag{2.24}$$

where the endpoints are double nodes. That means that when we use $\Delta'$, we interpolate the function values at all interior points, and, both the functional and the derivative values, at the endpoints.

**Theorem 2.4.** *Let $g \in C^2[a,b]$ be any function that interpolates $f$ on $\Delta'$. Then*

$$\int_a^b |s''_{compl}(f;x)|^2 dx \ \le \ \int_a^b |g''(x)|^2 dx, \tag{2.25}$$

*with equality if and only if $g(\cdot) \ = \ s_{compl}(f;\cdot)$.*

**Remark 2.5.** Taking $g(\cdot) = s_{compl}(f;\cdot)$ in Theorem 2.3, we get

$$\int_a^b |s''_{nat}(f;x)|^2 dx \quad \leq \quad \int_a^b |s''_{compl}(f;x)|^2 dx. \tag{2.26}$$

So, in a sense, the natural cubic spline is the "smoothest" interpolant.

**Example 2.6.** Consider the function $f(x) = \arctan x,\ x \in [-2, 2]$ and the nodes $\{-2, -1, 0, 1, 2\}$. Figure 3 shows the graphs of the function $f$, the nodes and the complete, natural, deBoor and piecewise Hermite cubic splines interpolating $f$. In Figure 4 we have the interpolation errors.

**Remark 2.7.** These minimality properties are at the origin of the name "spline". A *spline* is a flexible strip of wood used in drawing curves (or a musical instrument in that shape).

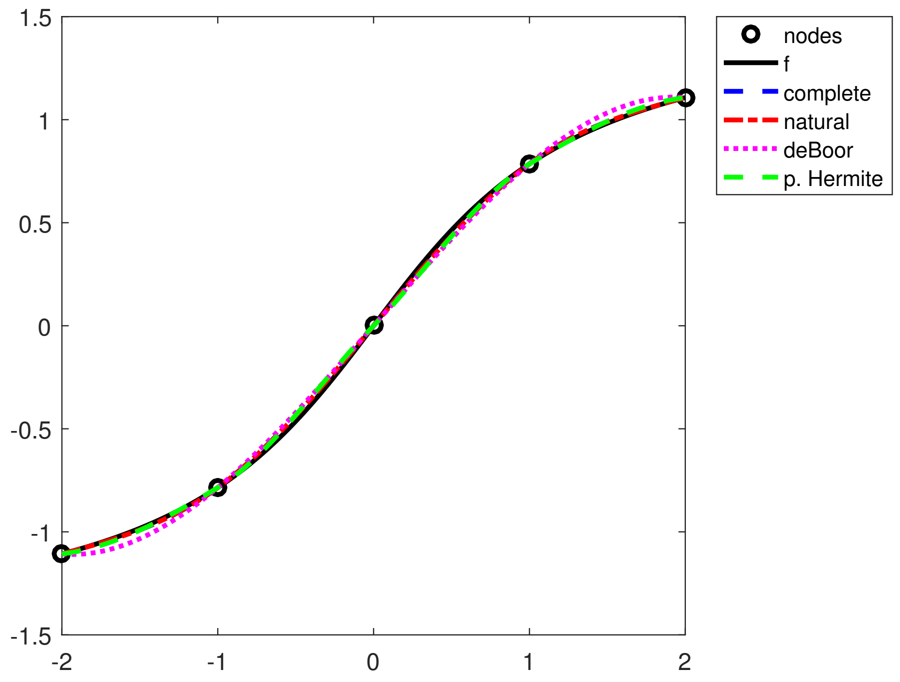

Fig. 3: Interpolation with cubic splines, $f(x) = \arctan x$

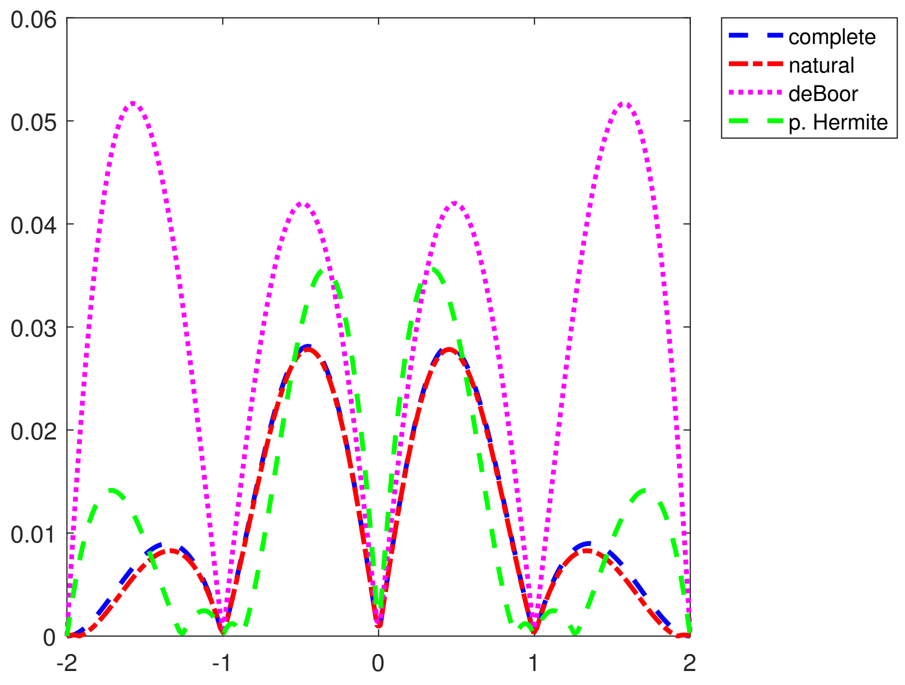

Fig. 4: Errors in cubic spline interpolation, $f(x) = \arctan x$

# 3 Least Squares Approximation

## 3.1 Best approximation problem

In general, an approximation problem can be described as follows: Let $f \in X$ be a function, $\Phi$, a family of approximants and $||\cdot||$ a norm on $X$. We seek an approximation $\hat{\varphi} \in \Phi$ of $f$ that approximates the given function "as well as possible".

$$||f - \hat{\varphi}|| \;\; \leq \;\; ||f - \varphi||, \; \forall \varphi \in \Phi. \tag{3.1}$$

This is called a *best approximation problem* of $f$ with elements of $\Phi$. The function $\hat{\varphi}$ is called a *best approximation* of $f$ relative to the norm $||\cdot||$. Given a basis $\{\pi_j\}_1^m$ of $\Phi$, we can write

$$\Phi \;=\; \Phi_m \;\;=\;\; \Big\{\varphi \mid \varphi(t) = \sum_{j=1}^{m} c_j \pi_j(t),\; c_j \in \mathbb{R}\Big\}. \tag{3.2}$$

$\Phi$ is a finite dimensional linear space or a subset of one. We have already seen examples of such spaces, $\Phi = \mathbb{P}_n$ or $\Phi = S_m^k(\Delta)$, when we discussed interpolation problems. The norms can be

$$\begin{aligned}
||u||_\infty &= \max_{t\in[a,b]} |u(t)|, \\
||u||_2 &= \Big(\int_a^b |u(t)|^2 \, dt\Big)^{1/2}, \\
||u||_{2,w} &= \Big(\int_a^b w(t) \, |u(t)|^2 \, dt\Big)^{1/2},
\end{aligned}$$

or their discrete versions,

$$\begin{aligned}
||u||_\infty &= \max_{i=\overline{1,m}} |u(t_i)|, \\
||u||_2 &= \Big(\sum_{i=1}^m |u(t_i)|^2\Big)^{1/2}, \\
||u||_{2,w} &= \Big(\sum_{i=1}^m w_i \, |u(t_i)|^2\Big)^{1/2},
\end{aligned}$$

where $w$ is a *weight* function. An intuitive, physical justification for a *weighted norm* would be that some observations are more important than others, or they are more common, so they "weigh more".

We will consider a particular case for problem (3.1) by choosing the (discrete or continuous) $||\cdot||_2$ norm. This is then called a *least squares approximation problem* or *square mean approximation problem*. Its solution was given by Gauss and Legendre at the beginning of the 19th century.

In what follows, we will only consider the *discrete* least squares approximation problem, or the problem of *curve (data) fitting*, as this is most common in various applications.

## 3.2 Normal Equations

To simplify the writing, let us recall the *inner (scalar) product* on $\mathbb{R}^m$:

$$(u, v) = \sum_{j=1}^m u(t_i)v(t_i).$$

Let $x$ and $y$ be two variables. For distinct values $x_1, x_2, \ldots, x_n$, we have (measure) the corresponding values of $y$, denoted by $y_1, y_2, \dots, y_n$. We seek a relation of the form

$$y \;=\; f(x) \;=\; c_1\pi_1(x) + \dots c_m\pi_m(x), \tag{3.3}$$

so that the *residues* (errors)

$$r_i \;=\; y_i - f(x_i),\; i = 1, \dots, n, \tag{3.4}$$

have a minimum norm. In other words, we want to find the values $c_j$ that minimize the error

$$E \;=\; \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} \big(y_i - \sum_{j=1}^{m} c_j\pi_j(x_i)\big)^2. \tag{3.5}$$

We solve this problem (finding the minimum of a function) by taking partial derivatives (with respect to each unknown) and setting them equal to 0. We get

$$\frac{\partial E}{\partial c_i} \;=\; -2\sum_{i=1}^{n} \pi_i(x_i)\big(y_i - \sum_{j=1}^{m} c_j\pi_j(x)\big) \;=\; 0,\; i = 1, \dots, n,$$

or

$$\sum_{j=1}^{m} (\pi_i, \pi_j)c_j \;=\; (\pi_i, y),\; i = 1, \dots, n. \tag{3.6}$$

The equations in (3.6) are called *normal equations*. The determinant of the system is the Gram determinant of the vectors $\{\pi_1, \dots, \pi_m\}$, which is *not* 0, as these vectors are linearly independent.

Let us see some examples.

**1.** If we seek a linear function $f$, i.e.

$$f(x) = ax + b, \tag{3.7}$$

then we want the minimum of the function

$$E(a, b) = \sum_{i=1}^{n} (ax_i + b - y_i)^2$$

This is a function of two variables, whose minimum we want to find. The normal equations are:

$$\begin{aligned} E'_a &= 2\sum_{i=1}^{n}(ax_i+b-y_i)x_i, & \sum_{i=1}^{n}(ax_i+b-y_i)x_i &= 0, \\ E'_b &= 2\sum_{i=1}^{n}(ax_i+b-y_i), & \sum_{i=1}^{n}(ax_i+b-y_i) &= 0, \end{aligned}$$

so, we have

$$\begin{aligned} a\sum_{i=i}^{n} x_i^2 \;+\; b\sum_{i=i}^{n} x_i &= \sum_{i=i}^{n} x_i y_i \\ a\sum_{i=i}^{n} x_i \;+\; nb &= \sum_{i=i}^{n} y_i. \end{aligned} \tag{3.8}$$

This $2 \times 2$ linear system has a unique solution, because its determinant is

$$n\sum_{i=i}^{n} x_i^2 - \Big(\sum_{i=i}^{n} x_i\Big)^2 \;>\; 0$$

(a consequence of Cauchy's inequality).

**2.** If we want a quadratic function $f$, of the form

$$f(x) \;=\; ax^2+bx+c, \tag{3.9}$$

then we seek the minimum of the error function

$$E(a,b,c) \;=\; \sum_{i=1}^{n}(ax_i^2+bx_i+c-y_i)^2.$$

The parameters $a, b$ and $c$ are determined from the normal system

$$\begin{aligned} a\sum_{i=1}^{n} x_i^4 + b\sum_{i=1}^{n} x_i^3 + c\sum_{i=1}^{n} x_i^2 &= \sum_{i=1}^{n} x_i^2 y_i, \\ a\sum_{i=1}^{n} x_i^3 + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} x_i y_i, \\ a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i + \; nc &= \sum_{i=1}^{n} y_i, \end{aligned} \tag{3.10}$$

obtained by setting the partial derivatives of $E$ (with respect to $a, b$ and $c$) equal to zero.

**Example 3.1.** Find the least squares polynomial approximation that fits the following data:

| $x_i$ | 0.5 | 1.5 | 2 | 3 | 3.5 | 4.5 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| $y_i$ | 5 | 5.8 | 5.8 | 6.8 | 6.9 | 7.6 | 7.8 | 8.2 | 9.2 | 9.9 |

**Solution.** The scatterplot is shown in Figure 5. We see from the graph that the best fit is given by a linear function,

$$f\,(x; a, b) \;=\; ax + b.$$

The solution of the normal equations (3.8) is

$$a = 0.63, \quad b = 4.71.$$

So, the least squares polynomial best fit is given by

$$f(x) \;=\; 0.63x + 4.71.$$

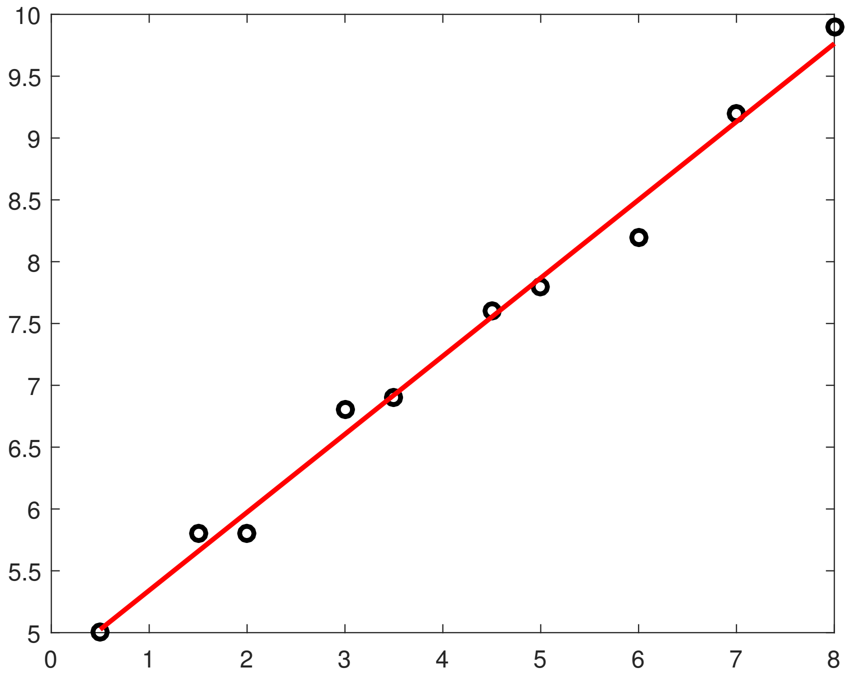

Fig. 5: Example 3.1

■

# Chapter 3. Numerical Differentiation and Integration

## 1 Approximation of Linear Functionals, Basic Notions

Let $X$ be a linear space and $L, L_1, \dots, L_m : X \to \mathbb{R}$ be real, linear functionals, that are linearly independent.

**Definition 1.1.** *An approximation formula of L using $L_1, \dots, L_m$, is a formula of the type*

$$L(f) = \sum_{i=1}^{m} A_i L_i(f) + R(f), \ f \in X. \tag{1.1}$$

*The real parameters $A_i$ are called* ***coefficients****, and $R(f)$ is the* ***remainder*** *of the formula.*

For an approximation formula of the form (1.1), given $L_i$, we want to determine the coefficients $A_i$ and study the corresponding remainder (error).
The functionals $L_i$ express the available information on $f$ and they also depend on the particular type of approximation we seek, i.e. on $L$.

**Example 1.2.** Let $X = \{f \mid f : [a,b] \to \mathbb{R}\}, L_i(f) = f(x_i)$, for some distinct nodes $x_i \in [a,b], i = \overline{0,m}$ and $L(f) = f(\alpha)$, for an arbitrary $\alpha \in [a,b]$. Formula (1.1) becomes

$$f(\alpha) = \sum_{i=0}^{m} l_i(\alpha) f(x_i) + (Rf)(\alpha),$$

i.e. the *Lagrange interpolation formula*. We have

$$A_i = l_i(\alpha),$$

where $l_i$ are the Lagrange fundamental polynomials. One of the expressions for the remainder is

$$(Rf)(\alpha) = \frac{u(\alpha)}{(m+1)!} f^{(m+1)}(\xi), \ \xi \in [a,b], \ u(x) = (x - x_0) \dots (x - x_m),$$

if $f^{(m+1)}$ exists on $[a,b]$.

**Example 1.3.** Let $X$ and $L_i$ be defined as in the previous example. Assuming that $f^{(k)}(\alpha), \ k \in \mathbb{N}^*$ exists, define $L(f) = f^{(k)}(\alpha)$. We get an approximation formula for the derivative of order $k$ of $f$

at $\alpha$,

$$f^{(k)}(\alpha) \;=\; \sum_{i=0}^{m} A_i f(x_i) + R(f),$$

called a *numerical differentiation formula*.

**Example 1.4.** Let $x_k \in [a,b], k = \overline{0,m}$ be distinct nodes and $I_k$ some sets of indices. Consider $X = \{f \mid f : [a,b] \to \mathbb{R}, f$ integrable on $[a,b]$, for which $f^{(j)}(x_k), k = \overline{0,m}, j \in I_k$ exist$\}$, $L_{kj}(f) = f^{(j)}(x_k)$ and $L(f) = \int\limits_a^b f(x)dx$. Formula (1.1) becomes

$$\int\limits_a^b f(x)dx \;=\; \sum_{k=0}^{m} \sum_{j\in I_k} A_{kj} f^{(j)}(x_k) + R(f),$$

called a *numerical integration (quadrature) formula*.

**Definition 1.5.** *If $\mathbb{P}_d \subset X$, the number $d \in \mathbb{N}$ satisfying the property $\ker R = \mathbb{P}_d$ is called **degree of precision (exactness)** of the approximation formula* (1.1).

**Remark 1.6.** Since $R$ is a linear functional, the following are equivalent:

$$\ker R = \mathbb{P}_d \iff \begin{cases} R(e_k) = 0,\ k = 0,1,\ldots,d, \\ R(e_{d+1}) \neq 0, \end{cases} \tag{1.2}$$

with $e_k(x) = x^k$.

In general, there are two approaches for solving the approximation problem (1.1):
$-$ the **interpolation method**: apply the functional $L$ to a suitable interpolation polynomial of $f$, instead of $f$ itself;
$-$ the **method of undetermined coefficients**: find the coefficients in (1.1), by using the relations from (1.2).

# 2 Numerical Differentiation

Numerical approximation of derivatives is used when the values of a function $f$ are given in tables, as empirical data, or the expression of $f$ is complicated.

We can derive simple, immediate numerical differentiation rules using divided and finite differences. Let $f : [a, b] \to \mathbb{R}$ be differentiable on $[a, b]$, $x \in [a, b]$, arbitrary and $h > 0$. We have

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} = \lim_{h\to 0} f[x, x+h].$$

From here, we immediately get the approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \equiv D_h f(x), \tag{2.1}$$

called the *forward difference numerical derivative*.
Expanding $f(x+h)$ in a Taylor's series around $x$, we get

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2} f''(\xi), \\ \frac{f(x+h) - f(x)}{h} &= f'(x) + \frac{h}{2} f''(\xi),\ \xi \in (x, x+h), \end{aligned}$$

from which we have the error formula

$$(RD_h f)(x) = f'(x) - D_h f(x) = -\frac{h}{2} f''(\xi),\ \xi \in (x, x+h). \tag{2.2}$$

The error is proportional to $h$, so formula (2.2) can be used for small steps $h$.

Similarly, we obtain the *backward difference numerical derivative*,

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \equiv \widetilde{D}_h f(x), \tag{2.3}$$

with approximation error

$$(R\widetilde{D}_h f)(x) = f'(x) - \frac{f(x) - f(x-h)}{h} = \frac{h}{2} f''(\xi),\ \xi \in (x-h, x). \tag{2.4}$$

**Example 2.1.** Use $D_h f(x)$ (formula (2.1)) to approximate the derivative of $f(x) = \cos x$ at $x = \pi/6$. Study the error of the approximation.

**Solution.** The exact value is

$$f'\left(\frac{\pi}{6}\right) = -\sin\frac{\pi}{6} = -\frac{1}{2}.$$

By (2.2), the error is

$$(RD_h f)\Big(\frac{\pi}{6}\Big) \;=\; f'\Big(\frac{\pi}{6}\Big) - D_h\Big(\frac{\pi}{6}\Big) \;=\; \frac{h}{2}\cos\xi,$$

thus,

$$\Big|(RD_h f)\Big(\frac{\pi}{6}\Big)\Big| \;\leq\; \frac{h}{2}.$$

Table 1 contains the approximation results for various values of $h$. Indeed, the value of $D_h f$ is approaching $-0.5$. Moreover, looking at the errors, we see that when $h$ is halved, the error is almost halved (see the ratio column). This confirms the fact that the error is proportional to $h$ (relation (2.2)). ■

| $h$ | $D_h f$ | Error | Ratio |
|---|---|---|---|
| $0.1$ | $-0.54243$ | $0.04243$ | |
| $0.05$ | $-0.52144$ | $0.02144$ | $1.98$ |
| $0.025$ | $-0.51077$ | $0.01077$ | $1.99$ |
| $0.0125$ | $-0.50540$ | $0.00540$ | $1.99$ |
| $0.00625$ | $-0.50270$ | $0.00270$ | $2.00$ |
| $0.003125$ | $-0.50135$ | $0.00135$ | $2.00$ |

Table 1: Example 2.1, $f(x) = \cos x$

## 2.1 Interpolation Method

Consider Newton's interpolation formula

$$f(x) \;=\; N_n f(x) + R_n f(x), \tag{2.5}$$

with

$$\begin{aligned} N_n f(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &+ f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) + \ldots \\ &+ f[x_0, \ldots, x_n](x - x_0) \ldots (x - x_{n-1}), \\ R_n f(x) &= f[x, x_0, \ldots, x_n](x - x_0) \ldots (x - x_n). \end{aligned}$$

Taking the first derivative in (2.5), we get

$$\begin{aligned}
f'(x) &= f[x_0,x_1] + f[x_0,x_1,x_2](x-x_1+x-x_0) + \dots \\
&+ f[x_0,\dots,x_n]\Big((x-x_0)\dots(x-x_{n-1})\Big)' \\
&+ (R_n f)'(x).
\end{aligned} \tag{2.6}$$

Assuming that $f$ has $n+1$ continuous derivatives on a suitable interval, for the remainder, we have

$$\begin{aligned}
(R_n f)'(x) &= \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\Big((x-x_0)\dots(x-x_n)\Big)' \\
&= \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\sum_{i=0}^{n}(x-x_0)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n),
\end{aligned} \tag{2.7}$$

where $\xi_x$ lies in the smallest interval containing $x_0,\dots,x_n$ and $x$.

Similarly, differentiating twice, we get

$$\begin{aligned}
f''(x) &= 2f[x_0,x_1,x_2] + f[x_0,x_1,x_2,x_3]\Big((x-x_0)(x-x_1)(x-x_2)\Big)'' + \dots \\
&+ f[x_0,\dots,x_n]f[x_0,\dots,x_n]\Big((x-x_0)\dots(x-x_{n-1})\Big)'' \\
&+ (R_n f)''(x).
\end{aligned} \tag{2.8}$$

Let us consider a few cases:

- For $n=1$ and the nodes $x_0 = x, x_1 = x+h$, from (2.6) we get the numerical differentiation formula

$$f'(x) \approx f[x,x+h] = \frac{f(x+h)-f(x)}{h},$$

  and for the remainder, by (2.7),

$$(R_1 f)'(x) = \frac{f''(\xi_x)}{2!}(2x-x_0-x_1) = -\frac{h}{2}f''(\xi_x),\ x \le \xi_x \le x+h,$$

  so, we find again the forward difference numerical derivative.

- In a similar manner, for $n=1$ and the nodes $x_0 = x-h, x_1 = x$, we obtain again the backward difference numerical derivative.

- An interesting case is $n = 2$, with the nodes $x_0 = x - h, x_1 = x, x_2 = x + h$. By (2.6),

$$\begin{aligned} f'(x) &\approx f[x_0, x_1] + f[x_0, x_1, x_2](x - x_1 + x - x_0) \\ &= \frac{f(x+h) + f(x-h)}{2h} \equiv \widehat{D}_h f(x), \end{aligned} \tag{2.9}$$

  known as the *central difference numerical derivative formula*. In this case, for the remainder, we get from (2.7),

$$\begin{aligned} (R_2 f)'(x) &= \frac{f'''(\xi_x)}{3!}(x - x_0)(x - x_2) \\ &= -\frac{h^2}{6} f'''(\xi_x), \; x - h \le \xi_x \le x + h. \end{aligned} \tag{2.10}$$

  This says that for small values of $h$, the formula (2.9) should be more accurate than the earlier approximations, because the error term of (2.10) decreases more rapidly with $h$.

**Example 2.2.** Consider again $f(x) = \cos x$. Let us approximate $f'(\pi/6)$ using (2.9).

**Solution.** The results of the approximation are given in Table 2, and they confirm the rate of convergence of $O(h^2)$ given in (2.10).

| $h$ | $\widehat{D}_h f$ | Error | Ratio |
|---|---|---|---|
| $0.1$ | $-0.49916708$ | $-8.329e-4$ | |
| $0.05$ | $-0.49979169$ | $-2.083e-4$ | $4.00$ |
| $0.025$ | $-0.49994792$ | $-5.208e-5$ | $4.00$ |
| $0.0125$ | $-0.49998698$ | $-1.302e-5$ | $4.00$ |
| $0.00625$ | $-0.49999674$ | $-3.255e-6$ | $4.00$ |

Table 2: Example 2.2, $f(x) = \cos x$

■

This procedure can be carried on, to approximate higher order derivatives.

## 2.2 Method of Undetermined Coefficients

In general, the method of undetermined coefficients is used when we seek an approximation method of a specific form, using specific values. We illustrate it by finding an approximation formula for the second derivative, of the form

$$f''(x) \approx D_h^{(2)} f(x) \equiv Af(x+h) + Bf(x) + Cf(x-h),\ h > 0. \tag{2.11}$$

We determine the unknown coefficients $A$, $B$ and $C$ the following way: replace $f(x+h)$ and $f(x-h)$ by their Taylor polynomials,

$$\begin{aligned} f(x+h) &\approx f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{6} f'''(x) + \frac{h^4}{24} f^{(4)}(x), \\ f(x-h) &\approx f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{6} f'''(x) + \frac{h^4}{24} f^{(4)}(x). \end{aligned}$$

Including more terms would give higher powers of $h$ and for small values of $h$, these additional terms should be much smaller than the terms already included above (so, negligible). Substituting these approximations into the formula for $D_h^{(2)} f(x)$ and collecting together common powers of $h$ give us

$$\begin{aligned} D_h^{(2)} f(x) &\approx (A+B+C)f(x) + h(A-C)f'(x) + \frac{h^2}{2}(A+C)f''(x) \\ &+ \frac{h^3}{6}(A-C)f'''(x) + \frac{h^4}{24}(A+C)f^{(4)}(x). \end{aligned} \tag{2.12}$$

To have

$$D_h^{(2)} f(x) \approx f''(x),$$

for arbitrary functions $f$, it is necessary to require that the coefficients of $f$ and $f'$ be $0$, while the coefficient of $f''$ should be $1$. We obtain the system

$$\begin{cases} A+B+C = 0 \\ h(A-C) = 0 \\ \dfrac{h^2}{2}(A+C) = 1 \end{cases}$$

with solution

$$A = C = \frac{1}{h^2},\ B = -\frac{2}{h^2}, \tag{2.13}$$

hence, the formula

$$D_h^{(2)} f(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{2.14}$$

To determine the error, notice that (with these values of the coefficients) in (2.12) we get

$$D_h^{(2)} f(x) \;\approx\; f''(x) + \frac{h^2}{12} f^{(4)}(x).$$

Thus, we have the error

$$(RD_h^{(2)} f)(x) \;=\; f''(x) - D_h^{(2)} f(x) \;\approx\; -\frac{h^2}{12} f^{(4)}(x). \tag{2.15}$$

**Example 2.3.** For $f(x) = \cos x$, approximate $f''(\pi/6)$ using formula (2.14).

**Solution.** The exact value is

$$f''\Big(\frac{\pi}{6}\Big) \;=\; -\cos\frac{\pi}{6} \;=\; -\frac{\sqrt{3}}{2} \;=\; -0.8660.$$

The results of the approximation are shown in Table 3. Note that the ratio column is consistent with the error formula (2.15), i.e. a rate of convergence of $O(h^2)$ .

| $h$ | $D_h^{(2)} f$ | Error | Ratio |
|---|---|---|---|
| $0.5$ | $-0.84813289$ | $-1.789e-2$ | |
| $0.25$ | $-0.86152424$ | $-4.501e-3$ | $3.97$ |
| $0.125$ | $-0.86489835$ | $-1.127e-3$ | $3.99$ |
| $0.0625$ | $-0.86574353$ | $-2.819e-4$ | $4.00$ |
| $0.03125$ | $-0.86595493$ | $-7.048e-5$ | $4.00$ |

Table 3: Example 2.3, $f(x) = \cos x$

■

**Remark 2.4.**

**1.** Alternately, the coefficients $A, B$ and $C$ could have been found by imposing relations $R(e_k) = 0, k = 0, 1, \ldots, d$ from (1.2). In this case, the remainder could be computed using Peano's Theorem.

**2.** Also, the numerical differentiation formula (2.14) (and its error) can be derived with the interpolation method, using (2.8).

**3.** One must be very cautious in using numerical differentiation, because of the sensitivity to errors in the function values. This is especially true if the function values are obtained empirically with relatively large experimental errors, as is common in practice. Numerical differentiation is an *unstable* operation, meaning that even if the approximation of a function is good, that *does not* guarantee that its derivative will be a good approximation for the derivative of the function. Here is such an example: Let

$$f(x) = g(x) + \frac{x^{n^2}}{n},\ n \geq 1,\ x \in [0,1],\ f, g \in C[0,1].$$

Notice that

$$\begin{aligned} ||f-g||_\infty &= \max_{x\in[0,1]} \frac{x^{n^2}}{n} = \frac{1}{n} \to 0,\ n \to \infty, \\ ||f'-g'||_\infty &= \max_{x\in[0,1]} nx^{n^2-1} = n \to \infty. \end{aligned}$$

Numerical derivatives can be used to find numerical methods for (ordinary or partial) differential equations. This is done in order to reduce the differential equation to a form that can be solved more easily than the original equation.

## 3 Numerical Integration

Let $f : [a,b] \to \mathbb{R}$ be integrable on $[a,b]$, $F_k(f), k = \overline{0,m}$ give information on $f$ (usually, linear functionals, such as values or derivatives) and let $w : [a,b] \to \mathbb{R}_+$ be a weight function which is integrable on $[a,b]$.

**Definition 3.1.** *A formula of the type*

$$\int_a^b w(x)f(x)dx = \sum_{j=0}^m A_j F_j(f) + R(f), \tag{3.1}$$

*is called a **numerical integration formula** for the function $f$ or a **quadrature formula**. The parameters $A_j, j = \overline{0,m}$ are called the **coefficients** of the formula, and $R(f)$ the **remainder**.*

**Definition 3.2.** *The natural number d satisfying the property that $\forall f \in \mathbb{P}_d, R(f) = 0$ and $\exists g \in \mathbb{P}_{d+1}$ such that $R(g) \neq 0$ is called **degree of precision** of the quadrature formula* (3.1).

**Remark 3.3.** Since $R$ is o linear functional, it follows that a quadrature formula has degree of precision $d$ if and only if

$$R(e_j) = 0,\ j = 0, 1, \ldots, d,\ R(e_{d+1}) \neq 0. \tag{3.2}$$

If the degree of precision of a quadrature formula is known, then the remainder can be determined using Peano's Theorem.

## 3.1 Interpolatory Quadratures, Newton-Cotes Formulas

Many numerical integration formulas are based on the idea of replacing $f$ by an approximating function whose integral can be evaluated. Most of the times, that approximating function is an interpolation polynomial. Then, we obtain a quadrature formula of the form

$$\int_a^b f(x)dx = \sum_{k=0}^{m} A_k f(x_k) + R(f), \tag{3.3}$$

called an **interpolatory quadrature**. If, in addition, the nodes used are equally spaced, it is called a **Newton-Cotes quadrature**. If the nodes include the endpoints of the interval, $a$ and $b$, then we have a *closed* Newton-Cotes formula, otherwise, an *open* one.
There are $2m+2$ unknowns ($m+1$ nodes and $m+1$ coefficients) in formula (3.3). Imposing conditions (3.2), it follows that the maximum possible degree of precision can be obtained for a polynomial with $2m+2$ coefficients, i.e. of degree $2m+1$, hence, $e_{2m+1}$. Thus, the maximum degree of precision of a quadrature formula (3.3) with $m+1$ nodes is

$$d_{\max} = 2m+1.$$

Any interpolatory numerical integration scheme (3.3) has degree of precision at least $m$ (since the interpolation formula has that degree of exactness).

We start with three of the most widely used (but also, simplest) quadratures, obtained from low degree polynomial interpolation.

#### Rectangle (Midpoint) Rule

We interpolate $f$ at a single *double* node, $x_0 = \dfrac{a+b}{2}$, the midpoint of the interval (hence, the name of the method). So we use the Taylor polynomial of degree 1. Assuming that $f$ has second order continuous derivatives on $(a,b)$, we have

$$\begin{aligned} f(x) &= T_1 f(x) + R_1 f(x) \\ &= f\Big(\frac{a+b}{2}\Big) + \Big(x - \frac{a+b}{2}\Big) f'\Big(\frac{a+b}{2}\Big) + \frac{1}{2!}\Big(x - \frac{a+b}{2}\Big)^2 f''(\xi),\ \xi \in (a,b). \end{aligned}$$

Integrating, we get

$$\begin{aligned}\int_a^b f(x)dx &= (b-a)f\Big(\frac{a+b}{2}\Big)+f'\Big(\frac{a+b}{2}\Big)\int_a^b \Big(x-\frac{a+b}{2}\Big)dx+R(f)\\ &= (b-a)f\Big(\frac{a+b}{2}\Big)+f'\Big(\frac{a+b}{2}\Big)\frac{1}{2}\Big(x-\frac{a+b}{2}\Big)^2\Big|_a^b+R(f)\\ &= (b-a)f\Big(\frac{a+b}{2}\Big)+R(f),\end{aligned}$$

because the second integral is $\frac{1}{2}\Big[\Big(\frac{b-a}{2}\Big)^2-\Big(\frac{b-a}{2}\Big)^2\Big]=0.$
Check the conditions (3.2). We have

$$\begin{aligned}R(e_0) &= \int_a^b e_0(x)dx-(b-a)e_0\Big(\frac{a+b}{2}\Big) \;=\; b-a-(b-a) \;=\; 0,\\ R(e_1) &= \int_a^b xdx-(b-a)\frac{a+b}{2} \;=\; \frac{b^2-a^2}{2}-\frac{b^2-a^2}{2} \;=\; 0.\end{aligned}$$

So, we found the formula

$$\int_a^b f(x)dx \;=\; (b-a)f\Big(\frac{a+b}{2}\Big)+R(f), \tag{3.4}$$

called the **rectangle rule**, an open Newton-Cotes formula, having degree of precision $d=1$, which is the *maximum* possible for a formula with a single node ($m=0$).
We compute the remainder by

$$R(f) \;=\; \frac{f''(\xi)}{2!}\int_a^b \Big(x-\frac{a+b}{2}\Big)^2dx \;=\; \frac{(b-a)^3}{24}f''(\xi),\ \xi\in(a,b). \tag{3.5}$$

Let us see a geometrical interpretation of this formula. Recall that, if $f(x)\geq 0$ for $x\in[a,b]$, the definite integral in (3.4) represents the area of the region that lies below the graph of $f(x)$, above the $Ox$ axis and between the lines $x=a$ and $x=b$. This area is approximated by the area of the *rectangle* with base $b-a$ and height $f\left(\frac{a+b}{2}\right)$ (see Figure 1). Hence, the other name of the method.

**Remark 3.4.** The rectangle rule (3.4) can also be obtained using the method of undetermined coef-

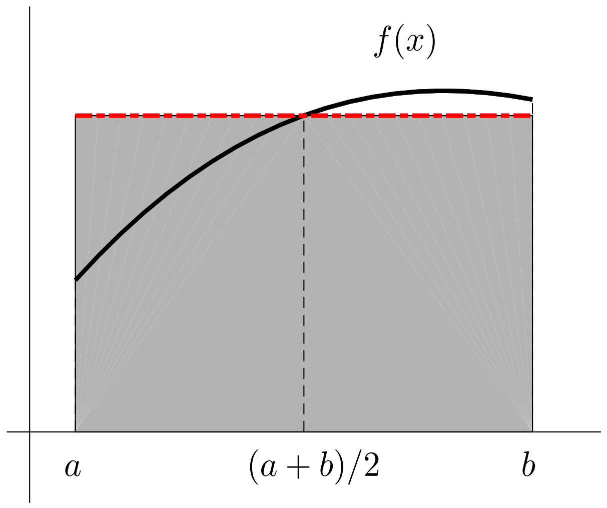

Fig. 1: Geometrical illustration of the rectangle rule

ficients, starting with

$$\int_a^b f(x)dx \;=\; A_0 f(x_0) + R(f)$$

and imposing conditions (3.2). From $R(e_0) = 0$, we get $A_0 = b - a$, and in order to have $R(e_1) = 0$, the node $x_0$ must be the midpoint of the interval, $x_0 = \dfrac{a+b}{2}$.

To improve on the approximation of the integral, break the interval $[a, b]$ into $n$ smaller subintervals determined by the equidistant nodes $x_i = a + ih, i = \overline{0, n}, h = (b - a)/n$, and apply the rectangle rule (3.4) on each subinterval, i.e.,

$$\int_{x_i}^{x_{i+1}} f(x)dx \;=\; hf\Big(\frac{x_i + x_{i+1}}{2}\Big) + \frac{h^3}{24} f''(\xi_i),\; \xi_i \in [x_i, x_{i+1}].$$

We have

$$\int_a^b f(x)dx \;=\; \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx \;=\; h \sum_{i=0}^{n-1} f\Big(\frac{x_i + x_{i+1}}{2}\Big) + \frac{h^3}{24} \sum_{i=0}^{n-1} f''(\xi_i),\; \xi_i \in [x_i, x_{i+1}].$$

Using a mean value formula for the continuous function $f''$,

$$f''(\xi) \;=\; \frac{f''(\xi_0) + \cdots + f''(\xi_{n-1})}{n},\; \xi \in (a,b),$$

we get

$$\int\limits_a^b f(x)dx \;=\; h\sum_{i=0}^{n-1} f\Big(a + \big(i + \frac{1}{2}\big)h\Big) + \frac{h^2(b-a)}{24} f''(\xi),\; \xi \in (a,b), \tag{3.6}$$

called the **composite (repeated) rectangle (midpoint) formula**.

**Trapezoidal Rule**

We proceed similarly, approximating the integrand by the Lagrange interpolation polynomial with 2 nodes, $x_0 = a, x_1 = b$, the endpoints of the interval. If $f$ is twice continuously differentiable on $(a,b)$, we have

$$f(x) \;=\; \frac{x-b}{a-b} f(a) + \frac{x-a}{b-a} f(b) + \frac{f''(\xi)}{2!}(x-a)(x-b),\; \xi \in (a,b).$$

Integrating, after doing all the computations, we get

$$\int\limits_a^b f(x)dx \;=\; \frac{b-a}{2}\Big(f(a) + f(b)\Big) - \frac{(b-a)^3}{12} f''(\xi),\; \xi \in (a,b), \tag{3.7}$$

called the **trapezoidal rule**, a closed Newton-Cotes formula. Again, the name comes from the geometrical interpretation (see Figure 2), where the area of the region that lies between the graph of $f$, the $x-$axis and the lines $x = a$ and $x = b$, is approximated by the area of the trapezoid with bases $f(a), f(b)$ and height $b - a$.

Since this rule is derived from Lagrange interpolation with two nodes (the degree of the interpolation polynomial being 1), we know that its degree of precision is *at least* $d = 1$ (without checking $R(e_0) = R(e_1) = 0$). Let us check if $d > 1$.

$$R(e_2) \;=\; \int\limits_a^b x^2 dx - \frac{b-a}{2}\big(a^2 + b^2\big) \;=\; \frac{1}{3}\big(b^3 - a^3\big) - \frac{b-a}{2}\big(a^2 + b^2\big) \;=\; -\frac{(b-a)^3}{6} \;\neq\; 0.$$

Thus, the degree of precision is $d = 1$.

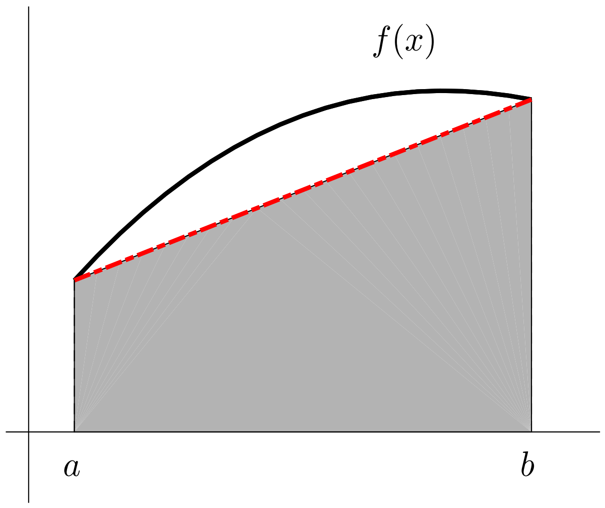

Fig. 2: Geometrical illustration of the trapezoidal rule

Now, just as we did with the rectangle rule, we divide the interval $[a, b]$ into $n$ subintervals $[x_i, x_{i+1}],\ x_i = a + ih, i = \overline{0, n}$, of length $h = \dfrac{b-a}{n}$. We have

$$\int\limits_a^b f(x)dx \;=\; \sum_{i=0}^{n-1} \int\limits_{x_i}^{x_{i+1}} f(x)dx \;=\; \frac{h}{2}\sum_{i=0}^{n-1}\Big(f(x_i) + f(x_{i+1})\Big) - \frac{h^3}{12}\sum_{i=0}^{n-1} f''(\xi_i),\ \xi_i \in [x_i, x_{i+1}]$$

Using again the mean value theorem and denoting by $f_i = f(x_i)$, we get the **composite (repeated) trapezoidal rule**,

$$\int\limits_a^b f(x)dx \;=\; \frac{h}{2}\Big[f(a) + 2\big(f_1 + \cdots + f_{n-1}\big) + f(b)\Big] - \frac{h^2(b-a)}{12} f''(\xi),\ \xi \in (a, b). \quad (3.8)$$

**Remark 3.5.** Obviously, for larger $n$, we get increasingly accurate approximations of the definite integral. But which sequence of values of $n$ should be used? If $n$ is doubled repeatedly, $n \to 2n$, then the function values used in each approximation (3.8) will include all of the earlier function values used in the preceding approximation. Thus, the *doubling* of $n$ will ensure that all previously computed information is used in the new calculation, making the trapezoidal rule less expensive than it would be otherwise.

**Simpson's Rule**

For this formula, we consider Hermite interpolation at the nodes $x_0 = a, x_1 = \dfrac{a+b}{2}$, *double* and $x_2 = b$. Then the corresponding Hermite interpolation polynomial has degree 3 and is of the form

$$\begin{aligned} H_3(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_1](x - x_0)(x - x_1) + \\ &+ f[x_0, x_1, x_1, x_2](x - x_0)(x - x_1)^2. \end{aligned}$$

If $f$ has continuous derivatives of order 4 on $[a, b]$, the error of the approximation can be written as

$$R_3(x) = \frac{(x - x_0)(x - x_1)^2(x - x_2)}{4!} f^{(4)}(\xi),\ \xi \in (a, b).$$

Integrating on $[a, b]$ the relation $f(x) = H_3(x) + R_3(x)$, we get a new closed Newton-Cotes formula,

$$\int\limits_a^b f(x)dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] - \frac{(b-a)^5}{2880} f^{(4)}(\xi),\ \xi \in (a, b), \quad (3.9)$$

called the **(Cavalieri–) Simpson rule**. Its degree of precision is $d = 3$.

Dividing the interval $[a, b]$ into an *even* number $n = 2m$ of subintervals of length $h = \dfrac{b-a}{2m}$, and denoting by $x_i = a + ih, f_i = f(x_i), i = \overline{0, 2m}$, we have

$$\begin{aligned} \int\limits_a^b f(x)dx &= \sum_{i=1}^{m} \int\limits_{x_{2i-2}}^{x_{2i}} f(x)dx \\ &= \sum_{i=1}^{m} \left[\frac{h}{3}\Big(f_{2i-2} + 4f_{2i-1} + f_{2i}\Big) - \frac{h^5}{90} f^{(4)}(\xi_i)\right],\ \xi_i \in [x_{2i-2}, x_{2i}]. \end{aligned}$$

By the mean value theorem, we get the **composite (repeated) Simpson's rule**

$$\begin{aligned} \int\limits_a^b f(x)dx &= \frac{h}{3}\left[f(a) + 4\sum_{i=1}^{m} f_{2i-1} + 2\sum_{i=1}^{m-1} f_{2i} + f(b)\right] \\ &- \frac{h^4(b-a)}{180} f^{(4)}(\xi),\ \xi \in (a, b). \qquad (3.10) \end{aligned}$$

**Remark 3.6.**
**1.** Simpson's formula can be derived by considering interpolation with $3$ *simple* nodes, so a polynomial of degree $2$. We get the same coefficients, but the integral of the remainder will be zero. This is why Hermite interpolation was used instead.
**2.** These are three of the simplest quadrature formulas. The rectangle and trapezoidal rules are comparable precision-wise ($O(h^2)$) and also from the computational cost point of view (number of flops per iteration). The trapezoidal rule is usually preferred when the number of nodes is doubled at each iteration (see Remark 3.5). Simpson's rule is superior in precision ($O(h^4)$), but it also incurs a higher computational load.

**Example 3.7.** Approximate the integral

$$\int_0^1 \frac{1}{1+x}dx$$

using the three methods above.

**Solution.** The exact value of the integral is

$$\int_0^1 \frac{1}{1+x}dx \;=\; \ln(1+x)\Big|_0^1 \;=\; \ln 2 \;=\; 0.693147180559945.$$

By the rectangle rule, we have the approximation

$$\int_0^1 \frac{1}{1+x}dx \;\approx\; 1\cdot f\Big(\frac{1}{2}\Big) \;=\; \frac{2}{3} \;=\; 0.6667,$$

with error

$$E_1 \;=\; 0.0265.$$

Using the trapezoidal rule, we obtain

$$\int_0^1 \frac{1}{1+x}dx \;\approx\; \frac{1}{2}\big(f(0)+f(1)\big) \;=\; \frac{3}{4} \;=\; 0.75,$$

with error

$$E_2 \;=\; -0.0569.$$

Finally, with Simpson's rule, we get

$$\int_0^1 \frac{1}{1+x}dx \;\approx\; \frac{1}{6}\left[f(0)+4f\left(\frac{1}{2}\right)+f(1)\right] \;=\; \frac{25}{36} \;=\; 0.6944,$$

with approximation error

$$E_3 \;=\; -0.0013.$$

■

**Example 3.8.** Let us approximate

$$\int_0^1 e^{-x^2}dx \;=\; 0.746824132812427,$$

with the composite trapezoidal and Simpson's rules.

**Solution.** The approximation errors (as well as the ratio of successive approximations) for the two methods are given in Table 4, for various values of $n$. These confirm the higher rate of convergence, $O(h^4)$, of Simpson's repeated method over the composite trapezoidal rule.

| | Composite Trapezoidal | | Repeated Simpson | |
|---|---|---|---|---|
| $n$ | Error | Ratio | Error | Ratio |
| 2 | $1.55e-2$ | | $3.56e-4$ | |
| 4 | $3.84e-3$ | 4.02 | $3.12e-5$ | 11.4 |
| 8 | $9.59e-4$ | 4.01 | $1.99e-6$ | 15.7 |
| 16 | $2.40e-4$ | 4.00 | $1.25e-7$ | 15.9 |
| 32 | $5.99e-5$ | 4.00 | $7.79e-9$ | 16.0 |
| 64 | $1.50e-5$ | 4.00 | $4.87e-10$ | 16.0 |
| 128 | $3.74e-6$ | 4.00 | $3.04e-11$ | 16.0 |

Table 4: Example 3.8

■

## 3.2 Adaptive and Iterated Quadratures; Romberg's Method

### 3.2.1 Adaptive Quadratures

As seen so far, the errors in numerical integration methods depend not only on the size of the interval, but also on values of certain higher order derivatives of the function to be integrated. Newton-Cotes methods (including the three simple ones, that use low degree polynomial interpolation) work well for smooth integrands (even with a small number of nodes), but perform poorly for functions having large values of higher order derivatives – especially for functions having large oscillations on some subintervals or on the whole interval. As a simple example, consider

$$\int_0^1 \sqrt{x}\,dx = \frac{2}{3}.$$

This integrand has infinite derivative in $x = 0$, but is smooth at points close to $x = 1$.

Generally, numerical integration schemes use evenly spaced nodes. When the function to be integrated has a singularity at some point $\alpha \in [a, b]$, this requires many nodes in the vicinity of that point, to reduce the errors caused by the chaotic behaviour of the function in that neighborhood. But this implies that many more nodes (more than necessary) are used throughout the *entire* interval of integration, increasing (unnecessarily) the computational cost of the method. Ideally, we want to use small subintervals where the derivatives are large, and larger subintervals where the derivatives are small and well-behaved.

A method that does this systematically is called **adaptive quadrature**. The general approach in an adaptive quadrature is to use two different methods on each subinterval, compare the results, and divide the interval when the differences are large. The structure of such an algorithm would be "Divide and conquer".

In Algorithm 3.1 we present an example of a general structure for a recursive adaptive quadrature. The parameter "met" is a function that implements a composite quadrature rule, such as the trapezoidal or Simpson's rule, and $m$ is the number of subintervals.

Unlike other methods, that decide what amount of work is needed to achieve a desired precision, an adaptive quadrature computes only as much as is necessary.

**Algorithm 3.1.** [Adaptive quadrature]

```
function $I = adquad(f, a, b, \varepsilon, met, m)$
        $I1 = met(f, a, b, m)$;
        $I2 = met(f, a, b, 2m)$;
```

```
      if $|I1 - I2| < \varepsilon$ % success
        $I = I2$;
        return
      else   % recursive subdivision
        $I = adquad(f, a, \frac{a+b}{2}, \varepsilon, met, m) + adquad(f, \frac{a+b}{2}, b, \varepsilon, met, m)$;
      end
  end
```

### 3.2.2 Richardson Extrapolation

*Extrapolation* is a method for generating high-accuracy numerical schemes using low-order formulas. The most widely used is *Richardson extrapolation*.

Consider the integral

$$I := \int_a^b f(x)\,dx$$

and a numerical integration scheme

$$I \approx I_n,$$

for which we have an asymptotic error formula of the form

$$I - I_n \approx \frac{c}{n^p}, \tag{3.1}$$

where $c$ depends on $a, b$ and the derivatives of a certain order of the function $f$ on $[a, b]$. The difficulty in using this estimate is not knowing the value of the constant $c$. We can obtain a computable estimate of the error without needing to know $c$ explicitly. We write (3.1) for a larger $n$:

$$I - I_{2n} \approx \frac{c}{(2n)^p} = \frac{c}{2^p n^p} \tag{3.2}$$

and eliminate the unknown $c$ from relations (3.1)-(3.2). We obtain

$$I - I_n \approx 2^p\big(I - I_{2n}\big),$$

and then the approximation

$$I \approx \frac{2^p I_{2n} - I_n}{2^p - 1} = I_{2n} + \frac{I_{2n} - I_n}{2^p - 1} \overset{\text{not}}{=} R_{2n}, \tag{3.3}$$

called **Richardson's extrapolation formula**. From this, we can get another error estimate for $I_{2n}$,

$$I - I_{2n} \approx \frac{I_{2n} - I_n}{2^p - 1}, \tag{3.4}$$

called **Richardson's error estimate**. The term $R_{2n}$ is an improved estimate of $I$, based on using $I_n, I_{2n}, p$ and the assumption (3.1). It is a more accurate approximation to $I$ than is $I_{2n}$. How much more accurate it is depends on the validity of (3.1)–(3.2).

**Example 3.2.** Let us consider a few simple Newton-Cotes formulas.

**Solution.** For the composite trapezoidal rule, if $f$ has continuous second order derivatives on $[a, b]$, we have

$$\begin{aligned}\int\limits_a^b f(x)dx &= \frac{b-a}{n}\Big[f(a) + f(b) + \sum_{i=1}^{n-1} f\Big(a + \frac{b-a}{n}i\Big)\Big] - \frac{(b-a)^3}{12n^2}f''(\xi)\\ &= T_n + \frac{c}{n^2},\end{aligned}$$

hence, $p = 2$. Using Richardson extrapolation, we get

$$I \approx \frac{4T_{2n} - T_n}{3} = T_{2n} + \frac{1}{3}\big(T_{2n} - T_n\big) \tag{3.5}$$

and the error formula

$$I - T_{2n} \approx \frac{1}{3}\big(T_{2n} - T_n\big). \tag{3.6}$$

With Simpson's repeated rule, if $f$ has continuous fourth order derivatives on $[a, b]$ and $f_j = f\Big(a + \dfrac{b-a}{2n}j\Big),\ j = \overline{0, 2n}$, we have

$$\begin{aligned}\int\limits_a^b f(x)dx &= \frac{b-a}{6n}\Big[f(a) + f(b) + 4\sum_{i=1}^{n} f_{2i-1} + 2\sum_{i=1}^{n-1} f_{2i}\Big] - \frac{(b-a)^5}{2880n^4}f^{(4)}(\xi)\\ &= S_n + \frac{c}{n^4},\end{aligned}$$

so in this case, $p = 4$. With Richardson extrapolation, we obtain

$$I \approx \frac{16S_{2n} - S_n}{15} = S_{2n} + \frac{1}{15}\big(S_{2n} - S_n\big) \tag{3.7}$$

and the error

$$I - S_{2n} \approx \frac{1}{15}\big(S_{2n} - S_n\big). \tag{3.8}$$

■

**Example 3.3.** Consider again the problem in Example 3.8 in Lecture 7: the approximation of the integral

$$\int_0^1 e^{-x^2} dx = 0.746824132812427,$$

using the composite trapezoidal rule.

**Solution.** Last time we obtained the errors

| $n$ | Error | Ratio |
|---|---|---|
| 2 | $1.55e-2$ | |
| 4 | $3.84e-3$ | $4.02$ |
| 8 | $9.59e-4$ | $4.01$ |
| 16 | $2.40e-4$ | $4.00$ |
| 32 | $5.99e-5$ | $4.00$ |

Table 1: Errors in repeated trapezoidal rule, Example 3.3

We have

$$T_2 = 0.7313702518,$$
$$T_4 = 0.7429840978.$$

By (3.5), we get the approximation

$$I \approx R_4 = \frac{1}{3}\big(4T_4 - T_2\big) = 0.7468553798,$$

with absolute error

$$0.0000312 \; = \; 3.12e - 5.$$

Notice in Table 1 that the error of $R_4$ is smaller than that of $T_{32}$, so $R_4$ (after only 2 steps) gives a better approximation of $I$ than $T_{32}$, obtained after 5 steps!

Now, let us estimate the error in $T_4$ using Richardson extrapolation. By (3.6), we have

$$I - T_4 \;\approx\; \frac{1}{3}\big(T_4 - T_2\big) \;=\; 0.00387.$$

The actual error in $T_4$ is $0.00384$, so we obtained a very accurate error estimate.

■

**Remark 3.4.** Richardson's extrapolation and error estimation are not always as accurate as this example might suggest, but it is usually a fairly accurate procedure. The main assumption that must be satisfied is (3.1), with a known $p$. And the extrapolation itself provides a way of testing whether this assumption is valid for the actual values of $I_n$ being used: Continue the ideas in (3.1)–(3.3) and write successively

$$\begin{aligned} I - I_n &\approx 2^p\big(I - I_{2n}\big), \\ I - I_{2n} &\approx 2^p\big(I - I_{4n}\big). \end{aligned}$$

We get

$$\begin{aligned} I_{2n} - I_n &= \big(I - I_n\big) - \big(I - I_{2n}\big) \\ &\approx 2^p\big(I - I_{2n}\big) - \big(I - I_{2n}\big) \;=\; (2^p - 1)\big(I - I_{2n}\big). \end{aligned}$$

Similarly, we have

$$\begin{aligned} I_{4n} - I_{2n} &= \big(I - I_{2n}\big) - \big(I - I_{4n}\big) \\ &\approx \big(I - I_{2n}\big) - 2^{-p}\big(I - I_{2n}\big) \;=\; (1 - 2^{-p})\big(I - I_{2n}\big). \end{aligned}$$

Then,

$$\frac{I_{2n} - I_n}{I_{4n} - I_{2n}} \;\approx\; \frac{2^p - 1}{1 - \dfrac{1}{2^p}} \;=\; 2^p.$$

We obtained the (computable) estimate

$$2^p \approx \frac{I_{2n}-I_n}{I_{4n}-I_{2n}},$$

or

$$p \approx \log_2\left(\frac{I_{2n}-I_n}{I_{4n}-I_{2n}}\right) = \frac{1}{\ln 2}\ \ln\left(\frac{I_{2n}-I_n}{I_{4n}-I_{2n}}\right). \tag{3.9}$$

This gives a practical means of checking/finding the value of $p$ in (3.1), using three successive values $I_n, I_{2n}, I_{4n}$.

### 3.2.3 Iterated Quadratures; Romberg's Method

Just like in the case of Lagrange interpolation, we want algorithms for which it is easy to go from one step (iteration) to the next, by using previously computed values of the function.

One drawback of adaptive quadratures is that they compute repeatedly the function values at the nodes and when such an algorithm is executed, there is an extra computational cost due to recursion. *Iterated quadratures* overcome this shortcoming. They apply at the first step a composite quadrature rule and then divide the interval into equal parts using at each step the previously computed approximations. **Romberg's method** is such an iterative algorithm, starting with the composite trapezoidal (or midpoint) rule and then improving the convergence by using Richardson extrapolation.

The initial approximations are obtained by applying either the trapezoid or midpoint rule with $n_k = 2^{k-1}, k \in \mathbb{N}$. Then the value of the step $h_k$ is

$$h_k = \frac{b-a}{n_k} = \frac{b-a}{2^{k-1}}.$$

With these notations, we have (for the trapezoidal rule)

$$\int_a^b f(x)dx = \frac{h_k}{2}\left[f(a)+f(b)+2\sum_{i=1}^{2^{k-1}-1} f\big(a+ih_k\big)\right] - \frac{b-a}{12}h_k^2 f''(\xi_k),\ \xi_k \in [a,b].$$

Denote by $R_{k,1}$ the approximation above, i.e.,

$$R_{1,1} = \frac{h_1}{2}\Big[f(a)+f(b)\Big] = \frac{b-a}{2}\Big[f(a)+f(b)\Big], \tag{3.10}$$

$$\begin{aligned}
R_{2,1} &= \frac{h_2}{2}\Big[f(a)+f(b)+2f\big(a+h_2\big)\Big] \\
&= \frac{b-a}{4}\Big[f(a)+f(b)+2f\Big(a+\frac{1}{2}h_1\Big)\Big] \\
&= \frac{1}{2}\Big[\frac{b-a}{2}\Big(f(a)+f(b)\Big)+(b-a)f\Big(a+\frac{1}{2}h_1\Big)\Big] \qquad (3.11)\\
&= \frac{1}{2}\Big[R_{1,1}+h_1 f\Big(a+\frac{1}{2}h_1\Big)\Big] \\
\dots \\
R_{k,1} &= \frac{1}{2}\Big[R_{k-1,1}+h_{k-1}\sum_{i=1}^{2^{k-2}} f\Big(a+\big(i-\frac{1}{2}\big)h_{k-1}\Big)\Big],\ k=\overline{2,n}.
\end{aligned}$$

Since $h_k = \dfrac{1}{2}h_{k-1}$, each successive level of improvement increases the order of the error term from $O(h^{2k-2})$ to $O(h^{2k})$, so by

$$O(h^2) \;=\; O\Big(\frac{1}{n^2}\Big).$$

Then we can use Richardson extrapolation with $p = 2$, by eliminating the term in $h_k^2$ from the approximation of $I$ by $R_{k-1,1}$ and $R_{k,1}$, respectively. We obtain

$$I \;=\; \frac{4R_{k,1}-R_{k-1,1}}{3}+O(h_k^4)$$

and define

$$R_{k,2} \;=\; \frac{4R_{k,1}-R_{k-1,1}}{3}. \qquad (3.12)$$

We apply Richardson extrapolation to these values, too. In general, if $f \in C^{2n+2}[a,b]$, then, for $k=\overline{1,n}$, we can write

$$\int_a^b f(x)dx \;=\; \frac{h_k}{2}\Big[f(a)+f(b)+2\sum_{i=1}^{2^{k-1}-1} f\big(a+ih_k\big)\Big]+\sum_{i=1}^{k} K_i h_k^{2i}+O(h_k^{2k+2}),$$

where $K_i$ does not depend on $h_k$.

Successively eliminating the powers of $h$ from the relation above, we get

$$R_{k,j} \;=\; \frac{4^{j-1}R_{k,j-1} - R_{k-1,j-1}}{4^{j-1}-1},\; k=\overline{2,n},\; j=\overline{2,k}. \tag{3.13}$$

The computations can be arranged in a table (from (3.11) and (3.13)):

$$\begin{array}{lllll}
R_{1,1} & & & & \\
R_{2,1} & R_{2,2} & & & \\
R_{3,1} & R_{3,2} & R_{3,3} & & \\
\vdots & \vdots & \vdots & \ddots & \\
R_{n,1} & R_{n,2} & R_{n,3} & \dots & R_{n,n}
\end{array}$$

Since the sequence $\{R_{n,1}\}_n$ converges, so does $\{R_{n,n}\}_n$, at a faster rate. We can use the stopping criterion

$$|R_{n-1,n-1} - R_{n,n}| \;<\; \varepsilon.$$

**Remark 3.5.** The second column in Romberg's method corresponds to Simpson's composite rule. We introduce the notation

$$S_{k,1} \;=\; R_{k,2}.$$

Then, the values in the third column are

$$R_{k,3} \;=\; \frac{4^2R_{k,2} - R_{k-1,2}}{4^2-1} \;=\; \frac{16S_{k,1} - S_{k-1,1}}{15},$$

which is Richardson's extrapolation for Simpson's rule. The relation

$$S_{k,1} \;=\; \frac{16S_{k,1} - S_{k-1,1}}{15} \tag{3.14}$$

is at the core of a well-known (and oftenly used) adaptive quadrature algorithm (due to Gander and Gautschi).

**Example 3.6.** Approximate

$$\int_0^{\pi} \sin x \, dx$$

by Romberg's method, with precision $\varepsilon = 10^{-1}$.

**Solution.** The exact value of the integral is

$$I \;=\; -\cos x\Big|_0^{\pi} \;=\; 2.$$

Using the repeated trapezoidal rule with $n_1 = 2^0, h_1 = \pi$ (i.e. nodes $x_0 = 0, x_1 = \pi$) and $n_2 = 2^1, h_2 = \pi/2$ (so nodes $x_0 = 0, x_1 = \pi/2, x_2 = \pi$), we get

$$\begin{aligned} R_{1,1} &= \frac{\pi}{2}(\sin 0 + \sin \pi) \;=\; 0, \\ R_{2,1} &= \frac{1}{2}\Big(R_{1,1} + h_1 \sin\frac{\pi}{2}\Big) \;=\; \frac{\pi}{2} \;=\; 1.5708. \end{aligned}$$

Richardson extrapolation is next:

$$R_{2,2} \;=\; \frac{4R_{2,1} - R_{1,1}}{3} \;=\; \frac{2\pi}{3} \;=\; 2.0944.$$

We have

$$|R_{2,2} - R_{1,1}| \;=\; 2.0944 \;>\; 0.1,$$

so we continue. We compute

$$\begin{aligned} R_{3,1} &= \frac{1}{2}\Big[R_{2,1} + h_2\Big(\sin\frac{\pi}{4} + \sin\frac{3\pi}{4}\Big)\Big] \;=\; 1.8961, \\ R_{3,2} &= \frac{4R_{3,1} - R_{2,1}}{3} \;=\; 2.0046, \\ R_{3,3} &= \frac{16R_{3,2} - R_{2,2}}{15} \;=\; 1.9986 \end{aligned}$$

and

$$|R_{3,3} - R_{2,2}| \;=\; 0.0958 \;<\; 0.1.$$

Hence, we obtained the approximation

$$I \;\approx\; R_{3,3} \;=\; 1.9986,$$

(with an error of $1.4e-3$), which is obviously better than the trapezoidal rule with $n = 4$, $R_{3,1}$ (with the error of $0.1039$). Also, it is more accurate than Simpson's approximation with $4$ nodes,

$I \approx 2.005$ (with error $5e-3$).

In fact, for this example, the algorithm converges very fast, as seen below:

```
0
1.5708   2.0944
1.8961   2.0046   1.9986
1.9742   2.0003   2.0000   2.0000
```

■

## 3.3 Gaussian Quadratures

### 3.3.1 General Framework

**Definition 3.7.** *An interpolatory formula of the form*

$$\int_a^b w(x)f(x)\,dx = \sum_{k=1}^{m} A_k f(x_k) + R_m(f) \tag{3.15}$$

*is called* ***Gaussian quadrature*** *if it has maximum degree of precision,* $d = 2m-1$.

The function $w : (a,b) \to \mathbb{R}_+$ is a *weight function*, a function for which the *moments*

$$\mu_j = \int_a^b w(x)x^j\,dx \tag{3.16}$$

exist and are finite for each $j \in \mathbb{N}$. The purpose of a weight function is to "absorb" some singularities of the integrand.

We want to determine the coefficients $A_k$ and the nodes $x_k$ such that

$$R_m(e_0) = R_m(e_1) = \ldots = R_m(e_{2m-1}) = 0. \tag{3.17}$$

Let us start with a simple example. Consider the integral

$$\int_{-1}^{1} f(x)\,dx. \tag{3.18}$$

So, in this case,

$$w(x) \equiv 1.$$

**Case $m = 1$.**
We seek a numerical integration formula

$$\int_{-1}^{1} f(x)\,dx \approx A_1 f(x_1).$$

From the first two relations in (3.17), we get

$$\begin{aligned} A_1 &= 2, \\ A_1 x_1 &= 0, \end{aligned}$$

and, thus, the formula

$$\int_{-1}^{1} f(x)\,dx \approx 2f(0),$$

which is the midpoint (rectangle) rule.

**Case $m = 2$.**
Now, we want a quadrature of the form

$$\int_{-1}^{1} f(x)\,dx \approx A_1 f(x_1) + A_2 f(x_2),$$

with $4$ unknowns, which are determined from the first $4$ relations (3.17). This leads to the system

$$\begin{aligned} A_1 + A_2 &= 2 \\ A_1 x_1 + A_2 x_2 &= 0 \\ A_1 x_1^2 + A_2 x_2^2 &= \tfrac{2}{3} \\ A_1 x_1^3 + A_2 x_2^3 &= 0 \end{aligned} \tag{3.19}$$

with solution

$$A_1 = A_2 = 1,\ x_1 = -\frac{\sqrt{3}}{3},\ x_2 = \frac{\sqrt{3}}{3}. \tag{3.20}$$

Hence, we found the quadrature formula

$$\int_{-1}^{1} f(x)\,dx \;\approx\; f\Big(-\frac{\sqrt{3}}{3}\Big) + f\Big(\frac{\sqrt{3}}{3}\Big), \tag{3.21}$$

with degree of precision $d = 3$.

**General case $m > 2$.**
In a similar fashion, we obtain the system

$$\begin{aligned}
A_1 + A_2 + \cdots + A_{2m-1} &= 2\\
A_1x_1 + A_2x_2 + \cdots + A_{2m-1}x_{2m-1} &= 0\\
A_1x_1^2 + A_2x_2^2 + \cdots + A_{2m-1}x_{2m-1}^2 &= \frac{2}{3}\\
A_1x_1^3 + A_2x_2^3 + \cdots + A_{2m-1}x_{2m-1}^3 &= 0\\
\cdots \quad & \cdots\\
A_1x_1^{2m-2} + A_2x_2^{2m-2} + \cdots + A_{2m-1}x_{2m-1}^{2m-2} &= \frac{2}{2m-1}\\
A_1x_1^{2m-1} + A_2x_2^{2m-1} + \cdots + A_{2m-1}x_{2m-1}^{2m-1} &= 0.
\end{aligned} \tag{3.22}$$

**Example 3.8.** Consider again the integral in Example3.3,

$$I \;=\; \int_{0}^{1} e^{-x^2} dx \;=\; 0.746824132812427.$$

**Solution.** The linear change of variables

$$x \;=\; \frac{b + a + t(b-a)}{2}$$

maps the interval $[-1, 1]$ to $[a, b]$. So, with the substitution

$$x \;=\; \frac{1+t}{2}, \;\; t \;=\; 2x - 1,$$

we get

$$I \;=\; \frac{1}{2}\int_{-1}^{1} e^{-\frac{1}{4}(1+t)^2}\, dt.$$

We apply Gaussian quadratures to the above integral. The errors are given in Table2.

| $m$ | Error |
|---|---|
| 2 | $2.29e-4$ |
| 3 | $9.55e-6$ |
| 4 | $3.35e-7$ |
| 5 | $6.05e-9$ |
| 6 | $7.77e-11$ |
| 7 | $7.89e-13$ |

Table 2: Gaussian quadratures errors, Example 3.8

Comparing these with the ones given by the composite trapezoid or Simpson's rules (even with extrapolation), we see that these approximations are much more accurate, with fewer nodes.

■

Gaussian quadrature formulas for a general weight function $w$ can be found completely similarly. From relations (3.17), we obtain the system

$$\begin{array}{rcl} A_1 + A_2 + \cdots + A_{2m-1} & = & \mu_1 \\ A_1 x_1 + A_2 x_2 + \cdots + A_{2m-1} x_{2m-1} & = & \mu_2 \\ A_1 x_1^2 + A_2 x_2^2 + \cdots + A_{2m-1} x_{2m-1}^2 & = & \mu_2 \\ \ldots & & \ldots \\ A_1 x_1^{2m-1} + A_2 x_2^{2m-1} + \cdots + A_{2m-1} x_{2m-1}^{2m-1} & = & \mu_{2m-1}. \end{array} \tag{3.23}$$

This system is not linear in the nodes. It is linear in the coefficients, but with a Vandermonde system matrix, which is known to have *conditioning* (stability) problems. Solving this system is not an easy task. Even when a solution can be found (numerically), it is possible that some of the nodes are complex, or have values outside the interval $[a, b]$. Which is why we use another approach, one that involves orthogonal polynomials.

### 3.3.2 Orthogonal Polynomials

The use of orthogonal polynomials is justified by the following result.

**Theorem 3.9.** *Let $u(x) = (x - x_1)(x - x_2) \ldots (x - x_m)$. Then, the quadrature formula* (3.15) *is exact for all polynomials $p \in \mathbb{P}_{2m-1}$ if and only if $u$ is orthogonal to the set $\mathbb{P}_{m-1}$, $u \perp \mathbb{P}_{m-1}$, with* *respect to the inner product*

$$< f, g >_w \;=\; \int\limits_a^b w(x)f(x)g(x)\,dx. \tag{3.24}$$

*Proof.*
"$\Rightarrow$"
Let $p \in \mathbb{P}_{m-1}$. Since $u$ has degree $m$, it follows that $up \in \mathbb{P}_{2m-1}$, so formula (3.15) is exact for $up$,

$$\int\limits_a^b w(x)u(x)p(x)\,dx \;=\; \sum_{k=1}^{m} A_k u(x_k)p(x_k) \;=\; 0,$$

because $u(x_k) = 0, \forall k = \overline{1,m}$. Hence, $u \perp p$ and, further, $u \perp \mathbb{P}_{m-1}$.

"$\Leftarrow$"
Let $f \in \mathbb{P}_{2m-1}$, arbitrary. By the division algorithm, there exist $q, r \in \mathbb{P}_{m-1}$ such that $f \;=\; uq + r$. Thus, we have

$$\int\limits_a^b w(x)f(x)\,dx \;=\; \int\limits_a^b w(x)u(x)q(x)\,dx + \int\limits_a^b w(x)r(x)\,dx \;=\; 0 + \int\limits_a^b w(x)r(x)\,dx,$$

since $u \perp q$.

Now, formula (3.15) is an interpolatory one, and as such, has degree of exactness at least $d = m - 1$. Since $r \in \mathbb{P}_{m-1}$, we have

$$\int\limits_a^b w(x)r(x)\,dx \;=\; \sum_{k=1}^{m} A_k r(x_k).$$

But for any $k = \overline{1,m}$, $f(x_k) \;=\; u(x_k)q(x_k) + r(x_k) \;=\; r(x_k)$ and, thus,

$$\int\limits_a^b w(x)f(x)\,dx \;=\; \sum_{k=1}^{m} A_k f(x_k),$$

i.e. formula (3.15) is exact for every $f \in \mathbb{P}_{2m-1}$. $\square$

**Remark 3.10.** So we now know that the nodes of a Gaussian quadrature are the roots of a poly-

nomial orthogonal to $\mathbb{P}_{m-1}$ with respect to the weight $w$. Such families of orthogonal polynomials have been studied extensively. Table 3 contains such examples. A few immediate conclusions:
**1.** A first consequence is the fact that all the nodes in (3.15) are real, distinct and interior to the interval $(a, b)$.
**2.** There exists a linear recurrence relation between 3 consecutive monic orthogonal polynomials on the interval $[a, b]$ with respect to the weight $w$:

$$\pi_{k+1}(t) \;=\; (t-\alpha_k)\pi_k(t) - \beta_k\pi_{k-1}(t),\; k=0,1,\ldots,\;\; \pi_{-1}(t) \;=\; 0,\;\; \pi_0(t) \;=\; 1, \quad (3.25)$$

where

$$\alpha_k \;=\; \frac{< t\pi_k, \pi_k >}{||\pi_k||^2},\; k=0,1,\ldots,\;\; \beta_k \;=\; \frac{||\pi_k||^2}{||\pi_{k-1}||^2},\; k=1,2,\ldots,\;\; \beta_0 \;=\; \mu_0. \quad (3.26)$$

| **Name** | **Notation** | **Polynomial** | **Weight fn.** | **Interval** | $\boldsymbol{\alpha_k}$ | $\boldsymbol{\beta_k}$ |
|---|---|---|---|---|---|---|
| Legendre | $l_m$ | $\left[(x^2-1)^m\right]^{(m)}$ | $1$ | $[-1,1]$ | $0$ | $\beta_0 = 2$, $\beta_k = (4-k^{-2})^{-1}, k \geq 1$ |
| Chebyshev 1st | $T_m$ | $\cos(m \arccos x)$ | $(1-x^2)^{-\frac{1}{2}}$ | $[-1,1]$ | $0$ | $\beta_0 = \pi$, $\beta_1 = \frac{1}{2}$, $\beta_k = \frac{1}{4}, k \geq 2$ |
| Chebyshev 2nd | $Q_m$ | $\dfrac{\sin[(m+1)\arccos x]}{\sqrt{1-x^2}}$ | $(1-x^2)^{\frac{1}{2}}$ | $[-1,1]$ | $0$ | $\beta_0 = \frac{\pi}{2}$, $\beta_k = \frac{1}{4}, k \geq 1$ |
| Laguerre | $L_m^a$ | $x^{-a}e^x\left(x^{m+a}e^{-x}\right)^{(m)}$ | $x^a e^{-x},\ a > -1$ | $[0,\infty)$ | $2k+a+1$ | $\beta_0 = \Gamma(1+a)$, $\beta_k = k(k+a), k \geq 1$ |
| Hermite | $H_m$ | $(-1)^m e^{x^2}\left(e^{-x^2}\right)^{(m)}$ | $e^{-x^2}$ | $\mathbb{R}$ | $0$ | $\beta_0 = \sqrt{\pi}$, $\beta_k = \frac{k}{2}, k \geq 1$ |

Table 3: Orthogonal polynomials and recurrence coefficients

**Example 3.11.** Now we can solve system (3.19) much easier.

**Solution.** We know that the nodes are the roots of the Legendre polynomial

$$l_2(x) \;=\; \left[(x^2-1)^2\right]'' \;=\; \left[4x(x^2-1)\right]' \;=\; 4(3x^2-1),$$

i.e. $\pm\dfrac{\sqrt{3}}{3}$. Then the coefficients $A_0 = A_1 = 1$ are immediately found.

■

Other properties of Gauss quadratures:

**Proposition 3.12.** *The coefficients $A_k, k = \overline{1,m}$ in* (3.15) *are positive.*

*Proof.* Recall Lagrange fundamental polynomials

$$l_i(x) \;=\; \frac{u_i(x)}{u_i(x_i)} \;=\; \frac{(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_m)}{(x_i-x_1)\dots(x_i-x_{i-1})(x-x_{i+1})\dots(x_i-x_m)},\; i=1,\dots,m.$$

This polynomial has degree $m-1$ and $l_i(x_k) = \delta_{i,k}$. Then the degree of $l_i^2$ is $2m-2$ and $l_i^2(x_k) = \delta_{i,k}$. Hence, formula (3.15) is exact for $f = l_i^2, i = \overline{1,m}$ and we have

$$0 \;<\; \int\limits_a^b w(x) l_i^2(x)\,dx \;=\; \sum_{k=1}^m A_k l_i^2(x_k) \;=\; A_i.$$

□

Regarding the convergence of Gaussian quadratures, we have:

**Theorem 3.13.** *If $[a,b]$ is bounded and $f \in C[a,b]$, then the Gaussian formula* (3.15) *converges, $R_m(f) \to 0,\; m \to \infty$.*

The proof is based on Weierstrass' theorem.

For the remainder of the quadrature formula, the following holds:

**Proposition 3.14.** *If $f \in C^{2m}[a,b]$, then there exists $\xi \in (a,b)$ such that*

$$R_m(f) \;=\; \frac{f^{(2m)}(\xi)}{(2m)!}\int\limits_a^b w(x)u^2(x)\,dx. \tag{3.27}$$

*Proof.* Consider the Hermite interpolation polynomial at the double nodes $x_1,\dots,x_m$ (so a polynomial of degree $2m-1$, for which the Gaussian formula is exact) and its remainder in Newton's form. We have

$$f(x) \;=\; (H_{2m-1}f)(x) + f[x,x_1,x_1,\dots,x_m,x_m]u^2(x).$$

Multiply by $w(x)$ and integrate, keeping in mind that $(H_{2m-1}f)(x_k) = f(x_k), k = \overline{1,m}$.

$$\int\limits_a^b w(x)f(x)\,dx \;=\; \int\limits_a^b w(x)(H_{2m-1}f)(x)\,dx + \int\limits_a^b w(x)u^2(x)f[x,x_1,x_1,\dots,x_m,x_m]\,dx$$

$$= \sum_{k=1}^{m} A_k f(x_k) + \frac{f^{(2m)}(\xi)}{(2m)!} \int_a^b w(x)u^2(x)\,dx,$$

since $w(x)u^2(x) > 0$ on $(a, b)$, so the mean value formula can be used. It follows that the last term represents the remainder of the quadrature formula.

□

Now we have procedures for finding the nodes and coefficients of a Gaussian quadrature formula. However, they are still not enough for an efficient implementation. For that, we will make use of the recurrence relation and the parameters in (3.25)–(3.26). With these, we define

$$J_m(w) = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & & 0 \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \sqrt{\beta_2} & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \sqrt{\beta_{m-1}} \\ 0 & & & \sqrt{\beta_{m-1}} & \alpha_{m-1} \end{bmatrix}, \tag{3.28}$$

called the **Jacobi matrix** of order $m$ for the weight function $w$ on the interval $[a, b]$. The following holds:

**Theorem 3.15.** *The nodes $\{x_k\}_{k=1}^m$ of the Gaussian formula* (3.15) *are the eigenvalues of $J_m$,*

$$J_m v_k = x_k v_k, \; v_k^T v_k = 1, \; k = 1, \dots, m, \tag{3.29}$$

*while the coefficients $\{A_k\}_{k=1}^m$ are given by*

$$A_k = \beta_0 v_{k,1}^2, \; k = 1, \dots, m, \tag{3.30}$$

*where $v_{k,1}$ is the first component of the normalized ($||v_k|| = 1$) eigenvector associated with the eigenvalue $x_k$.*

This is easily proved by writing the recurrence relation (3.25) in matrix (vector) form.

**Remark 3.16.** Thus, the problem of determining a Gauss numerical integration formula is now reduced to that of finding e-values and e-vectors for a *symmetric* and *tridiagonal* matrix. This problem has been studied extensively in linear algebra, there is a vast literature on it and there are many very efficient methods for solving it.

# Chapter 4. Numerical Solution of Systems of Linear Algebraic Equations

Systems of simultaneous linear equations occur in solving problems in a wide variety of disciplines, including Mathematics, Statistics, physical, biological and social sciences, engineering, business and many more. They arise directly in solving real-world problems, and they also occur as part of the solution process for other problems. Numerical solutions of boundary value problems and initial boundary value problems for differential equations are a rich source of linear systems, especially large-size ones.

In this chapter, we will examine the following problem: given a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $b \in \mathbb{R}^n$, find $x \in \mathbb{R}^n$ such that

$$Ax = b.$$

There are two types of methods for the solution of algebraic linear systems:

- *direct (exact)* methods, that provide a solution in a finite number of steps (e.g., Cramer, Gaussian elimination, factorizations);
- *iterative* methods, which approximate the solution by a sequence converging to it (e.g., Jacobi, Gauss-Seidel, SOR).

## 1 Direct Methods

### 1.1 Gaussian Elimination

A linear system is easy to solve when the matrix of the system is *triangular*:

**Definition 1.1.** *A matrix $A = [a_{ij}]_{i,j=\overline{1,n}}$ is called*

- ***upper triangular**, if $a_{ij} = 0, \forall i > j$,*

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ & a_{22} & \dots & a_{2n} \\ & & \ddots & \vdots \\ 0 & & & a_{nn} \end{bmatrix}, \tag{1.1}$$

- ***lower triangular**, if* $a_{ij} = 0, \forall i < j$,

$$A = \begin{bmatrix} a_{11} & & & 0 \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \tag{1.2}$$

- ***diagonal**, if it is both upper and lower triangular,* $a_{ij} = 0, \forall i \neq j$,

$$A = \begin{bmatrix} a_{11} & & & 0 \\ & a_{22} & & \\ & & \ddots & \\ 0 & & & a_{nn} \end{bmatrix}. \tag{1.3}$$

**Remark 1.2.** The determinant of an upper or lower triangular matrix is equal to the product of its diagonal elements

$$\det(A) = a_{11}a_{22}\dots a_{nn}.$$

So, an upper or lower triangular matrix is nonsingular if and only if all of its diagonal entries are nonzero.

**Example 1.3.** Solve the triangular systems

**a)**

$$\begin{bmatrix} 2 & 4 & 2 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} x = \begin{bmatrix} 8 \\ 0 \\ -1 \end{bmatrix}, \tag{1.4}$$

**b)**

$$\begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & 1 & 1 \end{bmatrix} x = \begin{bmatrix} 8 \\ 4 \\ 3 \end{bmatrix}. \tag{1.5}$$

**Solution.**

**a)** The upper triangular system is

$$\begin{cases} 2x_1 + 4x_2 + 2x_3 = 8 \\ \quad\quad\; -x_2 + x_3 = 0 \\ \quad\quad\quad\quad\; -x_3 = -1 \end{cases}$$

We start from the bottom (the last equation) and solve recursively for each unknown:

$$\begin{aligned} x_3 &= \frac{-1}{-1} = 1, \\ x_2 &= \frac{1}{-1}\big(0 - x_3\big) = 1, \\ x_1 &= \frac{1}{2}\big(8 - 4x_2 - 2x_3\big) = 1. \end{aligned}$$

We found the solution

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = [1\ 1\ 1]^T.$$

**b)** For the lower triangular system:

$$\begin{cases} x_1 = 8 \\ 1/2x_1 + x_2 = 4 \\ 1/2x_1 + x_2 + x_3 = 3 \end{cases}$$

we start from the top and solve each equation going down:

$$\begin{aligned} x_1 &= \frac{8}{1} = 8, \\ x_2 &= \frac{1}{1}\Big(4 - \frac{1}{2}x_1\Big) = 0, \\ x_3 &= \frac{1}{1}\Big(3 - \frac{1}{2}x_1 - x_2\Big) = -1. \end{aligned}$$

So the solution is

$$x = \begin{bmatrix} 8 \\ 0 \\ -1 \end{bmatrix} = [8 \;\; 0 \; -1]^T.$$

■

So, in general, for a nonsingular upper triangular matrix $U$, the system $Ux = b$ is easily solved by **backward substitution**:

$$\begin{aligned} x_n &= \frac{b_n}{u_{nn}}, \\ x_i &= \frac{1}{u_{ii}}\Big(b_i - \sum_{j=i+1}^{n} u_{ij}x_j\Big), \; i = \overline{n-1, 1} \end{aligned} \tag{1.6}$$

and if the nonsingular matrix $L$ is lower triangular, then the system $Lx = b$ is solved by **forward substitution**:

$$\begin{aligned} x_1 &= \frac{b_1}{l_{11}}, \\ x_i &= \frac{1}{l_{ii}}\Big(b_i - \sum_{j=1}^{i-1} l_{ij}x_j\Big), \; i = \overline{2, n}. \end{aligned} \tag{1.7}$$

**Gaussian elimination** is a procedure for transforming a system into an equivalent (upper) triangular one, by doing the following elementary row operations:

- multiplying a row (equation) by a constant $\lambda \neq 0$,

$$(\lambda R_i) \rightarrow (R_i),$$

- multiplying a row by a constant $\lambda \neq 0$ and adding it to another row,

$$(R_i + \lambda R_j) \rightarrow (R_i),$$

- interchanging (permuting) two rows,

$$(R_i) \longleftrightarrow (R_j).$$

All these elementary operations are performed on the *augmented (extended)* matrix of the system

$$\widetilde{A} = [A \mid b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \dots & a_{2n} & a_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & a_{n,n+1} \end{array}\right], \tag{1.8}$$

where $a_{i,n+1} = b_i,\ i = \overline{1,n}$.
Gaussian elimination goes as follows:
Assuming $a_{11} \neq 0$, at the first step, we eliminate (make $0$) the coefficients of $x_1$ from every row below, i.e., every $R_j, j = \overline{2,n}$, using $a_{11}$, i.e. by

$$\left(R_j - \frac{a_{j1}}{a_{11}} R_1\right) \to (R_j).$$

Then we proceed the same for the coefficients of each $x_i, i = \overline{2, n-1}, j = \overline{i+1, n}$. This way we obtain a finite sequence of augmented matrices

$$\widetilde{A}^{(1)},\ \widetilde{A}^{(2)},\ \dots,\ \widetilde{A}^{(n)},$$

where $\widetilde{A}^{(1)} = \widetilde{A}$ and (at step $k$) $\widetilde{A}^{(k)} = \left[a_{ij}^{(k)}\right]$ obtained by

$$\left(R_i - \frac{a_{i,k-1}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} R_{k-1}\right) \to (R_i).$$

We denoted by $a_{ij}^{(l)}$ the $(i,j)$ entry at step $l$. The system corresponding to the augmented matrix $\widetilde{A}^{(k)}$ is equivalent to the original linear system and in it, the variable $x_{k-1}$ was eliminated from the equations $E_k, E_{k+1}, \dots, E_n$. Then the system corresponding to $\widetilde{A}^{(n)}$ is an equivalent upper triangular one:

$$\left\{\begin{array}{ccccccccc} a_{11}^{(1)} x_1 & + & a_{12}^{(1)} x_2 & + & \dots & + & a_{1n}^{(1)} x_n & = & a_{1,n+1}^{(1)} \\ & + & a_{22}^{(2)} x_2 & + & \dots & + & a_{2n}^{(2)} x_n & = & a_{2,n+1}^{(2)} \\ & & & \ddots & & & \vdots & & \\ & & & & & & a_{nn}^{(n)} x_n & = & a_{n,n+1}^{(n)} \end{array}\right., \tag{1.9}$$

which is solved by backward substitution (1.6).

Of course, in all this we need $a_{ii}^{(i)} \neq 0$.The element $a_{ii}^{(i)}$ is called **pivot**. If at any time during the elimination process, we find $a_{kk}^{(k)} = 0$, then we look further down in that column for a pivot, i.e., we interchange rows

$$(R_k) \longleftrightarrow (R_p),$$

where $p$ is the smallest integer $k+1 \le p \le n$ with $a_{pk}^{(k)} \ne 0$.
In fact, in practice, when implementing Gaussian elimination, pivoting is necessary even if the pivot is *not* zero, but small, compared to the rest of the elements in that column. That is because such a pivot can produce substantial rounding errors and even cancellations. This can be fixed by doing several types of *pivoting*.

- We can choose the pivot to be the largest element (in absolute value) in that column, below the main diagonal, i.e.

$$\left|a_{pk}^{(k)}\right| = \max_{k \le l \le n} \left|a_{lk}^{(k)}\right|. \tag{1.10}$$

  This is called **partial pivoting (maximal pivoting on columns)**.

- We can do **scaled pivoting on columns**: First, we define a scaling factor for each row

$$s_i = \max_{j=\overline{1,n}} |a_{ij}| \text{ or } s_i = \sum_{j=1}^{n} |a_{ij}|, \; i = \overline{1,n}.$$

  If there exists an $i$ such that $s_i = 0$, then the matrix is singular. For a nonsingular matrix, we use the scaling factor to choose the pivot. At each step $i$, we find the smallest $p, \; i \le p \le n$ such that

$$\frac{|a_{pi}|}{s_i} = \max_{1 \le j \le n} \frac{|a_{ji}|}{s_j} \tag{1.11}$$

  and then interchange rows $(R_i) \longleftrightarrow (R_p)$ so the pivot is $a_{pi}$. This ensures the fact that the maximal element in each column has the relative size $1$, before we compare and interchange rows. Also, dividing by the scaling factor does not produce any extra rounding errors.

- The third method is **total (maximal) pivoting**. At each step $k$, we find

$$|a_{pq}| = \max\{|a_{ij}|, i, j = \overline{k,n}\} \tag{1.12}$$

  and interchange both the rows and the columns,

$$(R_k) \longleftrightarrow (R_p), \; (C_k) \longleftrightarrow (C_q).$$

But then we have to keep track of the columns (unknowns) interchanges.

**Remark 1.4.** If $A$ is singular of rank $p-1$, then at step $p$ we get

$$\widetilde{A}^{(p)} = \left[\begin{array}{cccccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1,p-1}^{(1)} & a_{1p}^{(1)} & \dots & a_{1n}^{(1)} & & a_{1,n+1}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2,p-1}^{(2)} & a_{2p}^{(2)} & \dots & a_{2n}^{(2)} & & a_{2,n+1}^{(2)} \\ \vdots & & \ddots & \vdots & \vdots & & \vdots & & \vdots \\ \vdots & & & a_{p-1,p-1}^{(p-1)} & a_{p-1,p}^{(p-1)} & & a_{p-1,n}^{(p-1)} & & a_{p-1,n+1}^{(p-1)} \\ \vdots & & & & 0 & \dots & 0 & & a_{p,n+1}^{(p)} \\ \vdots & & & & & \ddots & \vdots & & \vdots \\ 0 & \dots & \dots & \dots & \dots & \dots & 0 & & a_{n,n+1}^{(n)} \end{array}\right].$$

So, if $a_{i,n+1}^{(i)} = b_i^{(i)} = 0$, for *all* $i = p, p+1, \dots, n$, then the system is compatible, but undetermined (i.e., it has an infinite number of solutions), otherwise, the system is incompatible (no solution). Thus, Gaussian elimination can also be used to discuss the solvability of the linear system.

**Example 1.5.** Solve the system

$$\begin{cases} x_1 - x_2 + x_3 = -1 \\ -2x_1 + 2x_2 + x_3 = 2 \\ -3x_1 - x_2 + 5x_3 = -5 \end{cases},$$

by Gaussian elimination with different types of pivoting.

**Solution.** The augmented matrix of the system is

$$\widetilde{A} = \left[\begin{array}{rrr|r} 1 & -1 & 1 & -1 \\ -2 & 2 & 1 & 2 \\ -3 & -1 & 5 & -5 \end{array}\right],$$

**Partial pivoting**

On the first column, the largest element in absolute value is $-3$, so that will be the pivot. Thus, first we interchange $(R_1) \longleftrightarrow (R_3)$. We get

$$\widetilde{A} \sim \left[\begin{array}{rrr|r} \boxed{-3} & -1 & 5 & -5 \\ -2 & 2 & 1 & 2 \\ 1 & -1 & 1 & -1 \end{array}\right] \sim \left[\begin{array}{rrr|r} -3 & -1 & 5 & -5 \\ 0 & 8/3 & -7/3 & 16/3 \\ 0 & -4/3 & 8/3 & -8/3 \end{array}\right] \begin{array}{r} (-2/3\, R_1 + R_2) \rightarrow (R_2) \\ (1/3\, R_1 + R_3) \rightarrow (R_3) \end{array}$$

Further, we have

$$\widetilde{A} \sim \left[\begin{array}{rrr|r} -3 & -1 & 5 & -5 \\ 0 & \boxed{8/3} & -7/3 & 16/3 \\ 0 & -4/3 & 8/3 & -8/3 \end{array}\right] \sim \left[\begin{array}{rrr|r} -3 & -1 & 5 & -5 \\ 0 & 8/3 & -7/3 & 16/3 \\ 0 & 0 & 3/2 & 0 \end{array}\right] \quad (\tfrac{1}{2}\,R_2 + R_3) \to (R_3)$$

Now we solve by back substitution (1.6), to get

$$x = [1\ \ 2\ \ 0]^T.$$

**Scaled partial pivoting**

We compute the scaling factors using sums on each row. At the first step $k = 1$, we get

$$\begin{aligned} s &= [3,\ 5,\ 9] \\ \left[\frac{|a_{j,1}|}{s_j}\right] &= [1/3,\ 2/5,\ 3/9] = [5/15,\ 6/15,\ 5/15],\ j = 1, 2, 3. \end{aligned}$$

The maximum of the three fractions is the second, so $p = 2$. We interchange $(R_1) \longleftrightarrow (R_2)$ and make zeros below it.

$$\widetilde{A} \sim \left[\begin{array}{rrr|r} \boxed{-2} & 2 & 1 & 2 \\ 1 & -1 & 1 & -1 \\ -3 & -1 & 5 & -5 \end{array}\right] \sim \left[\begin{array}{rrr|r} -2 & 2 & 1 & 2 \\ 0 & 0 & 3/2 & 0 \\ 0 & -4 & 7/2 & -8 \end{array}\right] \quad \begin{array}{c} (1/2\,R_1 + R_2) \to (R_2) \\ (-3/2\,R_1 + R_3) \to (R_3) \end{array}$$

At step $k = 2$, obviously, $p = 3$, so we interchange $(R_2) \longleftrightarrow (R_3)$,

$$\widetilde{A} \sim \left[\begin{array}{rrr|r} -2 & 2 & 1 & 2 \\ 0 & -4 & 7/2 & -8 \\ 0 & 0 & 3/2 & 0 \end{array}\right]$$

and we are done. By back substitution we get the (obviously, same) solution

$$x = [1\ \ 2\ \ 0]^T.$$

**Total pivoting**

At step $k = 1$, since

$$\max_{i,j=\overline{1,3}} |a_{ij}| = 5 = |a_{33}|,$$

we interchange both rows and columns, $(R_1) \longleftrightarrow (R_3),\ (C_1) \longleftrightarrow (C_3)$, to get

$$\widetilde{A} \sim \left[\begin{array}{rrr|r} -3 & -1 & 5 & -5 \\ -2 & 2 & 1 & 2 \\ 1 & -1 & 1 & -1 \end{array}\right] \sim \left[\begin{array}{rrr|r} 5 & -1 & -3 & -5 \\ 1 & 2 & -2 & 2 \\ 1 & -1 & 1 & -1 \end{array}\right],$$

which is now a system for the *new unknown* $x' = [x_3\ x_2\ x_1]^T$. We proceed to make zeros on the first column below the diagonal.

$$\widetilde{A} \sim \left[\begin{array}{rrr|r} \boxed{5} & -1 & -3 & -5 \\ 1 & 2 & -2 & 2 \\ 1 & -1 & 1 & -1 \end{array}\right] \sim \left[\begin{array}{rrr|r} 5 & -1 & -3 & -5 \\ 0 & 11/5 & -7/5 & 3 \\ 0 & -4/5 & 8/5 & 0 \end{array}\right] \begin{array}{l} (-1/5\, R_1 + R_2) \to (R_2) \\ (-1/5\, R_1 + R_3) \to (R_3) \end{array}$$

At step $k = 2$,

$$\max_{i,j=2,3} |a_{ij}| = \frac{11}{5} = |a_{22}|,$$

so no (row or column) interchanges are necessary. We have

$$\widetilde{A} \sim \left[\begin{array}{rrr|r} 5 & -1 & -3 & -5 \\ 0 & \boxed{11/5} & -7/5 & 3 \\ 0 & -4/5 & 8/5 & 0 \end{array}\right] \sim \left[\begin{array}{rrr|r} 5 & -1 & -3 & -5 \\ 0 & 11/5 & -7/5 & 3 \\ 0 & 0 & 12/11 & 12/11 \end{array}\right] \quad (\tfrac{4}{11}\, R_2 + R_3) \to (R_3)$$

By back substitution, we get

$$x' = [0\ 2\ 1]^T \text{ and } x = [1\ 2\ 0]^T.$$

■

**Remark 1.6.**
**1.** The elements under the main diagonal (which become $0$) need not be computed.
**2.** When pivoting, we do not need to *physically* interchange rows or columns. Just keep one (or two) permutation vector(s) $p$ $(q)$ with $p[i]$ $(q[j])$ meaning that the row (column) $p$ $(q)$ has been interchanged with row (column) $i$ $(j)$. This is especially a good solution if matrices are stored row by row or column by column.
**3.** Gaussian elimination can be used to find the inverse $A^{-1}$ of a nonsingular matrix. For each $k = \overline{1,n}$, column $k$ of $A^{-1}$ can be found by solving the system $Ax = e_k$, where $\{e_k\}$ is the canonical basis of $\mathbb{R}^n$, $e_k = [0\ 0\ \ldots\ 0\ 1\ 0\ \ldots\ 0]^T$, with $1$ on the $k$th slot. Alternatively, the

inverse of $A$ can be found by Gaussian elimination on the matrix

$$[A \mid I] \ \sim \ \ldots \sim \ [I \mid A^{-1}].$$

**4.** Let us assess the computational cost of Gaussian elimination. At step $k = 1$, we perform $n-1$ divisions, $(n-1)n$ multiplications and $(n-1)n$ additions, so a total of $2n(n-1) + (n-1)$ flops. At step $k = 2$, there are $2(n-1)(n-2) + (n-2)$ flops and so on until step $k = n-1$. So, the actual elimination process requires

$$\sum_{k=1}^{n-1} \Big[2(n-k)(n-k+1) + (n-k)\Big] \ = \ \sum_{i=1}^{n-1} \Big[2i(i+1) + i\Big] \ = \ \frac{n(n-1)(4n+7)}{6}$$

flops. Back substitution adds another

$$1 + 3 + \cdots + 2n - 1 \ = \ \sum_{i=1}^{2n-1} i - 2\sum_{i=1}^{n-1} i \ = \ n^2$$

flops, for a total of

$$\frac{n(4n^2 + 9n - 7)}{6} \ = \ O\Big(\frac{2}{3}n^3\Big)$$

flops. For comparison, if the determinants in Cramer's rule are computed using expansion by minors, then the operation count is $(n+1)!$. For $n = 10$, Gaussian elimination uses about $805$ operations, while Cramer's rule uses around $3,628,800$ operations. This should emphasize the point that Cramer's rule is not a practical computational tool, and that it should be considered as just a theoretical mathematics tool.

## 1.2 Factorization Based Methods

These are methods using the fact that the matrix of coefficients of a linear system being solved can be *factored (decomposed)* into the product of two triangular matrices.

### 1.2.1 LU Factorization

**Theorem 1.7.** *If no row interchanges are necessary in the Gaussian elimination process for solving the system $Ax = b$, then $A$ can be factored as*

$$A \ = \ LU, \tag{1.13}$$

*where $L$ and $U$ are lower and upper triangular matrices, respectively. The pair $(L, U)$ is called an $LU$* ***factorization (decomposition)*** *of the matrix $A$.*

*Sketch of Proof.* The first step is to partition $A$ as

$$A = \left[\begin{array}{c|ccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \hline a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array}\right] = \begin{bmatrix} a_{11} & w^* \\ v & A' \end{bmatrix},$$

where $v$ is a column vector of length $n-1$, $w^*$ is a row vector of length $n-1$ and $A'$ is an $(n-1)\times(n-1)$ matrix. Then we can factor $A$ as

$$A = \begin{bmatrix} a_{11} & w^* \\ v & A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & w^* \\ 0 & A' - vw^*/a_{11} \end{bmatrix}.$$

The matrix $A' - vw^*/a_{11}$ is called the **Schur complement** of $A$ with respect to $a_{11}$.
Then, we proceed recursively:

$$A' - vw^*/a_{11} = L'U'.$$

So

$$\begin{aligned} A &= \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & w^* \\ 0 & A' - vw^*/a_{11} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & w^* \\ 0 & L'U' \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ v/a_{11} & L' \end{bmatrix} \begin{bmatrix} a_{11} & w^* \\ 0 & U' \end{bmatrix} \end{aligned}$$

until we get a scalar (a $1 \times 1$ matrix) that can no longer be partitioned.

□

**Remark 1.8.**
**1.** If $A = LU$, then solving the system $Ax = b$ is reduced to solving two triangular systems

$$\begin{aligned} Ly &= b \text{ and} \\ Ux &= y. \end{aligned} \qquad (1.14)$$

**2.** If all that is required is that $L$ be lower and $U$ be upper triangular, then the $LU$ decomposition is *not* unique. We can make it unique by imposing more conditions. For instance, if we require $l_{ii} = 1, i = \overline{1,n}$, we have *Doolittle* factorization and if we impose $u_{ii} = 1, i = \overline{1,n}$, we get the *Crout* factorization. The procedure described in the proof of Theorem 1.7 leads to Doolittle factorization.

**3.** The matrix $U = [u_{ij}]$ in the Doolittle factorization is the upper triangular matrix obtained by Gaussian elimination,

$$u_{ij} \;=\; a_{i,j}^{(i)},\;\; i \le j, \tag{1.15}$$

while $L = [l_{ij}]$ is the matrix of the *multipliers*

$$l_{ij} \;=\; m_{ij} \;=\; \frac{a_{i,j}^{(j)}}{a_{j,j}^{(j)}},\; i \ge j. \tag{1.16}$$

**4.** Examples of cases when no row interchanges are necessary:

- $A$ is **diagonally dominant on rows**,

$$|a_{ii}| \;>\; \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|,\; i = \overline{1,n}. \tag{1.17}$$

- $A$ is **positive definite**,

$$x^T A x \;>\; 0,\; \forall x \neq 0. \tag{1.18}$$

**Example 1.9.** Use $LU$ decomposition to solve the system

$$\left\{\begin{array}{ccccccc} 2x_1 & + & 4x_2 & + & 2x_3 & = & 8 \\ x_1 & + & x_2 & + & 2x_3 & = & 4 \\ x_1 & + & x_2 & + & x_3 & = & 3 \end{array}\right.$$

**Solution.** We have

$$A \;=\; \begin{bmatrix} 2 & 4 & 2 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \;=\; \left[\begin{array}{c|cc} 2 & 4 & 2 \\ \hline 1 & 1 & 2 \\ 1 & 1 & 1 \end{array}\right],$$

so, at the first step,

$$a_{11} = 2, \; v = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; w^* = [4 \;\; 2], \; A' = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}, \; \left[\begin{array}{c|cc} 2 & 4 & 2 \\ \hline 1/2 & & \\ 1/2 & & \end{array}\right].$$

The first Schur complement is

$$A' - vw^*/a_{11} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}[4 \;\; 2] = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} - \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix}$$

and, for now, we have

$$\left[\begin{array}{c|cc} 2 & 4 & 2 \\ \hline 1/2 & -1 & 1 \\ 1/2 & -1 & 0 \end{array}\right] = \left[\begin{array}{c|c|c} 2 & 4 & 2 \\ \hline 1/2 & -1 & 1 \\ \cline{3-3} 1/2 & -1 & 0 \end{array}\right] = \left[\begin{array}{c|c|c} 2 & 4 & 2 \\ \hline 1/2 & -1 & 1 \\ \cline{2-3} 1/2 & 1 & \end{array}\right].$$

The last Schur complement is

$$0 - (-1)/(-1) \cdot 1 = -1$$

and the final decomposition is

$$\left[\begin{array}{c|c|c} 2 & 4 & 2 \\ \hline 1/2 & -1 & 1 \\ \cline{2-3} 1/2 & 1 & -1 \end{array}\right].$$

We take the upper triangular part (*including* the main diagonal) for $U$ and the lower triangular part (*without* the main diagonal) for $L$ (which will have all 1's on the main diagonal), to get

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & 1 & 1 \end{bmatrix}, \; U = \begin{bmatrix} 2 & 4 & 2 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

and check that indeed $A = LU$.

Now, solve $Ly = b = [8 \;\; 4 \;\; 3]^T$, which, from Example 1.3b) has solution $y = [8 \;\; 0 \; -1]^T$ and then $Ux = [8 \;\; 0 \; -1]^T$, which, from Example 1.3a), gives the solution

$$x = [1 \;\; 1 \;\; 1]^T.$$

Check that $Ax = b$. ■

### 1.2.2 LUP Factorization

What if row interchanges (pivoting) *are* necessary? A row interchange is a permutation of two rows. We keep track of those in a *permutation* matrix, which is simply a matrix obtained from the corresponding identity matrix $I$ by permuting rows. So, for a matrix $A$ we find its $LUP$ **factorization (decomposition)**, i.e., a triplet $(L, U, P)$, with $L$ a lower triangular, $U$ an upper triangular and $P$ a permutation matrix, such that

$$PA = LU. \tag{1.19}$$

**Remark 1.10.**
**1.** Solving the system $Ax = b$ is now equivalent to solving two triangular systems

$$\begin{aligned} Ly &= Pb \text{ and} \\ Ux &= y. \end{aligned} \tag{1.20}$$

**2.** Multiplication of a matrix $A$ to the *left* by a permutation matrix $P$ will yield the same *row* interchanges on the matrix $A$ as in $P$, while multiplication on the *right* will result in the same *column* interchanges in $A$ as in $P$.
**3.** The procedure for obtaining an $LUP$ factorization is similar to the previous one, while keeping track of the row interchanges in a permutation matrix $P$.

**Example 1.11.** Find an $LUP$ factorization for the matrix

$$A = \begin{bmatrix} 2 & 1 & -2 \\ 1 & 1 & -1 \\ 3 & -1 & 1 \end{bmatrix}.$$

**Solution.** At the first step, we do partial pivoting and interchange $(R_1) \longleftrightarrow (R_3)$.
At each row interchange, instead of writing the entire matrix $P$, we only emphasize which rows are permuted. Other than that, we proceed as before. We have

$$A \sim \begin{bmatrix} 3 & -1 & 1 \\ 1 & 1 & -1 \\ 2 & 1 & -2 \end{bmatrix} = \left[\begin{array}{c|cc} 3 & -1 & 1 \\ \hline 1 & 1 & -1 \\ 2 & 1 & -2 \end{array}\right] \sim \left[\begin{array}{c|cc} 3 & -1 & 1 \\ \hline 1/3 & & \\ 2/3 & & \end{array}\right], \quad \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}.$$

Schur complement

$$\begin{bmatrix} 1 & -1 \\ 1 & -2 \end{bmatrix} - \frac{1}{3}\begin{bmatrix} 1 \\ 2 \end{bmatrix}[-1\ \ 1] = \begin{bmatrix} 1 & -1 \\ 1 & -2 \end{bmatrix} - \begin{bmatrix} -1/3 & 1/3 \\ -2/3 & 2/3 \end{bmatrix} = \begin{bmatrix} 4/3 & -4/3 \\ 5/3 & -8/3 \end{bmatrix},$$

so,

$$A \sim \left[\begin{array}{c|cc} 3 & -1 & 1 \\ \hline 1/3 & 4/3 & -4/3 \\ 2/3 & 5/3 & -8/3 \end{array}\right] \sim \left[\begin{array}{c|cc} 3 & -1 & 1 \\ \hline 2/3 & 5/3 & -8/3 \\ 1/3 & 4/3 & -4/3 \end{array}\right], \quad \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix},$$

because we interchanged $(R_2) \longleftrightarrow (R_3)$. Further, we have

$$A \sim \left[\begin{array}{c|c|c} 3 & -1 & 1 \\ \hline 2/3 & 5/3 & -8/3 \\ \cline{3-3} 1/3 & 4/5 & \end{array}\right] \sim \left[\begin{array}{c|c|c} 3 & -1 & 1 \\ \hline 2/3 & 5/3 & -8/3 \\ \cline{3-3} 1/3 & 4/5 & 4/5 \end{array}\right],$$

the last Schur complement being

$$-\frac{4}{3} - \frac{3}{5}\cdot\frac{4}{3}\Big(-\frac{8}{3}\Big) = \frac{4}{5}.$$

So, we obtained

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2/3 & 1 & 0 \\ 1/3 & 4/5 & 1 \end{bmatrix}, \ U = \begin{bmatrix} 3 & -1 & 1 \\ 0 & 5/3 & -8/3 \\ 0 & 0 & 4/5 \end{bmatrix}, \ P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Check that $PA = LU$.

■

**Remark 1.12.**
**1.** The computational cost for $LU$ (and $LUP$) factorization is about the same as for Gaussian elimination, $O(n^3)$ flops. However, for *tridiagonal* matrices, that cost drops to $O(n)$ operations. The *Thomas algorithm*, based on $LUP$ decomposition is an efficient way of solving tridiagonal matrix systems. In addition, only three one-dimensional arrays for the three diagonals are needed to store the matrix. This means that very large systems can be solved rapidly and efficiently, and systems of order over $n = 10,000$ are not unusual in some applications, for example, in solving boundary value problems for differential equations.

**2.** More generally, a *band* or *banded* matrix is a sparse matrix whose non-zero entries are confined to a diagonal band, comprising of the main diagonal and zero or more diagonals on either side. If all matrix elements are zero outside a diagonally bordered band whose range is determined by constants $k_1, k_2 \geq 0$,

$$a_{ij} \;=\; 0, \text{ if } \; j < i - k_1 \; \text{ or } \; j > i + k_2$$

then the quantities $k_1$ and $k_2$ are called the *lower bandwidth* and *upper bandwidth*, respectively. The *bandwidth* of the matrix is then defined as

$$w \;=\; \max\{k_1, k_2\},$$

i.e., it is the number $w$ such that

$$a_{ij} \;=\; 0, \text{ if } \; |i - j| > w.$$

It can be shown that $LU$ factorization with partial pivoting for $n \times n$ banded matrices with bandwidth $w$ requires $O(w^2 n)$ flops, while triangular solvers require $O(wn)$ flops.

### 1.2.3 QR Factorization

**Definition 1.1.** *A real square matrix $Q$ is called* ***orthogonal*** *if*

$$Q \cdot Q^T \;=\; Q^T \cdot Q \;=\; I. \tag{1.1}$$

**Theorem 1.2.** *Let $A$ be a real square matrix. Then there exist unique matrices $Q$ and $R$ such that*

$$A \;=\; QR, \tag{1.2}$$

*with $Q$ orthogonal and $R$ upper triangular with positive elements on the main diagonal, $r_{ii} > 0,\ \forall i$. The pair $(Q, R)$ is called the* ***QR factorization*** *of $A$.*

**Remark 1.3.**
**1.** If $A = QR$, then solving the system $Ax = b$ is equivalent to solving the upper triangular systems

$$Rx \;=\; Q^T b. \tag{1.3}$$

**2.** Relation (1.1) automatically implies that any orthogonal matrix is nonsingular with $Q^{-1} \;=\; Q^T$. Orthogonal matrices are very useful in Numerical Analysis, as they preserve lengths, angles, and do not magnify errors.

### 1.2.4 Cholesky Factorization

**Definition 1.4.** *A real square matrix $A$ is called* ***positive definite****, if*

$$x^T A\, x \;=\; \sum_{i,j=1}^{n} a_{ij} x_i x_j \;>\; 0,\ \forall x \in \mathbb{R}^n,\ x \neq 0. \tag{1.4}$$

Symmetric positive definite matrices can be decomposed into triangular factors twice as fast as general matrices. The standard algorithm for this, *Cholesky factorization*, is a variant of Gaussian elimination, which operates both on the left and the right of the matrix at once, preserving and exploiting the symmetry. These matrices have many interesting properties. Among them, the fact that a symmetric matrix is positive definite if and only if all its e-values are real and positive. Also, the e-vectors corresponding to distinct e-values of such a matrix, are orthogonal. Systems having symmetric positive definite matrices play an important role in Numerical Linear Algebra and its applications. Many matrices that arise in physical systems are symmetric and positive definite because of the fundamental physical laws.

**Theorem 1.5.** *Let $A$ be a symmetric positive definite matrix. Then $A$ has a unique* ***Cholesky factorization***

$$A = R^T R, \tag{1.5}$$

*where $R$ is an upper triangular matrix with positive elements on the main diagonal, $r_{ii} > 0,\ \forall i$.*

*Sketch of Proof.* First off, let us use (1.4) for

$$x = e_1 = [1\ 0\ \dots\ 0\,]^T.$$

We get

$$\begin{aligned} x^T A x &= [1\ 0\ \dots\ 0\,] \begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= [1\ 0\ \dots\ 0\,] \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} = a_{11}. \end{aligned}$$

So, any positive definite matrix $A$ has $a_{11} > 0$ and we can set $\alpha = \sqrt{a_{11}}$. Then we proceed in a similar way as with LU factorization, keeping in mind that $A$ is also symmetric.

$$\begin{aligned} A &= \begin{bmatrix} a_{11} & w^* \\ w & A' \end{bmatrix} \\ &= \begin{bmatrix} \alpha & 0 \\ w/\alpha & I_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A' - ww^*/a_{11} \end{bmatrix} \begin{bmatrix} \alpha & w^*/\alpha \\ 0 & I_{n-1} \end{bmatrix} = R_1^T A_1 R_1. \end{aligned}$$

By induction, all matrices that appear during the factorization are positive definite and so, the process cannot break down. This procedure is repeated until

$$A = \underbrace{R_1^T R_2^T \dots R_n^T}_{R^T} \underbrace{R_n R_{n-1} \dots R_1}_{R} = R^T R.$$

The uniqueness follows from the fact that at each step, the value $\alpha = \sqrt{a_{11}}$ is uniquely determined from the factorization and once $\alpha$ is determined, all the rest of the $R_i$'s are also uniquely determined.

$\square$

**Remark 1.6.** This method requires only $n(n+1)/2$ storage locations for $R$, rather than the usual $n^2$ locations. Since only half the matrix needs to be stored, it follows that half of the arithmetic operations can be avoided and the number of operations is about $O\Big(\frac{1}{3}n^3\Big)$, rather than the number $O\Big(\frac{2}{3}n^3\Big)$ required for the usual LU decomposition.

**Example 1.7.** Find the Cholesky factorization (if it exists) of the matrix

$$A = \begin{bmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{bmatrix}.$$

**Solution.** The matrix is symmetric and its e-values are

$$0.0188, \quad 15.5040, \quad 123.4772,$$

real and positive. Therefore, $A$ is positive definite and has a Cholesky decomposition. We have

$$A = \left[\begin{array}{r|rr} 4 & 12 & -16 \\ \hline 12 & 37 & -43 \\ -16 & -43 & 98 \end{array}\right] \sim \left[\begin{array}{r|rr} \sqrt{4} & 6 & -8 \\ \hline 6 & & \\ -8 & & \end{array}\right].$$

The first Schur complement is

$$\begin{aligned} A' - ww^*/a_{11} &= \begin{bmatrix} 37 & -43 \\ -43 & 98 \end{bmatrix} - \begin{bmatrix} 6 \\ -8 \end{bmatrix} [6 \; -8] \\ &= \begin{bmatrix} 37 & -43 \\ -43 & 98 \end{bmatrix} - \begin{bmatrix} 36 & -48 \\ -48 & 64 \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ 5 & 34 \end{bmatrix} \end{aligned}$$

and

$$A \sim \left[\begin{array}{r|rr} 2 & 6 & -8 \\ \hline 6 & 1 & 5 \\ -8 & 5 & 34 \end{array}\right] \sim \left[\begin{array}{r|r|r} 2 & 6 & -8 \\ \hline 6 & \sqrt{1} & 5 \\ \cline{3-3} -8 & 5 & \sqrt{9} \end{array}\right] = \begin{bmatrix} 2 & 6 & -8 \\ 6 & 1 & 5 \\ -8 & 5 & 3 \end{bmatrix},$$

with the last Schur complement being $34 - 5 \cdot 5 = 9$ and its square root 3. Then

$$R = \begin{bmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{bmatrix} \quad \text{and} \quad A = R^T R.$$

■

## 2 Iterative Methods

The linear systems $Ax = b$ that occur in many applications can have a very large order ($10^3, 10^5, 10^6$). For such systems, the Gaussian elimination method (and consequent factorization methods) of the last section is often too expensive in either computation time or computer memory requirements, or possibly both. Moreover, the accumulation of round-off errors can sometimes prevent the numerical solution from being accurate. As an alternative, such linear systems are usually solved with *iteration methods*. In an iterative method, a sequence of progressively accurate iterates is produced to approximate the solution. Thus, in general, we do not expect to get the *exact* solution in a finite number of iteration steps, even if the round-off error effect is not taken into account. In the study of iteration methods, a most important issue is the *convergence property*. We provide a framework for the convergence analysis of a general iteration method.

### 2.1 Jacobi and Gauss-Seidel Methods

We begin with some numerical examples that illustrate two popular iteration methods. Following that, we give a more general discussion of iteration methods.

Consider the linear system

$$\begin{array}{rcrcrcl} 9x_1 & + & x_2 & + & x_3 & = & b_1 \\ 2x_1 & + & 10x_2 & + & 3x_3 & = & b_2 \\ 3x_1 & + & 4x_2 & + & 11x_3 & = & b_3 \end{array} \tag{2.1}$$

We proceed as follows: in the equation numbered $k$, solve for $x_k$ in terms of the remaining unknowns. In the above case,

$$\begin{aligned} x_1 &= \tfrac{1}{9}\left[b_1 - x_2 - x_3\right] \\ x_2 &= \tfrac{1}{10}\left[b_2 - 2x_1 - 3x_3\right] \\ x_3 &= \tfrac{1}{11}\left[b_3 - 3x_1 - 4x_2\right] \end{aligned} \tag{2.2}$$

Let

$$x^{(0)} = \left[x_1^{(0)}, x_2^{(0)}, x_3^{(0)}\right]^T$$

be an initial guess of the true solution $x$. Then define an iteration sequence:

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{9}\left[b_1 - x_2^{(k)} - x_3^{(k)}\right] \\ x_2^{(k+1)} &= \frac{1}{10}\left[b_2 - 2x_1^{(k)} - 3x_3^{(k)}\right] \\ x_3^{(k+1)} &= \frac{1}{11}\left[b_3 - 3x_1^{(k)} - 4x_2^{(k)}\right] \end{aligned} \tag{2.3}$$

for $k = 0, 1, \ldots$ . This is called the **Jacobi iteration method** or the *method of simultaneous replacements (substitution).*

| $k$ | $x_1^{(k)}$ | $x_2^{(k)}$ | $x_3^{(k)}$ | Error | Ratio |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $2.00e+0$ | |
| 1 | 1.1111 | 1.9000 | 0 | $1.00e+0$ | 0.500 |
| 2 | 0.9000 | 1.6778 | −0.9939 | $3.22e-1$ | 0.322 |
| 3 | 1.0351 | 2.0182 | −0.8556 | $1.44e-1$ | 0.448 |
| 4 | 0.9819 | 1.9496 | −1.0162 | $5.06e-2$ | 0.349 |
| 5 | 1.0074 | 2.0085 | −0.9768 | $2.32e-2$ | 0.462 |
| 6 | 0.9965 | 1.9915 | −1.0051 | $8.45e-3$ | 0.364 |
| 7 | 1.0015 | 2.0022 | −0.9960 | $4.03e-3$ | 0.477 |
| 8 | 0.9993 | 1.9985 | −1.0012 | $1.51e-3$ | 0.375 |
| 9 | 1.0003 | 2.0005 | −0.9993 | $7.40e-4$ | 0.489 |
| 10 | 0.9999 | 1.9997 | −1.0003 | $2.83e-4$ | 0.382 |
| 30 | 1.0000 | 2.0000 | −1.0000 | $3.01e-11$ | 0.447 |
| 31 | 1.0000 | 2.0000 | −1.0000 | $1.35e-11$ | 0.447 |

Table 1: Jacobi iteration for solving system (2.1)

In Table 1, we give a number of the iterations for the case that $b = [10, 19, 0]^T$, which yields the true solution

$$x = [1, 2, -1]^T.$$

In the table, the error is computed as

$$||x - x^{(k)}|| = \max_{1\le i\le n} |x_i - x_i^{(k)}|.$$

Notice that the errors decrease as $k$ increases and the values of the ratio eventually approach a limiting constant of approximately $0.447$ as $k$ becomes much larger.

As another approach to the iterative solution of system (2.1) through the use of (2.2), we use *all* the information we obtain in the calculation of each new component. Specifically, let us define

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{9}\left[b_1 - x_2^{(k)} - x_3^{(k)}\right] \\ x_2^{(k+1)} &= \frac{1}{10}\left[b_2 - 2x_1^{(k+1)} - 3x_3^{(k)}\right] \\ x_3^{(k+1)} &= \frac{1}{11}\left[b_3 - 3x_1^{(k+1)} - 4x_2^{(k+1)}\right] \end{aligned} \tag{2.4}$$

for $k = 0, 1, \ldots$ . This is called the **Gauss-Seidel iteration method** or the *method of successive replacements (substitution)*. This method is usually more rapidly convergent than the Jacobi method.

In Table 2, we give a number of iterations for solving the system (2.1). Compare these results to those in Table 1. The speed of convergence is much faster than with the Jacobi method (2.3). The values of the ratio, however, do not appear to approach a limiting value, even when looking at values of $k$ larger than those in the table.

| $k$ | $x_1^{(k)}$ | $x_2^{(k)}$ | $x_3^{(k)}$ | Error | Ratio |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $2.00e+0$ | |
| 1 | 1.1111 | 1.6778 | $-0.9131$ | $3.22e-1$ | 0.161 |
| 2 | 1.0262 | 1.9687 | $-0.9958$ | $3.13e-2$ | 0.097 |
| 3 | 1.0030 | 1.9981 | $-1.0001$ | $3.00e-3$ | 0.096 |
| 4 | 1.0002 | 2.0000 | $-1.0001$ | $2.24e-4$ | 0.074 |
| 5 | 1.0000 | 2.0000 | $-1.0000$ | $1.65e-5$ | 0.074 |
| 6 | 1.0000 | 2.0000 | $-1.0000$ | $2.58e-6$ | 0.155 |

Table 2: Gauss-Seidel iteration for solving system (2.1)

## 2.2 Iterative Methods – General Theory

To understand the behavior of iteration methods, it is best to put them into a vector-matrix format. To this end, we recall some notions and results from Linear Algebra.

**Definition 2.1.** *Let $A \in \mathbb{R}^{n\times n}$.*
*– The polynomial $p(\lambda) = \det(A - \lambda I_n)$ is called the **characteristic polynomial of** $A$ and the equation $p(\lambda) = 0$ the **characteristic equation of** $A$.*
*– The roots of $p(\lambda)$ are called **eigenvalues (e-values) of** $A$.*
*– If $\lambda \in \mathbb{C}$ is an e-value of $A$, a vector $x \in \mathbb{R}^n, x \neq 0$ satisfying $(A - \lambda I_n)x = 0$ is called an **eigenvector (e-vector) of** $A$, corresponding to the e-value $\lambda$.*
*– The set of all e-values of $A$, denoted by $\lambda(A)$ is called the **spectrum of** $A$.*
*– The value $\rho(A) = \max\{|\lambda| \mid \lambda \in \lambda(A)\}$ is called the **spectral radius of** $A$.*
*– The value $tr(A) = a_{11} + \cdots + a_{nn}$ is called the **trace of** $A$.*

**Definition 2.2.** *A **matrix norm** is a function $||\cdot|| : \mathbb{R}^{n\times n} \to \mathbb{R}$ satisfying the conditions: $\forall A, B \in \mathbb{R}^{n\times n}, \forall \alpha \in \mathbb{R}$,*
*(i)* $||A|| \geq 0,\ ||A|| = 0 \Leftrightarrow A = 0_n$.
*(ii)* $||\alpha A|| = |\alpha| \cdot ||A||$.
*(iii)* $||A + B|| \leq ||A|| + ||B||$.
*(iv)* $||AB|| \leq ||A|| \cdot ||B||$.

The first three conditions define *any* norm on a vector space. The fourth one is *specific* to matrix norms and it is necessary due to the fact that matrix multiplication is not done component-wise.

The easiest way of obtaining a matrix norm is from a vector one.

**Definition 2.3.** *Let $||\cdot||$ be a norm on $\mathbb{R}^n$. Then*

$$||A|| = \sup_{v\neq 0} \frac{||Av||}{||v||} = \sup_{||v||\leq 1} ||Av|| = \sup_{||v||=1} ||Av|| \tag{2.5}$$

*is the **natural (subordinate, induced)** matrix norm associated with the vector norm $||\cdot||$.*

**Remark 2.4.**
**1**. It can be easily checked that (2.5) satisfies the conditions of Definition 2.2 and is indeed a matrix norm.
**2.** A subordinate matrix norm is just a particular case for the norm of a linear mapping $A : \mathbb{R}^n \to \mathbb{R}^n$.
**3.** For any induced norm,

$$||I|| = 1. \tag{2.6}$$

**Theorem 2.5.** *Let $A \in \mathbb{R}^{n\times n}$. Then*
**a)**

$$\begin{aligned} ||A||_1 &= \sup_{v\neq 0} \frac{||Av||_1}{||v||_1} = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}| \text{ (the \textbf{Minkovski norm})}, \\ ||A||_\infty &= \sup_{v\neq 0} \frac{||Av||_\infty}{||v||_\infty} = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}| \text{ (the \textbf{Chebyshev norm})}, \\ ||A||_2 &= \sup_{v\neq 0} \frac{||Av||_2}{||v||_2} = \sqrt{\rho(A^TA)} \text{ (the \textbf{Euclidean norm})}. \end{aligned} \tag{2.7}$$

**b)** *The mapping $||\cdot||_F : \mathbb{R}^{n\times n} \to \mathbb{R}^n$ given by*

$$||A||_F = \left[\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|^2\right]^{1/2} = \sqrt{tr(A^TA)} \tag{2.8}$$

*is a nonsubordinate ($||I_n||_F = \sqrt{n}$) matrix norm, called the* ***Frobenius norm****.*

Now, to solve the system $Ax = b$, for a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, suppose there exist $T \in \mathbb{R}^{n\times n}$ and $c \in \mathbb{R}^n$, such that $I - T$ is invertible and the solution $x$ of $Ax = b$ is the unique fixed point of the equation

$$x = Tx + c. \tag{2.9}$$

Let $x^*$ be the solution and $x^{(0)}$ be an arbitrary vector (the initial approximation). Then, we use (2.9) to define an iterative method by

$$x^{(k+1)} = Tx^{(k)} + c,\ k \in \mathbb{N}. \tag{2.10}$$

The matrix $T$ should be chosen such that the system $Tx = f$ is "easily solvable" (diagonal, triangular, tridiagonal, etc.)
Regarding the convergence of such methods, we have the following results from Calculus and Linear Algebra:

**Lemma 2.6** (Geometric Series)**.** *Let $X \in \mathbb{R}^{n\times n}$. If $\rho(X) < 1$, then $(I - X)^{-1}$ exists and*

$$(I - X)^{-1} = I + X + \cdots + X^k + \ldots . \tag{2.11}$$

*Conversely, if the series in* (2.11) *is convergent, then* $\rho(X) < 1$.

**Theorem 2.7.** *The following are equivalent:*
**a)** *The iteration method* (2.10) *is convergent;*
**b)** $\rho(T) < 1$*;*
**c)** $||T|| < 1$ *for some matrix norm* $||\cdot||$.

**Theorem 2.8.** *If* $||T|| < 1$ *for some matrix norm* $||\cdot||$*, then the sequence* $\{x^{(k)}\}_{k\in\mathbb{N}}$ *defined in* (2.10) *converges to the unique fixed point* $x^*$*, starting with any* $x^{(0)} \in \mathbb{R}^n$*, and the error bounds*

$$||x^* - x^{(k)}|| \;\leq\; \frac{||T||}{1-||T||}||x^{(k)} - x^{(k-1)}|| \tag{2.12}$$

*and*

$$||x^* - x^{(k)}|| \;\leq\; \frac{||T||^k}{1-||T||}||x^{(1)} - x^{(0)}|| \;\leq\; \frac{||T||}{1-||T||}||x^{(1)} - x^{(0)}||, \tag{2.13}$$

*hold for every* $k \in \mathbb{N}^*$.

**Remark 2.9.**
**1.** By Theorem 2.8, for a given error $\varepsilon$, we compute iterations until

$$||x^{(k)} - x^{(k-1)}|| \;\leq\; \frac{1-||T||}{||T||}\,\varepsilon. \tag{2.14}$$

**2.** In particular, if $||T|| < 1/2$, then

$$||x^* - x^{(k)}|| \;\leq\; ||x^{(k)} - x^{(k-1)}||$$

and the stopping criterion can be

$$||x^{(k)} - x^{(k-1)}|| \;\leq\; \varepsilon.$$

Now, *how* to actually find the matrix $T$ and the scalar $c$, satisfying (2.9)? Suppose we can write $A$ as

$$A \;=\; M - N. \tag{2.15}$$

This is called a *splitting* of $A$. If $M$ is easily invertible (diagonal, triangular, etc.), then we can write

$$Ax = b \iff Mx = Nx + b \iff x = M^{-1}Nx + M^{-1}b,$$

which is of the form (2.9), with

$$\begin{aligned} T &= M^{-1}N = M^{-1}(M - A) = I - M^{-1}A, \\ c &= M^{-1}b. \end{aligned}$$

We then define the iteration method by

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b,\ k \in \mathbb{N}, \tag{2.16}$$

with $x^{(0)}$ an arbitrary vector.

Assume $A$ is nonsingular, with $a_{ii} \neq 0, i = \overline{1, n}$. We can write

$$A = D - L - U,$$

with

$$D = \begin{bmatrix} a_{11} & & & 0 \\ & a_{22} & & \\ & & \ddots & \\ 0 & & & a_{nn} \end{bmatrix},\ -L = \begin{bmatrix} 0 & & & 0 \\ a_{21} & 0 & & \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \dots & 0 \end{bmatrix},\ -U = \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ & 0 & \dots & a_{2n} \\ & & \ddots & \vdots \\ 0 & & & 0 \end{bmatrix},$$

the diagonal, the lower triangular (without the diagonal) and the upper triangular (without the diagonal) parts of $A$.

For **Jacobi iteration**, take

$$\begin{aligned} M &= D,\ N = L + U, \text{ so} \\ T_J &= D^{-1}(L + U),\ c_J = D^{-1}b. \end{aligned}$$

The method is defined by

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b,\ k \in \mathbb{N},\ x^{(0)} \in \mathbb{R}^n, \tag{2.17}$$

or, component-wise,

$$x_i^{(k+1)} \;=\; \frac{1}{a_{ii}}\Big[b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j^{(k)}\Big],\ i=\overline{1,n}. \tag{2.18}$$

What can be said about the convergence of the method? By Theorem 2.7, we need a matrix norm such that $||T_J|| < 1$. Using Theorem 2.5, we want

$$||T_J||_\infty \;=\; \max_{1\le i\le n}\sum_{j=1}^{n}\left|\frac{a_{ij}}{a_{ii}}\right| \;<\; 1,$$

which means

$$|a_{ii}| \;>\; \sum_{\substack{j=1\\ j\neq i}}^{n}|a_{ij}|,\ i=\overline{1,n},$$

so, a *diagonally dominant* matrix $A$. Thus, for any diagonally dominant system, the Jacobi iterative method converges and the error estimates from Theorem 2.8 can be used. More generally, a necessary and sufficient condition for the convergence of the Jacobi iteration is

$$\rho(T_J) \;<\; 1.$$

For **Gauss-Seidel iteration**, we take

$$\begin{aligned} M &= D-L,\ N \;=\; U, \text{ so}\\ T_{GS} &= (D-L)^{-1}U,\ c_{GS} \;=\; (D-L)^{-1}b. \end{aligned}$$

Then the method is defined by

$$x^{(k+1)} \;=\; (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b,\ k\in\mathbb{N},\ x^{(0)}\in\mathbb{R}^n, \tag{2.19}$$

and each component, by

$$x_i^{(k+1)} \;=\; \frac{1}{a_{ii}}\Big[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\Big],\ i=\overline{1,n}. \tag{2.20}$$

Although it is not so trivial, it can be shown that for a diagonally dominant matrix, $||T_{GS}|| < 1$ and so the Gauss-Seidel iterative method converges at least as fast as the Jacobi one.

### Acceleration methods; SOR Method

Most iterative methods have a regular pattern in which the error decreases. This can often be used to *accelerate* the convergence. Rather than giving a general theory for the acceleration of iteration methods for solving $Ax = b$, we just describe an acceleration of the Gauss-Seidel method. This is one of the main cases of interest in applications.
We introduce an *acceleration parameter* $\omega$ and consider the following modification of the method:

$$\begin{aligned} M &= \frac{D}{\omega} - L,\ N = \Big(\frac{1-\omega}{\omega}D + U\Big), \text{ so} \\ T_\omega &= \Big(\frac{D}{\omega} - L\Big)^{-1}\Big(\frac{1-\omega}{\omega}D + U\Big),\ c_\omega = \Big(\frac{D}{\omega} - L\Big)^{-1} b. \end{aligned}$$

The acceleration method is defined by

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}}\Big[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\Big] + (1-\omega)x_i^{(k)},\ i = \overline{1,n}. \tag{2.21}$$

This is called the *relaxation method*. We have the following cases:
– $\omega < 1$ is called *subrelaxation*;
– $\omega = 1$ is the Gauss-Seidel method;
– $\omega > 1$ is called *overrelaxation*, the **SOR method**, an abbreviation for *successive overrelaxation*.

It can be shown that, if $a_{ii} \neq 0,\ i = \overline{1,n}$, then $\rho(T_\omega) \geq |\omega - 1|$. Thus, by Theorem 2.7, a necessary condition for the convergence of the SOR method is

$$0 < \omega < 2. \tag{2.22}$$

Also, the following holds:

**Theorem 2.10 (Ostrowski-Reich).** *If $A$ is a positive definite matrix and $0 < \omega < 2$, then the SOR iteration method converges for any choice of the initial approximation $x^{(0)} \in \mathbb{R}^n$.*

The parameter $\omega$ is to be chosen to minimize the error, in order to make $x^{(k)}$ converge to $x$ as rapidly as possible. It was found that the optimal value for $\omega$ is

$$\omega^* = \frac{2}{1 + \sqrt{1 - \big(\rho(T_J)\big)^2}}. \tag{2.23}$$

**Remark 2.11.** Iterative methods are rarely used for systems of small order, because they are inefficient, since the time needed to get the desired precision exceeds the time required for Gaussian elimination. But for large systems ($n \geq 10^3$), especially for sparse matrices, they can really make a huge difference in the implementation and computational cost.

## 3 Condition of a Linear System

A number of mathematical problems have solutions that are quite sensitive to small computational errors, for example rounding errors. To deal with this phenomenon, we use the concepts of *stability* and *condition number*.

Consider a general problem of the type

$$y = f(x),\ f : \mathbb{R}^m \to \mathbb{R}^n.$$

We are interested in how "sensitive" is $f$ to small perturbations of $x$, that is, do small perturbations of $x$ lead to small perturbations in $y$ – which means the problem is *stable*, or not – *unstable* problem. In the latter case, approximation is not very helpful. We would like to "measure" the degree of sensitivity (how stable or unstable a problem is) by a single number, called the **condition number** of $f$ at $x$.

The function $f$ is assumed to be evaluated exactly, with infinite precision, as we perturb $x$. The condition number of $f$, therefore, is an inherent property of the map $f$ and does not depend on any algorithmic considerations concerning its implementation.

Recall from the propagation of error formula (in Lecture 1) that when $f : \mathbb{R}^m \to \mathbb{R}$, we found

$$\delta f \approx \sum_{i=1}^{m} x_i \frac{f'_{x_i}}{f} \delta x_i.$$

Now, for the general case $f : \mathbb{R}^m \to \mathbb{R}^n, x = [x_1, \dots, x_m]^T \in \mathbb{R}^m, f(x) = (f_1(x), \dots, f_n(x))^T \in \mathbb{R}^n$, let

$$\gamma_{ij} = (\mathrm{cond}_{ij} f)(x) = \frac{x_i \dfrac{\partial f_j}{\partial x_i}}{f_j(x)},\ i = \overline{1, m},\ j = \overline{1, n}$$

and

$$\Gamma(x) \;=\; \begin{bmatrix}\gamma_{ij}\end{bmatrix} \;=\; \begin{bmatrix} \dfrac{x_1\dfrac{\partial f_1}{\partial x_1}}{f_1(x)} & \cdots & \dfrac{x_m\dfrac{\partial f_1}{\partial x_m}}{f_1(x)} \\ \vdots & & \vdots \\ \dfrac{x_1\dfrac{\partial f_n}{\partial x_1}}{f_n(x)} & \cdots & \dfrac{x_m\dfrac{\partial f_n}{\partial x_m}}{f_n(x)} \end{bmatrix}, \tag{3.1}$$

called the **conditioning matrix**. Then, the **condition number** of $f$ at $x$ is defined by

$$(\text{cond } f)(x) \;=\; ||\Gamma(x)||, \tag{3.2}$$

for a matrix norm $||\cdot||$. If $f$ is a linear function, then

$$(\text{cond } f)(x) \;=\; \frac{||x||\left|\left|\dfrac{\partial f}{\partial x}\right|\right|}{||f(x)||}. \tag{3.3}$$

If the condition number of a problem is large ($(\text{cond } f)(x) \gg 1$), then even for small (relative) errors in the input data, we can expect large errors in the output data. Such problems are called *ill-conditioned* problems. There is no clear line between ill- and well-conditioned problems, it all depends on precision specifications.

Now, for a linear system, we have $A \in \mathbb{R}^{n\times n}$, nonsingular and $b \in \mathbb{R}^n$ given. The problem is finding $x \in \mathbb{R}^n$ such that

$$Ax \;=\; b.$$

So, in this case, the input data consists of $A$ and $b$ and the output data is the vector $x$. Then, we can regard this problem as

$$x \;=\; f(b) \;=\; A^{-1}b,\; f:\mathbb{R}^n \to \mathbb{R}^n. \tag{3.4}$$

Since $f$ is linear and $\dfrac{\partial f}{\partial b} \;=\; A^{-1}$, the condition number is

$$(\text{cond } f)(b) \;=\; \frac{||b||\;||A^{-1}||}{||A^{-1}b||} \;=\; \frac{||Ax||\;||A^{-1}||}{||x||} \;=\; ||A^{-1}||\frac{||Ax||}{||x||}.$$

Then

$$\max_{\substack{b\in\mathbb{R}^n\\ b\neq 0}} (\mathrm{cond} f)(b) \;=\; ||A^{-1}|| \max_{\substack{x\in\mathbb{R}^n\\ x\neq 0}} \frac{||Ax||}{||x||} \;=\; ||A^{-1}||\,||A||.$$

This is the **conditioning number of the matrix** $A$ (and of the system):

$$\mathrm{cond}(A) \;=\; ||A||\,||A^{-1}||. \tag{3.5}$$

If the matrix $A$ is singular, by convention, $\mathrm{cond}(A) = \infty$.

The number $\mathrm{cond}(A)$ will vary with the norm being used, but it is always bounded below by one, since

$$1 \;=\; ||I|| \;=\; ||AA^{-1}|| \;\leq\; ||A||\,||A^{-1}|| \;=\; \mathrm{cond}(A).$$

If the condition number is nearly $1$, then small relative perturbations in $b$ will lead to similarly small relative perturbations in the solution $x$. But if $\mathrm{cond}(A)$ is large, then there may be small relative perturbations of $b$ that will lead to large relative perturbations in $x$.

**Example 3.1. (Ill-conditioned Matrices)**
**1. Hilbert Matrix**

$$H_n \;=\; \begin{bmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \vdots & & & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \cdots & \frac{1}{2n-1} \end{bmatrix}. \tag{3.6}$$

This is a symmetric positive definite matrix, so it is nonsingular. However, it is very ill-conditioned, and increasingly so as $n$ increases.

| $n$ | $\mathrm{cond}_2(H_n)$ |
|---|---|
| 10 | $1.6e+13$ |
| 20 | $2.45e+28$ |
| 40 | $7.65e+58$ |

Table 3: Condition numbers of Hilbert matrix

**2. Vandermonde Matrix**

$$V_n = \begin{bmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_n \\ \vdots & \vdots & \ddots & \vdots \\ t_1^{n-1} & t_2^{n-1} & \dots & t_n^{n-1} \end{bmatrix}. \tag{3.7}$$

For $t_i = \frac{1}{i}, i = \overline{1, n}$, it can be shown that

$$\text{cond}_\infty(V_n) > n^{n+1}.$$

# Chapter 4. Numerical Solution of Nonlinear Equations

## 1 Introduction to Iterative Methods

Finding one or more roots of an equation

$$f(x) \quad = \quad 0, \tag{1.1}$$

is one of the most commonly occurring problems of Applied Mathematics. Even the simplest of nonlinear equations – e.g., algebraic equations – are known to not admit solutions that are expressible rationally in terms of the data. It is therefore impossible, in general, to compute roots of nonlinear equations in a finite numbers of arithmetic operations.

The function $f : \mathbb{R}^m \to \mathbb{R}^n$ is a nonlinear function, which will be assumed to have a certain degree of smoothness. If $n > 1$, then (1.1) represents a system of $n$ equations (at least one nonlinear) with $m$ unknowns.

For now, we will restrict our discussion to the case $m = n = 1$, although many of the procedures we describe can easily be generalized to the multidimensional case.

**Definition 1.1.** *A number $\alpha \in \mathbb{C}$ satisfying equation* (1.1) *is called a* ***zero*** *or a* ***root*** *of $f$.*

As mentioned before, in most cases, explicit solutions of equation (1.1) are not available and we must try to find a root to any specified degree of accuracy. The numerical methods for finding the roots will be *iterative methods* and will require the knowledge of one (or more) initial value(s) $x_0$ $(x_1, \ldots)$. Then the method will produce a sequence $\{x_n\}_{n\in\mathbb{N}}$ of approximations of $\alpha$, such that $\lim\limits_{n\to\infty} x_n = \alpha$. These initial values will be determined, in general, from the context of the problem or from the graph of the function.
The analysis of an iterative method will include
– the proof of convergence, $x_n \to \alpha$, as $n \to \infty$;
– finding the *interval of convergence*, i.e. the set of values of the initial guess(es) $x_0$ $(x_1, \ldots)$ for which the method converges;
– determining the *speed* of convergence.

What makes an iterative method better than another is *how fast* it converges to the desired solution. Regarding the speed of convergence, we define the following:

**Definition 1.2.** *We say that a sequence of iterates $\{x_n\}_{n\in\mathbb{N}}$ converges to $\alpha$ with* ***order of convergence*** $p \geq 1$*, if*

$$|x_{n+1} - \alpha| \leq c\,|x_n - \alpha|^p, \text{ for all } n \in \mathbb{N}, \tag{1.2}$$

*where $c > 0$ is a constant independent of $n$.*
*If $p = 1$, the method is said to* ***converge linearly*** *to $\alpha$, in which case we also require that $c < 1$. Then the constant $c$ is called the* ***rate of linear convergence*** *of $x_n$ to $\alpha$.*
*For $1 < p < 2$, we say that the convergence is* ***superlinear****.*

**Remark 1.3.** If $p = 1$, then

$$\begin{aligned} |x_n - \alpha| &\leq c\,|x_{n-1} - \alpha| \leq \dots \\ &\leq c^n\,|x_0 - \alpha|, \end{aligned}$$

which is why we require that $c < 1$.

## 2 Common Rootfinding Methods

We start with three simple methods and then give a general theory for one-point iteration methods. We recall some known results from Analysis, that will be used in the sequel.

**Theorem 2.1. [Intermediate Value Theorem]**
*If $f : [a, b] \to \mathbb{R}$ is a continuous function, then it takes on any given value between $f(a)$ and $f(b)$ at some point within the interval. As a consequence, if a continuous function has values of opposite sign inside an interval $[a, b]$, then it has at least a root in that interval.*

**Theorem 2.2. [Rolle's Theorem]**
*If a function $f$ is continuous on $[a, b]$ and differentiable on $(a, b)$, with $f(a) = f(b)$, then there exists a point $c \in (a, b)$ such that $f'(c) = 0$. As a consequence, between any two distinct real roots of $f$, there is a root of the derivative.*

So, combining the two, we can find the number of real zeros of a function (satisfying the conditions above) and locate them, by counting the number of *sign changes* of the function at the roots of the derivative and endpoints of the domain of definition.

## 2.1 Bisection Method

Assume that $f : \mathbb{R} \to \mathbb{R}$ is continuous on an interval $[a, b] \subset \mathbb{R}$ and that

$$f(a)f(b) \;<\; 0. \tag{2.1}$$

Then, by the Intermediate Value Theorem, there exists $\alpha \in (a, b)$ such that $f(\alpha) = 0$.
The simplest numerical procedure for finding a root is to repeatedly *halve (bisect)* the interval $[a, b]$, keeping the half on which $f(x)$ changes sign. This procedure is called the **bisection method**. Denoting by $[a_1, b_1] = [a, b]$, the method will produce a sequence of embedded intervals $[a_n, b_n]$, such that for every $n \in \mathbb{N}, \alpha \in [a_n, b_n], f(a_n)f(b_n) < 0$, and a sequence of approximations

$$c_n \;=\; \frac{a_n + b_n}{2} \tag{2.2}$$

of the root $\alpha$ (see Figure 1).

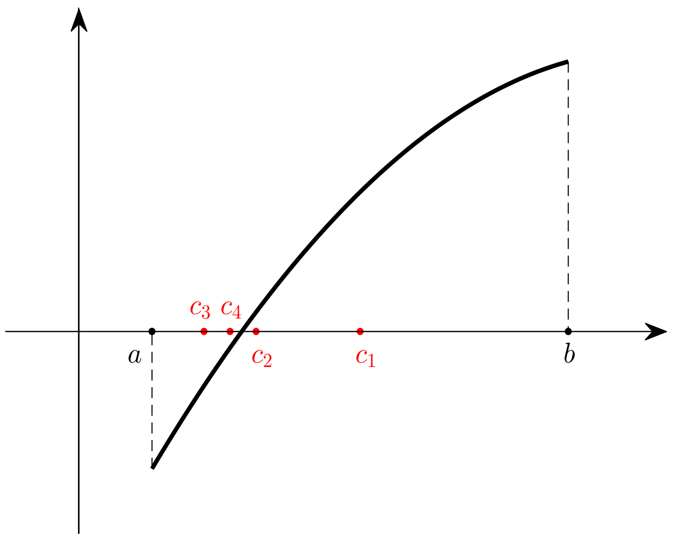

Fig. 1: Bisection method

Usually $[a, b]$ is chosen to contain only one root $\alpha$, but the following algorithm for the bisection method will always converge to some root $\alpha \in [a, b]$, because of (2.1).

**Algorithm 2.3.** [Bisection method]
function $\alpha \;=\; \text{Bisect}(f, a, b, \varepsilon)$

**1.** Define $c = (a+b)/2$.
**2.** If $b - c \leq \varepsilon$, then $\alpha = c$ and exit.
**3.** If $\text{sign}(f(b)) \cdot \text{sign}(f(c)) \leq 0$, then $a = c$; otherwise, $b = c$.
**4.** Return to step 1.

The sequence $\{a_n\}_{n\in\mathbb{N}}$ is monotonely increasing, sequence $\{b_n\}_{n\in\mathbb{N}}$ is monotonely decreasing and

$$\lim_{n\to\infty} a_n \;=\; \lim_{n\to\infty} b_n \;=\; \lim_{n\to\infty} c_n \;=\; \alpha.$$

Also, we have

$$\begin{aligned} |x_n - \alpha| &\leq b_n - a_n = \frac{b-a}{2^n}, \qquad (2.3)\\ |x_{n+1} - \alpha| &\leq \frac{1}{2}|x_n - \alpha|, \end{aligned}$$

which shows that the bisection method converges *linearly* (order of convergence $p = 1$) with a rate of convergence of $\frac{1}{2}$.

**Example 2.4.** Find the largest root of

$$f(x) \;\equiv\; x^6 - x - 1 \;=\; 0, \qquad (2.4)$$

with an error of $\varepsilon = 0.001$.

**Solution.** First, let us see how many reals roots are there and where they are (approximately) located. We have

$$\begin{aligned} f(x) &= x^6 - x - 1,\\ f'(x) &= 6x^5 - 1,\\ f''(x) &= 30x^4. \end{aligned}$$

The second derivative has only one real root, $0$, so the table of variation for $f'$ is

| $x$ | $-\infty$ | $0$ | $\infty$ |
|---|---|---|---|
| $f'$ | $-$ | $-$ | $+$ |

.

This shows that $f'$ has only one real root, which is positive. That real root is $\alpha' = \dfrac{1}{\sqrt[5]{6}}$. Now,

$$f(\alpha') \;=\; \frac{1}{6}\alpha' - \alpha' - 1 \;=\; -\frac{5}{6}\alpha' - 1 \;<\; 0,$$

so for $f$ we have

$$\begin{array}{c|ccc} x & -\infty & \alpha' & \infty \\ \hline f & + & - & + \end{array}.$$

Thus, $f$ has two real roots, one in $(-\infty, \alpha')$ and one, $\alpha \in (\alpha', \infty)$ (which we will approximate). In fact, since

$$\begin{aligned} f(-1) &= 1,\; f(0) \;=\; -1 \;\text{ and} \\ f(1) &= -1,\; f(2) \;=\; 61, \end{aligned}$$

we have a more precise location: a negative root between $(-1, 0)$ and the positive root that we seek, $\alpha \in (1, 2)$. That also gives us the starting interval for the bisection method. Alternatively, we can see from the graph the approximate location of the two real roots (see Figure 2).

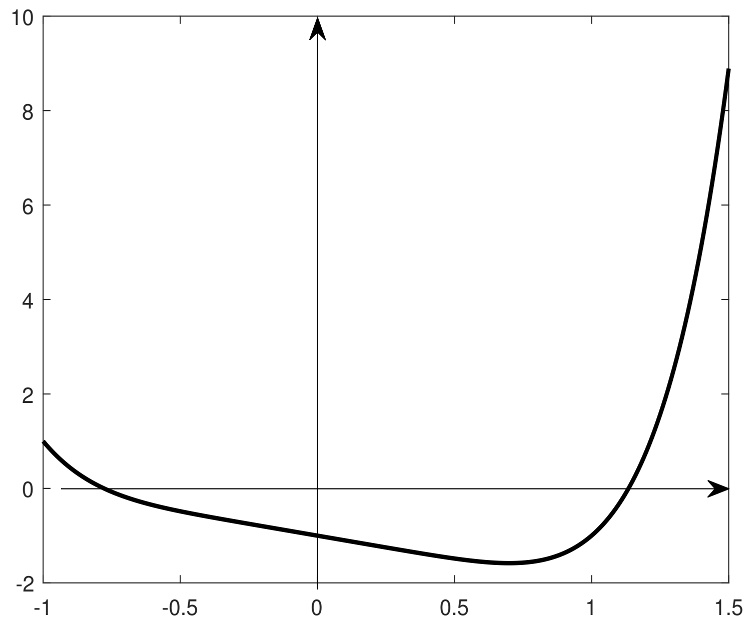

Fig. 2: Function $f(x) = x^6 - x - 1$

So, we start with the interval $[a_1, b_1] \;=\; [1, 2]$. How many iterations are needed for precision

$\varepsilon = 0.001$? We find $n$ from (2.3):

$$\begin{aligned} \frac{b-a}{2^n} &\leq \varepsilon, \text{ which means} \\ n &\geq \log_2\left(\frac{b-a}{\varepsilon}\right), \text{ i.e., in our example,} \\ n &\geq \log_2\left(\frac{1}{10^{-3}}\right) = 9.9658. \end{aligned}$$

| $n$ | $a$ | $b$ | $c$ | $b-c$ | $f(c)$ |
|---|---|---|---|---|---|
| 1 | 1.0000 | 2.0000 | 1.5000 | 0.5000 | 8.8906 |
| 2 | 1.0000 | 1.5000 | 1.2500 | 0.2500 | 1.5647 |
| 3 | 1.0000 | 1.2500 | 1.1250 | 0.1250 | −0.0977 |
| 4 | 1.1250 | 1.2500 | 1.1875 | 0.0625 | 0.6167 |
| 5 | 1.1250 | 1.1875 | 1.1562 | 0.0312 | 0.2333 |
| 6 | 1.1250 | 1.1562 | 1.1406 | 0.0156 | 0.0616 |
| 7 | 1.1250 | 1.1406 | 1.1328 | 0.0078 | −0.0196 |
| 8 | 1.1328 | 1.1406 | 1.1367 | 0.0039 | 0.0206 |
| 9 | 1.1328 | 1.1367 | 1.1348 | 0.0020 | 0.0004 |
| 10 | 1.1328 | 1.1348 | 1.1338 | 0.00098 | −0.0096 |

Table 1: Bisection Method for $x^6 - x - 1 = 0$

The results of the bisection method are shown in Table 1. Indeed, after $n = 10$ iterations, we obtain the desired precision.

■

**Remark 2.5.** The bisection method is a *two-point method*, since two approximate values are needed to obtain an improved value. There are several advantages to the bisection method. The principal one is that the method is guaranteed to converge, as long as the function $f$ is continuous and (2.1) is satisfied. In addition, the error bound given in (2.3) is guaranteed to decrease by one half with each iteration. This relation can also be used as a stopping criterion, as was done in the previous example. The principal disadvantage of the bisection method is that it generally converges slowly (only linearly), more slowly than most other methods. Also, it only approximates real roots.

The next two methods follow the same idea: approximate $f$ by a linear interpolation polynomial and find the root of that polynomial. In other words, the graph of $y = f(x)$ is approximated by a straight line and the $x$-intercept of that line is approximating the root of $f$.

## 2.2 Secant Method

Assume that two initial guesses to $\alpha$ are known and denote them by $x_0$ and $x_1$. We approximate $f$ by its Lagrange polynomial at the nodes $x_0$ and $x_1$. So the graph of $y = f(x)$ is approximated by the *secant* line determined by the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$. The root $\alpha$ of $f$ is then approximated by $x_2$, the $x$-intercept of the secant line. We hope $x_2$ will be an improved approximation of $\alpha$. This is illustrated in Figure 3.

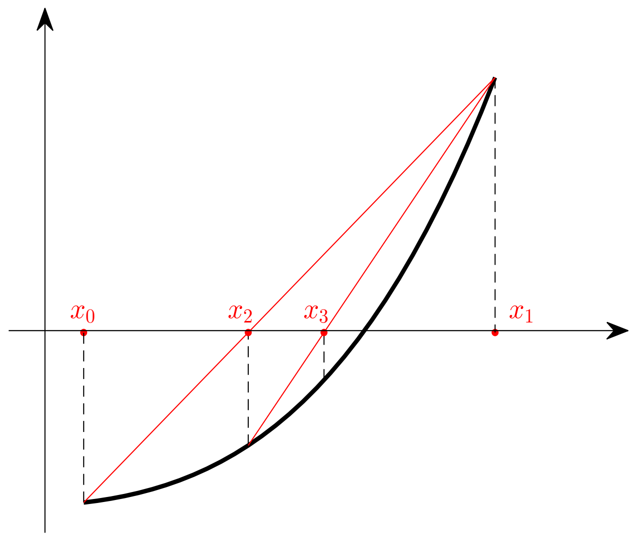

Fig. 3: Secant method

Let us find the value of $x_2$. The equation of the secant line is

$$y - f(x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1).$$

We find its point of intersection with the $x$-axis by letting $y = 0$ and solving for $x$. We get

$$x_2 = x_1 - f(x_1)\,\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

Having found $x_2$, we use $x_1$ and $x_2$ as a new set of approximate values for $\alpha$. This leads to an improved value $x_3$. Recursively, we obtain a sequence of iterates given by

$$x_{n+1} = x_n - f(x_n)\,\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})},\ n = 1, 2, \ldots, \tag{2.5}$$

called the **secant method**.

**Example 2.6.** We solve again the equation

$$f(x) \;\equiv\; x^6 - x - 1 \;=\; 0,$$

which was used previously as an example for the bisection method.

**Solution.** We start with

$$x_0 \;= 1,\;\; x_1 \;=\; 2.$$

The results are given in Table 2, including the quantities $x_n - x_{n-1}$ as an estimate of $\alpha - x_{n-1}$. The iterate $x_8$ equals $\alpha$ rounded to nine significant digits.

| $n$ | $x_n$ | $f(x_n)$ | $x_n - x_{n-1}$ | $\alpha - x_{n-1}$ |
|---|---|---|---|---|
| 0 | 2.0 | 61.0 | | |
| 1 | 1.0 | $-1.0$ | $-1.0$ | |
| 2 | 1.01612903 | $-9.15e-1$ | $1.61e-2$ | $1.35e-1$ |
| 3 | 1.19057777 | $6.57e-1$ | $1.74e-1$ | $1.19e-1$ |
| 4 | 1.11765583 | $-1.68e-1$ | $-7.29e-2$ | $-5.59e-2$ |
| 5 | 1.13253155 | $-2.24e-2$ | $1.49e-2$ | $1.71e-2$ |
| 6 | 1.13481681 | $9.54e-4$ | $2.29e-3$ | $2.19e-3$ |
| 7 | 1.13472365 | $-5.07e-6$ | $-9.32e-5$ | $-9.27e-5$ |
| 8 | 1.13472414 | $-1.13e-9$ | $4.92e-7$ | $4.92e-7$ |

Table 2: Secant Method for $x^6 - x - 1 = 0$

■

The secant method is also a two-point iterative method. Unlike the bisection method, it *does not* always converge. For a convergence and error analysis, let us compute, from (2.5),

$$\begin{aligned} x_{n+1} - \alpha &= x_n - \alpha - f(x_n)\,\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \\ &= x_n - \alpha - \frac{f(x_n)}{\dfrac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}} \\ &= x_n - \alpha - \frac{f(x_n) - f(\alpha)}{\dfrac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}}, \quad \text{since } \; f(\alpha) = 0. \end{aligned}$$

Further, we make use of divided differences and obtain

$$\begin{aligned} x_{n+1} - \alpha &= x_n - \alpha - (x_n - \alpha)\,\frac{f[x_n, \alpha]}{f[x_{n-1}, x_n]} \\ &= (x_n - \alpha)\left[1 - \frac{f[x_n, \alpha]}{f[x_{n-1}, x_n]}\right] \\ &= (x_n - \alpha)\,\frac{f[x_{n-1}, x_n] - f[x_n, \alpha]}{f[x_{n-1}, x_n]} \\ &= (x_n - \alpha)(x_{n-1} - \alpha)\,\frac{f[x_{n-1}, x_n, \alpha]}{f[x_{n-1}, x_n]}, \end{aligned}$$

so,

$$x_{n+1} - \alpha = (x_n - \alpha)(x_{n-1} - \alpha)\,\frac{f''(\xi_n)}{2f'(\zeta_n)}, \tag{2.6}$$

with $\zeta_n$ between $x_n$ and $x_{n-1}$, and $\xi_n$ between the smallest and the largest of the numbers $\alpha, x_n$ and $x_{n-1}$. Using (2.6) and a limiting argument, we have the following convergence result.

**Theorem 2.7.** *Assume $f$, $f'$ and $f''$ are continuous on an interval $I_\varepsilon = (\alpha - \varepsilon, \alpha + \varepsilon)$ containing the simple root $\alpha$ ($f'(\alpha) \neq 0$). Then, for starting values $x_0$ and $x_1$ sufficiently close to $\alpha$, the iterates in* (2.5) *converge to $\alpha$, with order of convergence*

$$p = r = \frac{1 + \sqrt{5}}{2} \approx 1.618033\ldots, \tag{2.7}$$

*the "golden ratio".*

Thus, the secant method converges *superlinearly*.

**Remark 2.8.**
**1.** To understand what "sufficiently close" means in the theorem above, let

$$M_\varepsilon = \frac{\max\limits_{I_\varepsilon} |f''(x)|}{2\min\limits_{I_\varepsilon} |f'(x)|}, \quad e_0 = |x_0 - \alpha|, \;\; e_1 = |x_1 - \alpha|. \tag{2.8}$$

Then the method above will converge if $x_0, x_1 \in I_\varepsilon$, with $\varepsilon > 0$ chosen so that

$$\max\{M_\varepsilon e_0, M_\varepsilon e_1\} < 1. \tag{2.9}$$

**2.** It is clear now that the secant method does not always converge, but when it does, it does so *faster* than the bisection method (its order of convergence is higher). That was obvious in our example.

**3.** Another advantage is that the secant method can be used to approximate complex roots, as well, if the initial values $x_0$ and $x_1$ are taken to be complex numbers satisfying the conditions above.
**4.** It can be shown that

$$\begin{aligned} \lim_{n\to\infty} \frac{|x_{n+1}-x_n|}{|x_n-\alpha|} &= 1 \text{ and, thus,} \\ |x_n-\alpha| &\approx |x_{n+1}-x_n|, \text{ for sufficiently large } n, \end{aligned} \tag{2.10}$$

which can be used as a stopping criterion.

## 2.3 Newton's Method

In a similar fashion, now we start with one initial value, $x_0$ and approximate $f$ by its linear Taylor polynomial at the double node $x_0$. In other words, the graph of $y = f(x)$ is approximated by the line *tangent* to the graph of $f$ at the point $(x_0, f(x_0))$. The root $\alpha$ of $f$ is then approximated by $x_1$, the point of intersection of the tangent line with the $x$-axis. If $x_0$ is close enough to $\alpha$, then the root of the Taylor polynomial should be close to $\alpha$.

The tangent line at $x_0$ has equation

$$y - f(x_0) = f'(x_0)(x - x_0),$$

so, for $x_1$, we find

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Repeat the process to further improve the estimate of $\alpha$. Recursively, we get

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ n = 0, 1, \ldots. \tag{2.11}$$

This is called **Newton's (tangent) method** and it is illustrated in Figure 4.

**Example 2.9.** Let us approximate the positive solution of

$$f(x) \equiv x^6 - x - 1 = 0,$$

using Newton's method.

**Solution.** An initial guess $x_0$ can be taken from the graph of $y = f(x)$ in Figure 2. The iterative

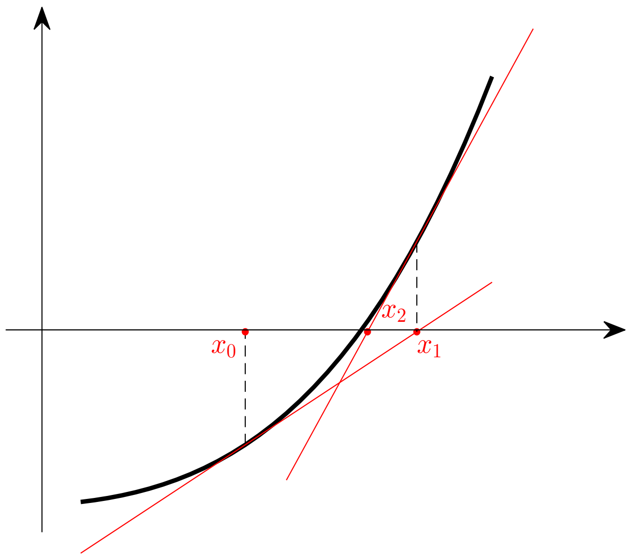

Fig. 4: Newton's method

method is given by

$$x_{n+1} = x_n - \frac{x_n^6 - x_n - 1}{6x_n^5 - 1}, \; n \geq 0.$$

Table 3 shows the results of Newton's method with initial value $x_0 = 1.5$.

| $n$ | $x_n$ | $f(x_n)$ | $x_n - x_{n-1}$ | $\alpha - x_{n-1}$ |
|---|---|---|---|---|
| 0 | 1.5 | $8.89e+1$ | | |
| 1 | 1.30049088 | $2.54e+1$ | $-2.00e-1$ | $-3.65e-1$ |
| 2 | 1.18148042 | $5.38e-1$ | $-1.19e-1$ | $-1.66e-1$ |
| 3 | 1.13945559 | $4.92e-2$ | $-4.20e-2$ | $-4.68e-2$ |
| 4 | 1.13477763 | $5.50e-4$ | $-4.68e-3$ | $-4.73e-3$ |
| 5 | 1.13472415 | $7.11e-8$ | $-5.35e-5$ | $-5.35e-5$ |
| 6 | 1.13472414 | $1.55e-15$ | $-6.91e-9$ | $-6.91e-9$ |

Table 3: Newton's Method for $x^6 - x - 1 = 0$

As seen from the table, the convergence is very rapid. The iterate $x_6$ is accurate to the machine precision of around 16 decimal digits.

■

As before, we can compute

$$\begin{aligned} x_{n+1} - \alpha &= (x_n - \alpha)^2 \, \frac{f[x_n, x_n, \alpha]}{f[x_n, x_n]} \\ &= (x_n - \alpha)^2 \, \frac{f''(\xi_n)}{2f'(x_n)}, \end{aligned} \tag{2.12}$$

with $\xi_n$ between $\alpha$ and $x_n$. Then we have the following convergence result.

**Theorem 2.10.** *Assume $f$, $f'$ and $f''$ are continuous on an interval $I_\varepsilon = (\alpha - \varepsilon, \alpha + \varepsilon)$ containing the simple root $\alpha$ ($f'(\alpha) \neq 0$). Then, if the initial value $x_0$ is sufficiently close to $\alpha$, the iterates in* (2.11) *converge to $\alpha$ and*

$$\lim_{n\to\infty} \frac{x_{n+1} - \alpha}{(x_n - \alpha)^2} = \frac{f''(\alpha)}{2f'(\alpha)}, \tag{2.13}$$

*which shows that the order of convergence of Newton's method is $p = 2$.*

**Remark 2.11.**
**1.** Similarly with Remark 2.8, "sufficiently close" means $x_0 \in I_\varepsilon$, where $\varepsilon$ is chosen so that $M_\varepsilon e_0 < 1$, with $M_\varepsilon$ and $e_0$ defined in (2.8).
**2.** Again, as before, Newton's method *does not* always converge, but when it does, it does so faster ($p = 2$) than the bisection method ($p = 1$) and the secant method ($p = (1 + \sqrt{5})/2 \approx 1.618$).
**3.** Also, Newton's method can be used to approximate complex roots, as well, if the initial value $x_0$ is a complex number satisfying the conditions above.
**4.** Again, for sufficiently large $n$,

$$|x_n - \alpha| \;\approx\; |x_{n+1} - x_n|,$$

which can be used as a stopping criterion.
**5.** Unlike the bisection and secant methods, Newton's method is a *one-step* iterative method, as it only requires one initial value. Later on, we will give a more comprehensive analysis of one-step iteration methods.

## 2.4 Comparison Between Newton's and Secant Methods

As we have seen, Newton's method and the secant method are closely related. If the approximation

$$f'(x_n) \;\approx\; \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

is used in Newton's formula (2.11), we obtain the secant formula (2.5).

The conditions for convergence are almost identical and the error formulas are similar. Nonetheless, there are two major differences. Newton's method requires two function evaluations per iterate, those of $f(x_n)$ and $f'(x_n)$, whereas the secant method requires only one function evaluation per iterate, that of $f(x_n)$ (provided that the value of $f(x_{n-1})$ is retained from the last iteration). So, Newton's method is generally more expensive per iteration. On the other hand, it converges more rapidly (order $p = 2$ versus $p = r \approx 1.62$) and consequently, it will require fewer iterations to attain a given desired accuracy. A comparison of the expenditure of computational time needed to approximate a root $\alpha$ within a desired tolerance, can be made.

To simplify the analysis, we assume that the initial guesses are quite close to the desired root, so both methods converge. Let $t$ be the time needed to evaluate $f(x)$, and $s \cdot t$ the time required to evaluate $f'(x)$. By writing the operations involved in the two methods, it can be then shown that the minimum time to obtain the desired accuracy with Newton's method is

$$T_N = \frac{(1+s)tK}{\log 2},$$

while, for the secant method, a similar calculation shows that the minimum time necessary to obtain the desired accuracy is

$$T_S = \frac{tK}{\log r},$$

where $K$ is a positive constant that depends on $\varepsilon, x_0$ and $c = \left|\frac{f''(\alpha)}{2f'(\alpha)}\right|$. Thus,

$$\frac{T_S}{T_N} = \frac{\log 2}{(1+s)\log r}.$$

The secant method is faster than Newton's method if the ratio is less than one, i.e.

$$\begin{aligned} \frac{T_S}{T_N} &< 1 \\ s &> \frac{\log 2}{\log r} - 1 \approx 0.44. \end{aligned}$$

In conclusion, if the time needed to evaluate $f'(x)$ is more than $44\%$ of that necessary to evaluate $f(x)$, then the secant method is more efficient. In practice, many other factors will affect the relative costs of the two methods, so that the $.44$ factor should be used with caution.

# 3 One-Point Iteration Methods – General Theory

## 3.1 Fixed Point Iteration

A classical approach is to reformulate equation $f(x) = 0$ as

$$x \;=\; g(x) \tag{3.1}$$

and find a *fixed point* for $g$. Let us first note that the form (3.1) is *not* restrictive in any way. In fact, any equation can be written as (3.1) in a multitude of ways.

**Example 3.1.** Consider the equation

$$x^2 - 3 \;=\; 0.$$

It can be rewritten, for instance, as

$$\begin{array}{lrcl} \textbf{(a)} & x & = & x^2 + x - 3, \\ \textbf{(b)} & x & = & \dfrac{3}{x}, \\ \textbf{(c)} & x & = & \dfrac{1}{2}\Big(x + \dfrac{3}{x}\Big), \\ \textbf{(d)} & x & = & x + c(x^2 - 3), \text{ for some constant } c \in \mathbb{R}, \end{array}$$

and many other ways.

Now we can employ fixed point theory to discuss the solvability of equation (3.1). In what follows, the notation

$$g\big([a, b]\big) \;\subseteq\; [a, b]$$

means

$$x \in [a, b] \implies g(x) \in [a, b].$$

**Lemma 3.2.** *Let $g \in C[a, b]$, such that $g\big([a, b]\big) \subseteq [a, b]$. Then $g$ has at least one fixed point in $[a, b]$.*

*Proof.* This follows immediately from the Intermediate Value Theorem applied to the function

$$G(x) = g(x) - x.$$

Since $G$ is continuous and $G(a) \;\geq\; 0, \;\; G(b) \;\leq\; 0$, $G$ must have at least one zero in $[a, b]$, which is, obviously, a fixed point of $g$.

□

**Theorem 3.3.** **[Banach]** *Let* $g \in C[a,b]$*, with* $g\big([a,b]\big) \subseteq [a,b]$*. Assume that there exists* $0 < \lambda < 1$ *such that*

$$\big|g(x) - g(y)\big| \;\; \le \;\; \lambda\,|x-y|,\ \forall x, y \in [a,b] \ \text{(i.e., } g \text{ is a } \textbf{contraction}\text{).} \tag{3.2}$$

*Then* $g$ *has a unique fixed point* $\alpha \in [a,b]$*. Furthermore, the iterates*

$$x_{n+1} \;\; = \;\; g(x_n),\ n \ge 0, \tag{3.3}$$

*converge to* $\alpha$*, for any choice of* $x_0 \in [a,b]$ *and the following error estimates hold:*

$$\begin{aligned} |x_n - \alpha| \;\; &\le \;\; \lambda\,|x_{n-1} - \alpha|,\ n \ge 1, \\ |x_n - \alpha| \;\; &\le \;\; \frac{\lambda^n}{1-\lambda}\,|x_1 - x_0|. \end{aligned} \tag{3.4}$$

*Proof.* The existence of the fixed point is guaranteed by Lemma 3.2.
To prove its uniqueness, assume there are two fixed points, $\alpha = g(\alpha),\ \beta = g(\beta),\ \alpha \ne \beta$. Then

$$|\alpha - \beta| \;\; = \;\; \big|g(\alpha) - g(\beta)\big| \overset{(3.2)}{\le} \lambda\,|\alpha - \beta|$$

and, so,

$$(1-\lambda)|\alpha - \beta| \;\; \le 0,$$

which is a contradiction. Thus, $\alpha = \beta$.
To prove the convergence, let us note that

$$x_0 \in [a,b] \Longrightarrow x_1 = g(x_0) \in [a,b] \Longrightarrow \cdots \Longrightarrow x_n \in [a,b],\ \forall n \ge 0.$$

Then,

$$|x_n - \alpha| \;\; = \;\; \big|g(x_{n-1}) - g(\alpha)\big| \;\le\; \lambda\,\big|x_{n-1} - \alpha\big| \;\le\; \ldots \;\le\; \lambda^n\,|x_0 - \alpha|.$$

Letting $n \to \infty$, since $\lambda^n \to 0$, it follows that $x_n \to \alpha$ and the first bound in (3.4) holds.
For the second bound, we write

$$\begin{aligned} |x_0 - \alpha| \;\; &\le \;\; |x_0 - x_1| + |x_1 - \alpha| \;\le\; |x_0 - x_1| + \lambda\,|x_0 - \alpha|, \\ |x_0 - \alpha| \;\; &\le \;\; \frac{1}{1-\lambda}|x_1 - x_0|. \end{aligned}$$

Combining this with the previous relation, we get the second bound in (3.4). □

**Remark 3.4.**

**1.** The first bound shows that $\{x_n\}_{n\in\mathbb{N}}$ converges *linearly*, with a rate of convergence bounded by the contraction constant $\lambda$.

**2.** From the proof of Theorem 3.3, we can also show that

$$\begin{aligned} |x_n-\alpha| &\leq \frac{1}{1-\lambda}|x_{n+1}-x_n| \text{ and, hence,} \\ |x_{n+1}-\alpha| &\leq \lambda\,|x_n-\alpha| \leq \frac{\lambda}{1-\lambda}|x_{n+1}-x_n|, \end{aligned}$$

which gives a stopping criterion

$$|x_{n+1}-x_n| \leq \frac{1-\lambda}{\lambda}\,\varepsilon. \tag{3.5}$$

**3.** If $g$ is also differentiable on $(a,b)$, then, by the MVT, there exists $c\in(a,b)$ such that

$$g(x)-g(y) = g'(c)(x-y), \forall x,y\in[a,b].$$

Letting $\lambda = \max\limits_{x\in[a,b]}\big|g'(x)\big|$, it follows that

$$\big|g(x)-g(y)\big| \leq \lambda\,|x-y|,\ \forall x,y\in[a,b].$$

Then, we can restate the convergence result.

**Theorem 3.5.** *Let $g\in C^1[a,b]$, such that $g\big([a,b]\big)\subseteq[a,b]$ and*

$$\lambda := \max_{x\in[a,b]}\big|g'(x)\big| < 1. \tag{3.6}$$

*Then:*

***a)*** *Function $g$ has a unique fixed point $\alpha\in[a,b]$.*

***b)*** *For any initial choice $x_0\in[a,b]$, the sequence $x_{n+1} = g(x_n)$ converges to $\alpha$, as $n\to\infty$.*

***c)*** $|x_n-\alpha| \leq \lambda^n|x_0-\alpha| \leq \dfrac{\lambda^n}{1-\lambda}|x_1-x_0|,\ n\geq 1.$

***d)***

$$\lim_{n\to\infty}\frac{x_{n+1}-\alpha}{x_n-\alpha} = g'(\alpha), \tag{3.7}$$

*so, if $g'(\alpha)\neq 0$, the iterative method $x_{n+1} = g(x_n)$ is linearly convergent to the root $\alpha$ with rate of convergence bounded by $\lambda$.*

The conditions of Theorem 3.5 can be relaxed and imposed only *locally*, near the root $\alpha$.

**Theorem 3.6.** *Assume $\alpha$ is a fixed point of $g$ and that $g$ is continuously differentiable in a neighborhood of $\alpha$, with*

$$|g'(\alpha)| \;<\; 1. \tag{3.8}$$

*Then the conclusions of Theorem 3.5 still hold, provided that $x_0$ is chosen sufficiently close to $\alpha$.*

**Example 3.7.** Refer back to the equation $x^2 - 3 \;=\; 0$ in Example 3.1, with $\alpha \;=\; \sqrt{3}$. Let us see which of the four iterative methods are convergent.

**Solution.**
**(a)**

$$g(x) \;=\; x^2 + x - 3,\; g'(x) \;=\; 2x + 1,\; g'(\alpha) \;=\; 2\sqrt{3} + 1 \;>\; 1,$$

so this method *does not converge*.
**(b)**

$$g(x) \;=\; \frac{3}{x},\; g'(x) \;=\; -\frac{3}{x^2},\; g'(\alpha) \;=\; -1,$$

so this method does not converge, either.
**(c)**

$$g(x) \;=\; \frac{1}{2}\Big(x + \frac{3}{x}\Big),\; g'(x) \;=\; \frac{1}{2}\Big(1 - \frac{3}{x^2}\Big),\; g'(\alpha) \;=\; 0.$$

This method *will converge* at least linearly.
**(d)**

$$g(x) \;=\; x + c(x^2 - 3),\; g'(x) \;=\; 1 + 2cx,\; g'(\alpha) \;=\; 1 + 2c\sqrt{3}.$$

For convergence, pick $c$ such that $|g'(\alpha)| \;<\; 1$, i.e., $-\dfrac{1}{\sqrt{3}} \;<\; c \;<\; 0.$

For a good rate of linear convergence, pick $c$ such that $1 + 2c\sqrt{3} \;\approx\; 0$, or $c \;\approx\; -\dfrac{1}{2\sqrt{3}}$, for example, $c \;=\; -\dfrac{1}{4}$.

■

## 3.2 Higher Order One-Point Iteration Methods

Let us recall the main fixed-point iteration results from last time.

**Theorem 3.1.** *Let $g \in C^1[a,b]$, such that $g\big([a,b]\big) \subseteq [a,b]$ and*

$$\lambda \ := \ \max_{x\in[a,b]} \big|g'(x)\big| \ < \ 1. \tag{3.1}$$

*Then:*

***a)*** *Function $g$ has a unique fixed point $\alpha \in [a,b]$.*

***b)*** *For any initial choice $x_0 \in [a,b]$, the sequence $x_{n+1} \ = \ g(x_n)$ converges to $\alpha$, as $n \to \infty$.*

***c)*** $|x_n - \alpha| \ \leq \ \lambda^n |x_0 - \alpha| \ \leq \ \dfrac{\lambda^n}{1-\lambda}|x_1 - x_0|, \ n \geq 1.$

***d)***

$$\lim_{n\to\infty} \frac{x_{n+1}-\alpha}{x_n - \alpha} \ = \ g'(\alpha), \tag{3.2}$$

*so, if $g'(\alpha) \ \neq \ 0$, the iterative method $x_{n+1} \ = \ g(x_n)$ is linearly convergent to the root $\alpha$ with rate of convergence bounded by $\lambda$.*

**Theorem 3.2.** *Assume $\alpha$ is a fixed point of $g$ and that $g$ is continuously differentiable in a neighborhood of $\alpha$, with*

$$|g'(\alpha)| \ < \ 1. \tag{3.3}$$

*Then the conclusions of Theorem 3.1 still hold, provided that $x_0$ is chosen sufficiently close to $\alpha$.*

So far, there isn't much information in the case $g'(\alpha) = 0$, although the convergence is clearly quite good. Moreover, what happens if the derivatives of $g$ of up to some order are all $0$ at $\alpha$? Can we expect a *faster* convergence? The answer is in the following result.

**Theorem 3.3.** *Assume $\alpha$ is a fixed point of $g$ and that $g$ is $p$ times continuously differentiable for all $x$ near $\alpha$, for some $p \geq 2$. Furthermore, assume that*

$$g'(\alpha) \ = \ \ldots \ = \ g^{(p-1)}(\alpha) \ = \ 0, \ \ g^{(p)}(\alpha) \ \neq \ 0. \tag{3.4}$$

*Then, if the initial value $x_0$ is chosen sufficiently close to $\alpha$, the iteration method $x_{n+1} \ = \ g(x_n)$ converges to $\alpha$ with order of convergence $p$, and*

$$\lim_{n\to\infty} \frac{x_{n+1}-\alpha}{(x_n-\alpha)^p} \ = \ \frac{1}{p!} g^{(p)}(\alpha). \tag{3.5}$$

*Proof.* Since $g'(\alpha) = 0$, by Theorem 3.2, it follows that the iterative method $x_{n+1} = g(x_n)$ converges to $\alpha$, if $x_0$ is sufficiently close to $\alpha$.

For the order of convergence, we use the Taylor series expansion of $g$ around $\alpha$:

$$x_{n+1} = g(x_n) = g(\alpha) + (x_n - \alpha)g'(\alpha) + \cdots + \frac{(x_n-\alpha)^{p-1}}{(p-1)!}g^{(p-1)}(\alpha) + \frac{(x_n-\alpha)^p}{p!}g^{(p)}(\xi_n),$$

for some $\xi_n$ between $x_n$ and $\alpha$. Using (3.4) and the fact that $g(\alpha) = \alpha$, we get

$$x_{n+1} - \alpha = \frac{(x_n-\alpha)^p}{p!}g^{(p)}(\xi_n), \quad \frac{x_{n+1}-\alpha}{(x_n-\alpha)^p} = \frac{1}{p!}g^{(p)}(\xi_n).$$

Letting $n \to \infty$, both $x_n,\ \xi_n \to \alpha$ and, hence, (3.5) follows.

□

**Example 3.4.** Recall Newton's iteration method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ n \geq 0. \tag{3.6}$$

Let us analyze it by this new result.

**Solution.** We have

$$g(x) = x - \frac{f(x)}{f'(x)},$$

for a simple root of $f$, $\alpha$, which means $f(\alpha) = 0$ and $f'(\alpha) \neq 0$. We have

$$g'(x) = 1 - \frac{\big(f'(x)\big)^2 - f(x)f''(x)}{\big(f'(x)\big)^2} = \frac{f(x)f''(x)}{\big(f'(x)\big)^2}.$$

We compute the second derivative, but discard the argument $x$:

$$g'' = \frac{\big(f'f'' + ff'''\big)\big(f'\big)^2 - 2f'f'' \cdot ff''}{\big(f'\big)^4}.$$

Then

$$\begin{aligned} g(\alpha) &= \alpha,\ \ g'(\alpha) \ = \ 0, \\ g''(\alpha) &= \frac{\big(f'(\alpha)f''(\alpha)\big)\big(f'(\alpha)\big)^2}{\big(f'(\alpha)\big)^4} \ = \ \frac{f''(\alpha)}{f'(\alpha)} \end{aligned}$$

and Theorem 3.3 gives the previously found quadratic convergence ($p = 2$) and error estimate

$$\lim_{n\to\infty} \frac{x_{n+1}-\alpha}{(x_n-\alpha)^2} \ = \ \frac{1}{2}g''(\alpha) \ = \ \frac{f''(\alpha)}{2f'(\alpha)}.$$

■

**Example 3.5.** Let us revisit the problem from Example 3.6 (Lecture 11): equation $x^2 - 3 = 0$, with $\alpha = \sqrt{3}$.

**Solution.** Last time we saw several ways of rewriting the equation in the form $g(x) = x$, some "better" than others, from the convergence point of view.

Let us use Newton's iteration. We have

$$f(x) \ = \ x^2 - 3,\ \ f'(x) \ = \ 2x,$$

so,

$$g(x) \ = \ x - \frac{f(x)}{f'(x)} = x - \frac{x^2-3}{2x} = \frac{1}{2}\Big(2x - x + \frac{3}{x}\Big) \ = \ \frac{1}{2}\Big(x + \frac{3}{x}\Big),\ \ g(\alpha) \ = \alpha,$$

which was one of the methods discussed (part **(c)**). Further, we have

$$\begin{aligned} g'(x) &= \frac{1}{2}\Big(1 - \frac{3}{x^2}\Big),\ \ g'(\alpha) \ = \ 0, \\ g''(x) &= \frac{3}{x^3},\ \ g''(\alpha) \ = \ \frac{1}{\sqrt{3}} \ \neq \ 0. \end{aligned}$$

So, indeed, the iteration $x_{n+1} = \frac{1}{2}\Big(x_n + \frac{3}{x_n}\Big)$ converges quadratically ($p = 2$) to $\alpha$.
As a side note, rewriting the equation as $x = \frac{1}{2}\Big(x + \frac{3}{x}\Big)$ (which leads to fast convergence) was *not* a "lucky guess", it is actually Newton's method.

■

**Example 3.6.** Consider the equation

$$x = g(x) = cx(1-x), \tag{3.7}$$

with $c \neq 0$. This is called a *logistic equation* and is of great interest in the *mathematical theory of chaos*. This equation has one nonzero solution, denoted by $\alpha_c$. For what values of $c$ will the iteration $x_{n+1} = g(x_n)$ converge to $\alpha_c$ (provided that $x_0$ is chosen sufficiently close to $\alpha_c$)? Determine the convergence order.

**Solution.** The nonzero solution of equation (3.7) is given by

$$c(1-x) = 1, \; x = \alpha_c = 1 - \frac{1}{c} = \frac{c-1}{c}.$$

We have

$$\begin{aligned} g(x) &= c(x - x^2), \; g(\alpha_c) = \alpha_c, \\ g'(x) &= c(1-2x), \; g'(\alpha_c) = c\Big(1 - 2 + \frac{2}{c}\Big) = 2 - c, \\ g''(x) &= -2c. \end{aligned}$$

In order to have the iteration $x_{n+1} = g(x_n)$ converge to $\alpha_c$, we impose the condition

$$|g'(\alpha_c)| < 1 \iff |2-c| < 1 \iff c \in (1,3).$$

So, for any $1 < c < 3$, the method converges (at least) linearly, with rate of convergence $|c-2|$. Now, if $g'(\alpha_c) = 0$, i.e., $c = 2$, then the convergence is quadratic ($p = 2$) and

$$\lim_{n\to\infty} \frac{x_{n+1} - \alpha_c}{(x_n - \alpha_c)^2} = \frac{1}{2} g''(\alpha_c) = \frac{-2c}{2} = -c = -2.$$

■

## 4 Numerical Approximation of Multiple Roots

**Definition 4.1.** *We say that a function $f$ has a* ***root*** *$\alpha$* ***of multiplicity*** *$m > 1$ if*

$$f(x) = (x-\alpha)^m \, h(x), \; h \text{ continuous at } x = \alpha \text{ and } h(\alpha) \neq 0. \tag{4.1}$$

We restrict our discussion to the case where $m$ is a positive integer, although some of our considerations are equally valid for non-integer values. If $h$ is smooth enough at $x = \alpha$, then (4.1) is equivalent to

$$f(\alpha) \;=\; f'(\alpha) \;=\; \ldots \;=\; f^{(m-1)}(\alpha) \;=\; 0, \;\; f^{(m)}(\alpha) \;\neq\; 0. \tag{4.2}$$

There are several challenges in approximating multiple roots.

## Uncertainty

When finding a root of any function on a computer, there is always an *interval of uncertainty* about the root, due to measuring/rounding/truncation errors, and this is made worse when the root is multiple.

**Example 4.2.** Consider evaluating the two functions

$$\begin{aligned} f_1(x) &= x^2 - 3, \\ f_2(x) &= x^2(x^2 - 6) + 9. \end{aligned}$$

Then $\alpha = \sqrt{3}$ has multiplicity 1 as a root of $f_1$ and multiplicity 2 as a root of $f_2$, since

$$f_2'(x) \;=\; 4x(x^2 - 3).$$

Using four-digit decimal arithmetic, we have

$$\begin{aligned} f_1(x) &< 0, \;\; \text{for } x \leq 1.731, \\ f_1(1.732) &= 0, \;\; \text{and} \\ f_1(x) &> 0, \;\; \text{for } x > 1.733, \end{aligned}$$

so $\alpha \in (1.731, 1.733)$. But for $f_2$,

$$f_2(x) \;=\; 0, \;\; \text{for } 1.726 \leq x \leq 1.738,$$

implying that $\alpha \in [1.726, 1.738]$, thus limiting the amount of accuracy that can be attained in finding a root of $f_2$.

## Loss of Precision

Another problem with multiple roots is that the earlier rootfinding methods will not perform as well when the root being sought is multiple. Let us investigate this for Newton's method. We consider Newton's method as a fixed-point iteration

$$x_{n+1} \;=\; g(x_n),\;\; g(x) \;=\; x - \frac{f(x)}{f'(x)},\; x \neq \alpha,\;\; f(x) \;=\; (x-\alpha)^m\, h(x),\; m > 1.$$

We have

$$f'(x) \;=\; m(x-\alpha)^{m-1}\, h(x) + (x-\alpha)^m\, h'(x) \;=\; (x-\alpha)^{m-1}\big[m\, h(x) + (x-\alpha)\, h'(x)\big],$$

so,

$$\begin{aligned} g(x) &= x - \frac{(x-\alpha)^m\, h(x)}{(x-\alpha)^{m-1}\big[m\, h(x) + (x-\alpha)\, h'(x)\big]} \\ &= x - (x-\alpha)\frac{h(x)}{m\, h(x) + (x-\alpha)\, h'(x)} \;=\; x - (x-\alpha)\varphi(x), \end{aligned}$$

where

$$\varphi(x) \;\overset{\text{not}}{=}\; \frac{h(x)}{m\, h(x) + (x-\alpha)\, h'(x)},\;\; \varphi(\alpha) \;=\; \frac{1}{m}.$$

Then,

$$\begin{aligned} g'(x) &= 1 - \varphi(x) - (x-\alpha)\varphi'(x), \\ g'(\alpha) &= 1 - \varphi(\alpha) \;=\; 1 - \frac{1}{m} \;\neq\; 0. \end{aligned}$$

Thus, in this case, Newton's method converges only *linearly*, with rate of convergence $1 - \dfrac{1}{m} \;<\; 1$.

One way to fix this loss of accuracy would be to change the problem into an equivalent one: instead of solving $f(x) = 0$ which has $\alpha$ as a multiple root, consider the equation

$$u(x) \;:=\; \frac{f(x)}{f'(x)} \;=\; 0, \tag{4.3}$$

for which $\alpha$ is a *simple* root. Then Newton's method is defined by

$$x_{n+1} = x_n - \frac{u(x_n)}{u'(x_n)}. \tag{4.4}$$

We have

$$u' = \frac{(f')^2 - ff''}{(f')^2}, \quad \frac{u}{u'} = \frac{\frac{f}{f'}}{\frac{(f')^2 - ff''}{(f')^2}} = \frac{ff'}{(f')^2 - ff''},$$

so Newton's method is given by

$$x_{n+1} = x_n - \frac{f(x_n)f'(x_n)}{\left(f'(x_n)\right)^2 - f(x_n)f''(x_n)}, \; n \geq 0. \tag{4.5}$$

Although this method restores the order of convergence $p = 2$, it has several disadvantages: it requires the computation of the second derivative $f''$, it involves more complex computations than the original method, and the denominator in (4.5) can take very small values, as $x_n \to \alpha$.

A better alternative is to modify the *method*, instead of the *function*.

## Newton's Method for Multiple Roots

To improve Newton's method, we would like a function $g$ for which $g'(\alpha) = 0$ (as before), even for multiple roots. Consider the following: If $\alpha$ is a root of $f$, then in the vicinity of $\alpha$, we have

$$f(x) = (x-\alpha)^m h(x) \approx (x-\alpha)^m c,$$

for some constant $c$. Then

$$f'(x) \approx m(x-\alpha)^{m-1} c, \quad \frac{f(x)}{f'(x)} \approx \frac{x-\alpha}{m},$$
$$x - \alpha \approx m \frac{f(x)}{f'(x)}, \quad \alpha \approx x - m \frac{f(x)}{f'(x)}.$$

Thus, we define **Newton's method for multiple roots** by

$$x_{n+1} = x_n - m \frac{f(x_n)}{f'(x_n)}. \tag{4.6}$$

Now, indeed, it is easy to check (by our earlier computations) that for $g(x) = x - m \dfrac{f(x)}{f'(x)}$, we have

$$g(\alpha) = \alpha, \; g'(\alpha) = 1 - m\,\varphi(\alpha) = 0,$$

so the method converges quadratically again and

$$\lim_{n\to\infty} \frac{x_{n+1}-\alpha}{(x_n-\alpha)^2} = \frac{1}{2} g''(\alpha).$$

## 5 Newton's Method for Nonlinear Systems

Many of the methods considered in the previous sections can be generalized to the multidimensional case, i.e. to systems of nonlinear equations. These problems are widespread in applications, and they are varied in form. There is a great variety of methods for the solution of such systems. We only consider the two-dimensional case

$$\begin{aligned} f_1(x_1, x_2) &= 0 \\ f_2(x_1, x_2) &= 0, \end{aligned} \tag{5.1}$$

or, in vector notation,

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}, \; \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \; \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix}. \tag{5.2}$$

The one-point iteration theory discussed in the previous sections still stands, with appropriate adjustments (norm instead of absolute value and *Jacobian matrix* instead of derivative). Newton's method is derived similarly with the one-dimensional case, considering Taylor series expansions of each $f_i, i = 1, 2$, expanding $f_i(\boldsymbol{\alpha})$ about $\boldsymbol{x_0} = [x_{1,0} \;\; x_{2,0}]^T$. We get

$$\boldsymbol{x_{n+1}} = \boldsymbol{x_n} - \big(\boldsymbol{J_f}(\boldsymbol{x_n})\big)^{\boldsymbol{-1}} \boldsymbol{f}(\boldsymbol{x_n}), \; n \geq 0, \tag{5.3}$$

where $\boldsymbol{J_f}(\boldsymbol{x_n})$ is the Jacobian matrix of $\boldsymbol{f}$ at $\boldsymbol{x_n}$

$$\boldsymbol{J_f}(\boldsymbol{x_n}) = \begin{bmatrix} \dfrac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \dfrac{\partial f_1(\boldsymbol{x})}{\partial x_2} \\ \dfrac{\partial f_2(\boldsymbol{x})}{\partial x_1} & \dfrac{\partial f_2(\boldsymbol{x})}{\partial x_2} \end{bmatrix} \tag{5.4}$$

This is **Newton's method for nonlinear systems**.

In actual practice, we *do not invert* $\boldsymbol{J_f}(\boldsymbol{x_n})$, particularly for systems of more than two equations. Instead we solve a linear system for a correction term to $\boldsymbol{x_n}$:

$$\begin{aligned} \boldsymbol{J_f}(\boldsymbol{x_n})\boldsymbol{\delta_{n+1}} &= -\boldsymbol{f}(\boldsymbol{x_n}), \\ \boldsymbol{x_{n+1}} &= \boldsymbol{x_n} + \boldsymbol{\delta_{n+1}}. \end{aligned} \tag{5.5}$$

This is more efficient in computation time, requiring only about one-third as many operations as inverting $\boldsymbol{J_f}(\boldsymbol{x_n})$.

**Example 5.1.** Solve the system

$$\begin{aligned} f_1(x_1, x_2) &\equiv 4x_1^2 + x_2^2 - 4 = 0 \\ f_2(x_1, x_2) &\equiv x_1 + x_2 - \sin(x_1 - x_2) = 0. \end{aligned}$$

**Solution.** There are only two roots, one near $(1, 0)$ and its reflection about the origin near $(-1, 0)$ (see Figure 1).

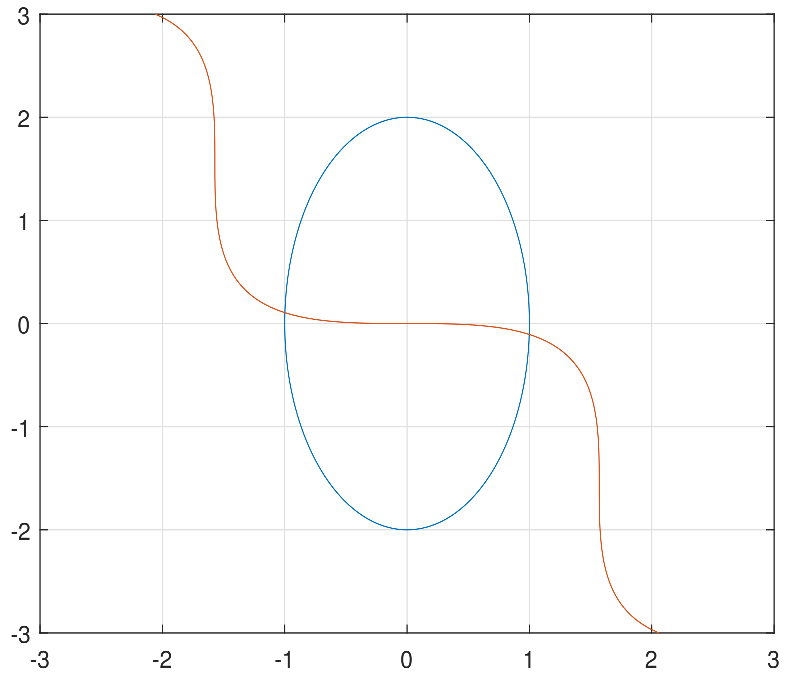

Fig. 1: Example 5.1

Using Newton's method with $\boldsymbol{x_0} = [1 \;\; 0]^T$ we obtain the results in Table 1.

| $n$ | $x_{1,n}$ | $x_{2,n}$ | $f_1(\boldsymbol{x_n})$ | $f_2(\boldsymbol{x_n})$ |
|---|---|---|---|---|
| 0 | 1.0 | 0.0 | 0.0 | $1.59e-1$ |
| 1 | 1.0 | $-0.1029207154$ | $1.06e-2$ | $4.55e-3$ |
| 2 | 0.9986087598 | $-0.1055307239$ | $1.46e-5$ | $6.63e-7$ |
| 3 | 0.9986069441 | $-0.1055304923$ | $1.32e-11$ | $1.87e-12$ |

Table 1: Newton's Method for Example 5.1

■