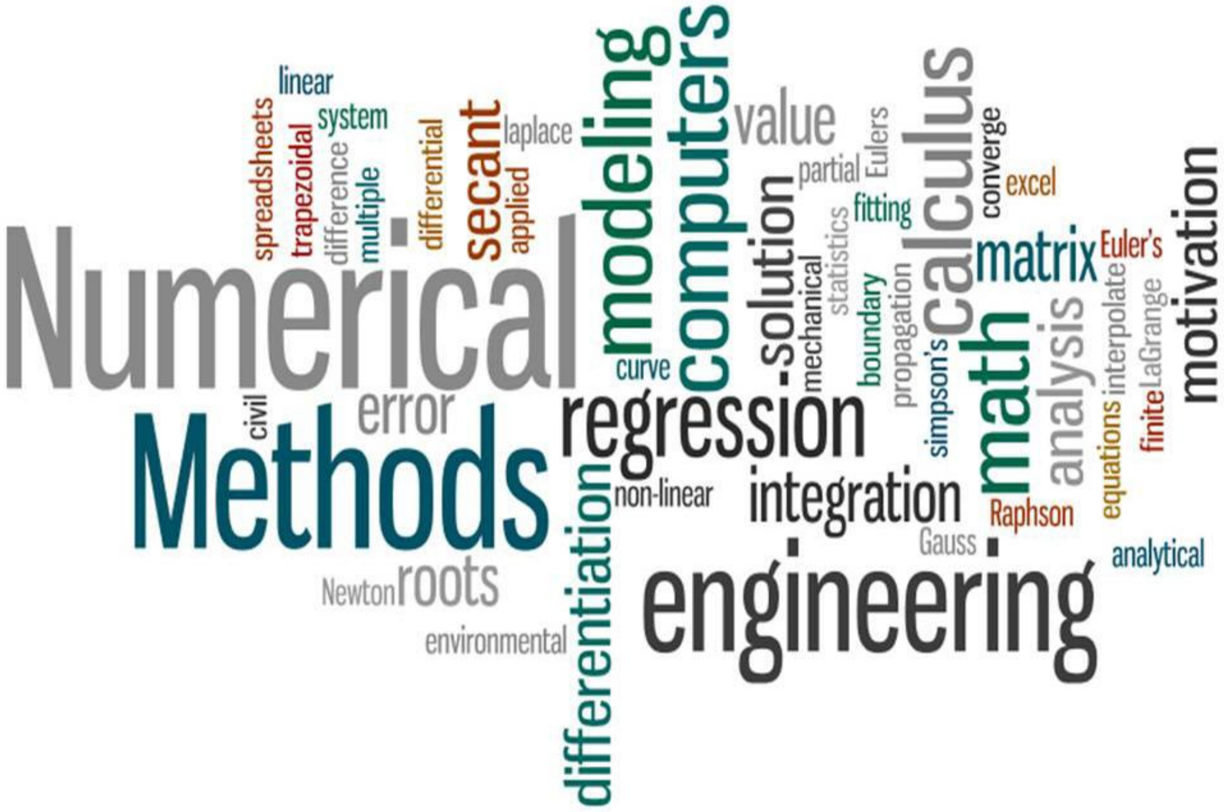
Sétif 1 University-Ferhat ABBAS
Faculty of Sciences
Department of mathematics

FACULTY OF SCIENCES

Numerical Methods and Programming

with MATLAB implementation

Dr. Raouf Ziadi



Conforming to the second-year undergraduate Physics/Chemistry curriculum
2024/2025

Raouf Ziadi

Numerical Methods and Programming

Conforming to the second-year undergraduate Physics/Chemistry curriculum

2024/2025

Setif 1 University - FERHAT ABBAS
Faculty of Sciences
Departement of Mathematics

Contents

Introduction

Engineers are constantly confronted with concrete problems in their respective fields. The majority of these problems can be formulated as mathematical problems (such as solving an equation, calculating an integral, and so on), and the majority of these problems are not resolvable by traditional analytical methods, or we are convinced that it will take a long time to solve them analytically if this is not possible, that is why we then resort to numerical methods.

For example, the integrals $\int_{-2}^{3} e^{-x^2} dx$ and $\int_{-\pi}^{\pi} \sin(x^2) dx$ cannot be calculated using classical methods such as integration by parts, substitution, etc. Using a numerical method, this type of integration can be approximated numerically with a given accuracy.

Over the past few decades, numerous algorithms based on theoretical studies have been developed to solve various mathematical problems. This course is designed for second-year undergraduate students of physics/chemistry. It constitutes an introduction to numerical calculation and consequently to the different techniques that these students will frequently have to use, without delving into the theoretical considerations and foundations of the methods presented. In fact, we have described the most well-known and widely used numerical methods to solve various problems encountered during their studies, focusing on the practical aspects in the presentation of these methods.

In this booklet, each section is followed by detailed examples, and at the end of each chapter, students are encouraged to work on additional exercises. The course is structured into five main chapters. The first chapter is dedicated to numerical integration. The second chapter focuses on the numerical solution of nonlinear equations. The third chapter describes numerical techniques for solving ordinary differential equations. The fourth chapter is dedicated to methods for solving systems of linear equations. Finally, the last chapter covers two methods of polynomial interpolation.

1
Numerical Integration

Often, the explicit computation of the integral of a continuous function f over an interval $[a, b]$ of $\mathbb{R}$ can be very expensive or can not be solved analytically. To overcome these difficulties, we use numerical methods to calculate an approximation of an integral $\int_a^b f(x)dx$ within a given precision. In this chapter, we present three common integration techniques: Midpoint method, Trapezoid method, and Simpson method. Using these algorithms we can even approximate the integral of functions known only through discrete data points.

The idea is to approximate $\int_a^b f(x)dx$ by a finite linear combination, i.e

$$I(f) = \int_a^b f(x)dx \simeq \sum_{i=0}^{n} \lambda_i f(x_i),$$

with $x_i \in [a, b]$, $\lambda_i \in \mathbb{R}$ and the calculation error is given by:

$$\mathcal{R}_n(f) = \int_a^b f(x)dx - \sum_{i=0}^{n} \lambda_i f(x_i).$$

1.1 Midpoint Method

The classical formula of the midpoint method (or rectangle method) is obtained by replacing f with its value at the midpoint of the interval $[a, b]$ (see Figure 1.1).

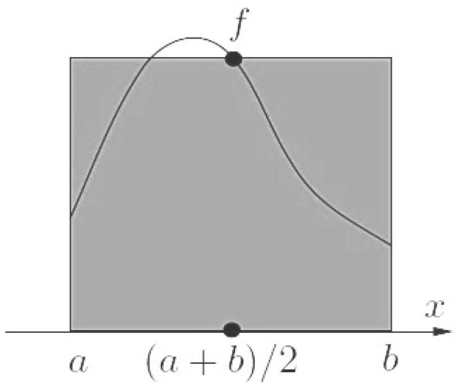


Figure 1.1: Midpoint formula

The simple midpoint formula is obtained by using the following formula on the interval $[a, b]$:

$$I(f) = (b-a)f\left(\frac{b+a}{2}\right)$$

Composite Midpoint rule

The composite midpoint formula is obtained by applying the previous midpoint formula to each subinterval $[x_{i-1}, x_i]$, $i = 1, ..., n$, with $x_i = a + i \times h, i = 0, \ldots, n$ and $h = (b-a)/n$.

By repeating the previous midpoint formula to each subinterval with centers $\tilde{x}_i = \frac{x_{i-1}+x_i}{2}$, the integral of the function is approximated by the following sum-up:

$$I(f) = h \times f(\tilde{x}_1) + h \times f(\tilde{x}_2) + \cdots + h \times f(\tilde{x}_n),$$

we then obtain the following general formula:

$$I(f) = h \times \sum_{i=1}^{n} f(\tilde{x}_i).$$

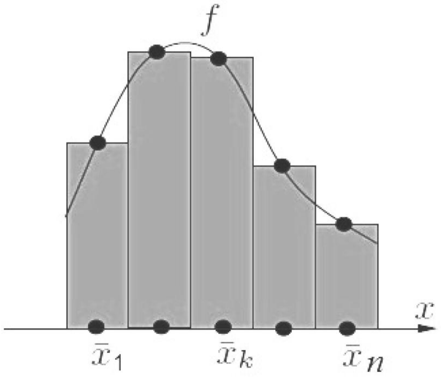


Figure 1.2: Composite midpoint formula over five subintervals

Example 1.1. *Let us integrate the function* $f(x) = 3x^2 + 2x$ *over the interval* $[1, 2]$. *This function is very simple to integrate analytically:*

$$\int_1^2 f(x)\,dx = 10.$$

Using the midpoint method with $n = 4$, *the steplength is* $h = \frac{2-1}{4} = 0.25$ *and the evaluated points are:* $\tilde{x}_1 = \frac{1+1.25}{2} = 1.1250, \tilde{x}_2 = 1.3750, \tilde{x}_3 = 1.6250,$ *and* $\tilde{x}_4 = 1.8750$. *Then,*

$$I(f) = 0.25[f(1.1250) + f(1.3750) + f(1.6250) + f(1.8750)] = 9.9844$$

By increasing n *to* 8*, the new the steplength is* $h = \frac{1}{8} = 0.125$*, and the new approximation is:*

$$\begin{aligned} I(f) &= 0.125[f(1.0625) + f(1.1875) + f(1.3125) + f(1.4375) \\ &\quad + f(1.5625) + f(1.6875) + f(1.8125) + f(1.9375)] \\ &= 9.9961, \end{aligned}$$

and with $n = 100$ *we get* $I(f) = 9.999975$*.*

Theorem 1.1. *Let* $f \in \mathcal{C}^2([a,b])$*. Then, there exists* $\xi \in [a,b]$ *such that:*

$$\mathcal{R}_n(f) = -\frac{(b-a)^3}{24n^2} f''(\xi) = -\frac{b-a}{24} h^2 f''(\xi),$$

where f'' *denotes the second derivative of the function* f*. The upper bound of the calculation error can be expressed as follows:*

$$\mathcal{R}_n(f) \leq \frac{(b-a)^3}{24n^2} \max_{x \in [a,b]} \left| f''(x) \right|$$

Remark 1.1. *Given a precision* ε*, the minimum number* n *of subdivisions can be determined using the following formula:*

$$n \geq \sqrt{\frac{(b-a)^3 \max\limits_{x \in [a,b]} \left| f''(x) \right|}{24\,\varepsilon}}.$$

1.2 Trapezoidal Method

This formula is very simple; it allows replacing the curve $\mathcal{C}_f$ of the function f to be integrated by a straight line that connects the points $\big(a, f(a)\big)$ and $\big(b, f(b)\big)$, which forms a trapezoid (see Figure 1.3 below).

The integral is thus replaced by the area of the trapezoid:

$$I(f) \simeq S = \frac{b-a}{2} \left[f(a) + f(b) \right].$$

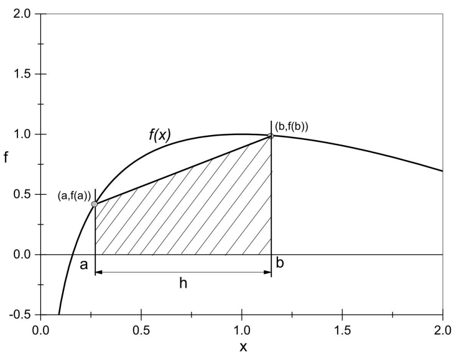


Figure 1.3: Trapezoidal method

We can observe that there is a significant difference between the function's curve and the straight line, which means that a calculation error was made. To minimize this error, we use a more suitable version of this formula.

Composite Trapezoidal rule

To obtain better results, we divide the interval $[a, b]$ into n subintervals, and we apply the trapezoidal method to each of them: $[a = x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, b = x_n]$. Applying the trapezoidal formula gives:

$$\begin{aligned} I(f) &\simeq \frac{h}{2}\big(f(x_0) + f(x_1)\big) + \frac{h}{2}\big(f(x_1) + f(x_2)\big) + \ldots \\ &+ \frac{h}{2}\big(f(x_{n-1}) + f(x_n)\big) \\ &\simeq \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]. \end{aligned}$$

Hence

$$I(f) \simeq \frac{h}{2}\left[f(x_0) + f(x_n) + 2\sum_{i=1}^{n-1}(f(x_i))\right]. \tag{1.1}$$

For example, in Figure 1.4 below, for these four trapezoids we have:

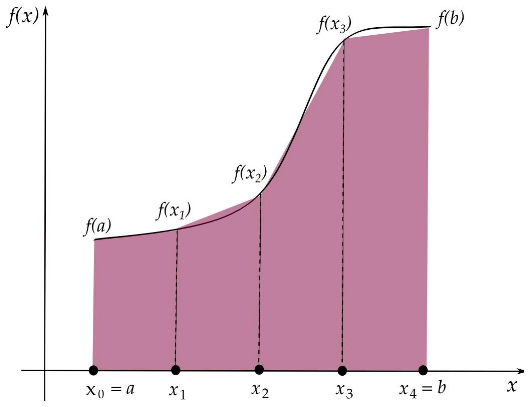


Figure 1.4: Composite trapezoidal formula represented over 4 subintervals.

$$
\begin{aligned}
I_1(f) &= \frac{h}{2}\big(f(x_1) + f(a)\big),\\
I_2(f) &= \frac{h}{2}\big(f(x_1) + f(x_2)\big),\\
I_3(f) &= \frac{h}{2}\big(f(x_2) + f(x_3)\big),\\
I_4(f) &= \frac{h}{2}\big(f(x_3) + f(b)\big).
\end{aligned}
$$

It follows that, $I(f) \simeq I_1(f) + I_2(f) + I_3(f) + I_4(f)$.

Theorem 1.2. *Let $f \in \mathcal{C}^2([a,b])$. Then, there exists $\xi \in [a,b]$ such that:*

$$\mathcal{R}_n(f) = -\frac{(b-a)^3}{12n^2} f''(\xi) = -\frac{b-a}{12} h^2 f''(\xi),$$

where f'' denotes the second derivative of the function f. We can write the upper bound of the error made as follows:

$$\mathcal{R}_n(f) \leq \frac{(b-a)^3}{12n^2} \max_{x\in[a,b]} \left|f''(x)\right|.$$

Remark 1.2. *Given a precision ε, we can determine the minimum number n of subintervals using the following formula:*

$$n \geq \sqrt{\frac{(b-a)^3 \max\limits_{x\in[a,b]} \left|f''(x)\right|}{12\,\varepsilon}}.$$

There exists an improved version of the trapezoidal method, called the ***Poncelet*** *method, whose numerical scheme is given by:*

$$I(f) \simeq \frac{h}{4}\Big(f(x_0)+f(x_{2n})+7\big(f(x_1)+f(x_{2n-1})\big)+8\sum_{i=1}^{n-2} f(x_{2i+1})\Big).$$

Example 1.2. *Let the function $f(x) = \sqrt{x+1}$ with $x \in [0,1]$. Using the trapezoidal method, calculate the integral $\displaystyle\int_0^1 \sqrt{x+1}\,dx$ with $n = 10$ and evaluate the calculation error.*

<u>Solution:</u>

For $n = 10$, the step size is $h = \frac{b-a}{n} = 0.1$, and the sequence of evaluated points are listed in the table below

x_i	*0*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1*
$f(x_i)$	*1*	*1.0481*	*1.0954*	*1.1401*	*1.1832*	*1.2247*	*1.2649*	*1.3038*	*1.3416*	*1.3784*	*1.4142*

Applying the composite trapezoidal formula, it results that

$$\begin{aligned} I(f) &= \frac{h}{2}\left[f(a)+f(b)+2\sum_{i=1}^{9} f(x_i)\right] \\ &= \frac{0.1}{2}\,[1+1.4142+2\,(1.0488+1.0954+1.1402+1.1832 \\ &\quad +1.2247+1.2649+1.3038+1.3416+1.3784)] = 1.2188. \end{aligned}$$

Hence, $\displaystyle\int_0^1 \sqrt{x+1}dx \simeq 1.2188.$

- *Calculation error:* $\mathcal{R}_n(f) \leq \frac{1}{12\times 10^2} \max\limits_{x\in[0,1]} \left|f''(x)\right|$, *on the other* *hand, we have*

$$\begin{aligned} f'(x) &= \frac{1}{2}(x+1)^{-\frac{1}{2}}, \\ f''(x) &= -\frac{1}{4}(x+1)^{-\frac{3}{2}}, \\ f^{(3)}(x) &= \frac{3}{8}(x+1)^{-\frac{5}{2}} > 0, \forall x \in [0,1]. \end{aligned}$$

Hence, $\max_{x\in[0,1]} |f''(x)| = |f''(0)| = 0.25$*, then:* $\mathcal{R}_n(f) \leq 0.25 \times \frac{1}{12\times 10^2} \simeq 2.08 \times 10^{-4}$.
Therefore,

$$\int_0^1 \sqrt{x+1}dx \simeq 1.2188 \pm 2.08 \times 10^{-4}.$$

Example 1.3. *Compute the integral* $\int_0^1 e^{-x^2}\, dx$ *with a precision of* 10^{-3} *by the trapezoid method.*
<u>Solution:</u>
We have first to determine the number of divisions n *needed to obtain this precision.*
The integration error is written as:

$$\mathcal{R}_n(f) \leq \frac{(b-a)^3}{12n^2} \max_{x\in[a,b]} \left|f''(x)\right| = \frac{b-a}{12}h^2 \max_{x\in[a,b]} \left|f''(x)\right| \leq 10^{-3}.$$

On the other hand, we have: $f''(x) = (4x^2-2)\exp(-x^2)$*, which is strictly increasing on the interval* $[0,1]$ *and* $\max_{x\in[0,1]} |f''(x)| = |f''(0)| = 2$.
Then, $\mathcal{R}_n(f) \leq \frac{b-a}{12}h^2 \max_{x\in[a,b]} \left|f''(x)\right| \leq 10^{-3}$*, so,* $h \leq \sqrt{\frac{12\times 0.001}{(1-0)\times 2}} = 0.0774$*, therefore* $n \geq \frac{1}{0.0774} = 12.91$*. We need* 13 *divisions, using the same technique as in Example 1, we get*

$$\int_0^1 e^{-x^2}\, dx \simeq 0.74646 \pm 10^{-3}.$$

Example 1.4. *Using the trapezoid method, calculate the integral* $\int_0^{\pi} \sin x^2 dx$ *with* 5 *intervals.*
- Knowing that the exact value is 0.7726*; compare the obtained result with the exact value.*

Solution:

For five subintervals, the steplength $h = \frac{b-a}{n} = \frac{\pi}{5}$ *and the evaluated points are:*

x_i	*0*	$\pi/5$	$2\pi/5$	$3\pi/5$	$4\pi/5$	π
$f(x_i)$	*0*	*0,3846*	*1,0000*	*- 0,3999*	*0,0333*	*- 0,4303*

We have also:

$$\begin{aligned} I(f) &= \frac{h}{2}\left[f(a) + f(b) + 2\left(\sum_{i=1}^{9} f(x_i)\right)\right] \\ &= \frac{\pi}{10}\left[0 - 0,4303 + 2(0.3846 + 1 - 0.3999 + 0.0333)\right] \\ &= 0,5044. \end{aligned}$$

Hence, $\int_0^{\pi} \sin x^2 dx \simeq 0,5044.$

- ***Comparison :***

We have $\int_0^{\pi} \sin x^2 dx = 0.7726$*,* $I(f) = 0,5044$ *and* $\left|I(f) - \int_0^{\pi} \sin x^2 dx\right| = 0.2682.$

For $n = 5$*, the absolute error between the exact and the obtained result is* 0.2682*.*

Example 1.5. *Consider the integral defined by* $\int_1^3 (1 + \log(x))\, dx.$
- Determine the number of subintervals required to achieve an integration error less than 10^{-3}*.*

Solution

Achieving an error $\mathcal{R}_n(f) < 10^{-3} \Longleftrightarrow \frac{(3-1)}{12n^2} \times 0.1111 < 10^{-3}$*, thus* $n^2 > 18.5$*, which implies* $n > 4.30$*. It follows that with five subintervals, we achieve an error of less than* 10^{-3}*.*

1.3 Simpson Method

In Simpson formula, the function is not replaced by a straight line but by a parabola that must pass through three points $(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))$, which means that this method is only applicable for an even number of slices; see Figure 1.5 below.

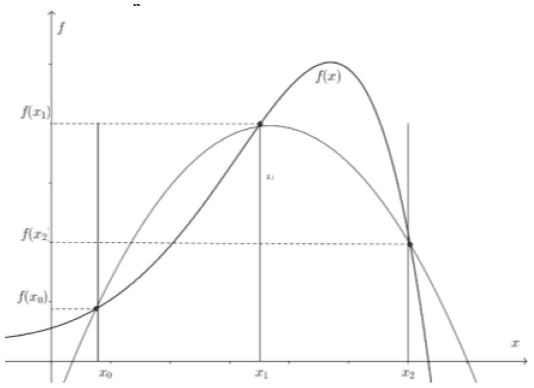


Figure 1.5: Simpson method

Simpson formula is written as:

$$\int_a^b f(x)\,dx \simeq \frac{b-a}{6}\left(f(x_0) + 4f(x_1) + f(x_2)\right)$$

Composite Simpson rule

We subdivide the interval $[a, b]$ into n (with n even $(n = 2k \mid k \in \mathbb{N})$) subintervals with $h = \frac{b-a}{n}$, and we apply the Simpson method for each interval of the form $[a, x_2], [x_2, x_4], \ldots, [x_{n-2}, b]$ (see Figure 1.6 below). The numerical scheme of this method is given by:

$$I(f) = \frac{h}{3}\left[f(a) + f(b) + 2\sum_{i=1}^{k-1} f(x_{2i}) + 4\sum_{i=1}^{k} f(x_{2i-1})\right]$$

In Figure 1.6 below, to compute the coloured area, we subdivide the interval $[a, b]$ into four slices, then we apply the standard Simpson formula for each slice, i.e. each subinterval is interpolated by its degree-two Lagrange polynomial over three nodes $x_{2i}, x_{2i} + h, x_{2i} + 2h$ for $i = 0, 1, 2, 3$.

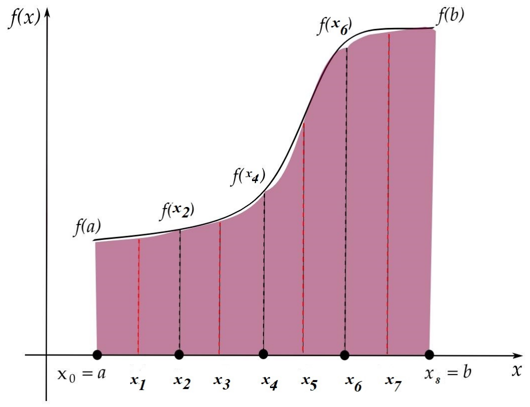


Figure 1.6: Composite Simpson formula represented over four subintervals.

For the four subintervals we write:

$$
\begin{aligned}
I_1(f) &= \frac{h}{3}\left(f(x_0) + 4f(x_1) + f(x_2)\right) \\
I_2(f) &= \frac{h}{3}\left(f(x_2) + 4f(x_3) + f(x_4)\right) \\
I_3(f) &= \frac{h}{3}\left(f(x_4) + 4f(x_5) + f(x_6)\right) \\
I_4(f) &= \frac{h}{3}\left(f(x_6) + 4f(x_7) + f(x_8)\right).
\end{aligned}
$$

Therefore $I(f) \simeq I_1(f) + I_2(f) + I_3(f) + I_4(f)$.

Theorem 1.3. *Let $f \in \mathcal{C}^4([a, b])$. Then, there exists $\xi \in [a, b]$ such that:*

$$\mathcal{R}_n(f) = -\frac{(b-a)^5}{180n^4} f^{(4)}(\xi) = -\frac{b-a}{180n^2} h^4 f^{(4)}(\xi),$$

where $f^{(4)}$ denotes the fourth derivative of the function f. We can write the upper bound of the error made as follows:

$$\mathcal{R}_n(f) \leq \frac{(b-a)^5}{180n^4} \max_{x\in[a,b]} \left|f^{(4)}(x)\right|$$

Remark 1.3. *Let ε be the required precision, the number of subintervals n can be determined by:*

$$n \geq \sqrt[4]{\frac{(b-a)^5 \max\limits_{x\in[a,b]} \left|f^{(4)}(x)\right|}{180\varepsilon}}.$$

Example 1.6. *Let the function $f(x) = \frac{1}{2x+1}$, with $x \in [0,1]$. Using Simpson method, calculate the integral $\displaystyle\int_0^1 \frac{1}{2x+1}\,dx$ with $n = 10$ and evaluate the error.*

<u>Solution</u>

For $n = 10$ (i.e. 5 slices), the step-length is $h = \frac{b-a}{n} = 0.1$ and the evaluated points are listed in table below

x_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$f(x_i)$	1	0.8333	0.7142	0.625	0.5555	0.5	0.4545	0.4166	0.3846	0.3571	0.3333

Using Simpson formula, we get

$$I(f) = \tfrac{0.1}{3}\left[1 + 0.3333 + 4\left(0.8333 + 0.625 + 0.5 + 0.4166 + 0.3571\right) \right.$$
$$\left. +2\left(0.7142 + 0.5555 + 0.4545 + 0.3846\right)\right] = 0.5493$$

Furthermore, the function $f^{(4)}(x) = \frac{384}{(2x+1)^5}$ *is strictly decreasing on* $[0,1]$. *Then,*

$$\max_{x\in[0,1]} \left|f^{(4)}(x)\right| = f(0) = 384,$$

hence,

$$\mathcal{R}_n(f) \leq \frac{384}{180\times 10^4} = 2.116\times 10^{-4}.$$

Therefore,

$$\int_0^1 \frac{1}{2x+1}\,dx \simeq 0.5493 \pm 2.116\times 10^{-4}.$$

Example 1.7. *Compute the integral* $\displaystyle\int_0^1 e^{-x^2}\,dx$ *with a precision of* 10^{-3} *using Simpson method.*

Solution

We must first determine the number of divisions n *required to achieve this precision.*
The integration error is written as:

$$\mathcal{R}_n(f) \leq \frac{(b-a)^5}{180n^4}\max_{x\in[a,b]}\left|f^{(4)}(x)\right| = h^4\frac{b-a}{180}\max_{x\in[a,b]}\left|f^{(4)}(x)\right| \leq 10^{-3}.$$

We have $f^{(4)}(x) = (16x^4 - 48x^2 + 12)e^{-x^2}$, *and* $\max_{x\in[0,1]}\left|f^{(4)}(x)\right| = f(0) = 12$. *Then*

$$h \leq \sqrt[4]{\frac{180\times 0.001}{(1-0)\times 12}} = 0.35$$

Thus due to fact that n *must be even number, i.e.* $n = 2k \geq \frac{1}{0.35} = 2.85$, *it results that* $k = 2$ *for which we take* $n = 4$ *with a bsteplength* $h = 0.25$. *Following the same steps as in the previous example, we obtain:*

$I(f) \simeq 0.7469$. *Therefore,*

$$\int_0^1 e^{-x^2}\, dx \simeq 0.7469 \pm 10^{-3}.$$

1.4 Matlab codes

1.4.1 Pseudo-code of Midpoint method

```
% Midpoint Method for Numerical
   Integration
clc; clear;
% Define the function to integrate
f = @(x)x.^2;  % Example: f(x) = x^2
a = 0;          % Lower limit
b = 1;          % Upper limit
n = 10;         % Number of subintervals
h = (b-a)/n;
integral_approx = 0;
for i = 1:n
      midpoint = a+(i-0.5)*h;
      integral_approx = integral_approx +
         f(midpoint);
end
integral_approx = integral_approx * h;
% Display the result
fprintf('Approximate integral using the
   Midpoint Method: %.6f\n',
   integral_approx);
```

1.4.2 Pseudo-code of Trapezoidal method

```
clc; clear;
% Define the function to integrate
f = @(x)x.^2;  % Example: f(x) = x^2
a = 0;          % Lower limit
b = 1;          % Upper limit
n = 10;         % Number of subintervals
h=(b-a)/n;
% Evaluate the function at the endpoints
sum=f(a)+f(b);
% Evaluate the function at the
   intermediate points and sum them up
for i = 1:n-1
    sum= sum+2*f(a+i*h);
end
% Multiply by the step size and divide by
   2
I=(h/2)*sum;
% Display the result
fprintf('Approximate integral using
   Trapezoidal Method: %.6f\n', I);
```

1.4.3 Pseudo-code of Simpson method

```
% Simpson algorithm for numerical
   integration
clc;
clear;
% Define the function to integrate
```

```
f = @(x) x.^2 + 3*x + 2;  % Example: f(x)
   = x^2 + 3x + 2
% Input limits of integration and number
   of subintervals (must be even)
a = 0;        % Lower limit
b = 4;        % Upper limit
n = 6;        % Number of subintervals (
   must be even)
% Check if n is even
if mod(n, 2) ~= 0
    error('Number of subintervals (n) must
        be even.');
end
h = (b - a) / n; % Step size
x = a:h:b;% Generate x values
y = f(x);% Evaluate function at x values
I = y(1) + y(end);
for i = 2:n
    if mod(i, 2) == 0
        I = I + 4 * y(i);  % 4*f(x_odd)
    else
        I = I + 2 * y(i);  % 2*f(x_even)
    end
end
I = I * h / 3;
% Display the result
fprintf('Approximate integral value using
   Simpson algorithm: %.6f\n', I);
```

1.5 Solved Exercises

Exercise 1.1. .

a- *Determine by the trapezoidal and Simpson methods an approximate value of the integral* $\int_0^2 f(x)dx$ *using the data of the following table:*

x_i	*0*	*1/2*	*1*	*1.5*	*2*
$f(x_i) = e^{x^2}$	*1*	*1.284*	*2.718*	*9.487*	*54.598*

b- *Estimate the calculation error in each case using* 10 *subdivision.*

c- *What is the number of required subdivision to achieve a precision of* $\epsilon(I) = 10^{-2}$ *for each method?*

Solution

a.1- The approximation of the integral by the trapezoidal method.

$$\begin{aligned} I(f) &= \frac{h}{2}\left[f(x_0) + f(x_n) + 2\sum_{i=1}^{n-1}(f(x_i))\right] \\ &= \frac{0.5}{2}[1 + 54.598 + 2(1.284 + 2.718 + 9.487)] \\ &= 20.644 \end{aligned}$$

a.2- Approximating the integral using the Simpson.

$$\begin{aligned}
I(f) &= \frac{h}{3}\left[f(0)+f(2)+2\sum_{i=1}^{k-1} f(x_{2i})+4\sum_{i=1}^{k} f(x_{2i-1})\right]\\
&= \frac{0.5}{3}[f(0)+f(2)+2f(1)+4(f(0.5)+f(1.5))]\\
&= \frac{0.5}{3}(1+54.598+2\times 2.718+4\times(1.284+9.487))\\
&= 26.0295
\end{aligned}$$

b.1- Evaluating the error by the trapezoidal method with $n = 10$.
We have $f''(x) = 2e^{x^2} + 4x^2e^{x^2}$ and $\max_{x\in[0,2]} \left|f''(x)\right| = 982.766$, thus:

$$\begin{aligned}
\mathcal{R}_n(f) &\leq \frac{(b-a)^3}{12n^2} \max_{x\in[a,b]} \left|f''(x)\right|\\
&\leq \frac{8}{12\times 10^2} 982.766 = 6.551
\end{aligned}$$

b.2- Evaluating the error by Simpson method with $n = 10$.
We have $f^{(4)}(x) = 12e^{x^2} + 48x^2e^{x^2} + 16x^4e^{x^2} = e^{x^2}(16x^4 + 48x^2 + 12)$ and $\max_{x\in[0,2]} \left|f^{(4)}(x)\right| = \left|f^{(4)}(2)\right| = 39092.275$, thus:

$$\begin{aligned}
\mathcal{R}_n(f) &\leq \frac{(b-a)^5}{180n^4} \max_{x\in[a,b]} \left|f^{(4)}(x)\right|\\
&\leq \frac{2}{180\times 10^4} 39092.275 = 0.6950
\end{aligned}$$

c.2- Calculating the number of points required to achieve a precision 10^{-2} with the trapezoidal method.

We have:

$$n \geq \sqrt{\frac{(b-a)^3 \max\limits_{x\in[a,b]} \left|f''(x)\right|}{12\,\varepsilon}}$$
$$\geq \sqrt{\frac{982.766 \times 2^3}{12 \times 10^2}} \simeq 256$$

Thus, the number of required subdivisions is $n \geq 256$.
c.1- Calculating the number of required subdivision to achieve a precision of 10^{-2} with Simpson method.
We have:

$$n \geq \sqrt[4]{\frac{(b-a)^5 \max\limits_{x\in[a,b]} \left|f^{(4)}(x)\right|}{180\varepsilon}}$$
$$\geq \sqrt[4]{\frac{39092.275 \times 2^5}{180 \times 10^{-2}}} \simeq 51.344$$

Thus, the number of required subdivisions using Simpson method is $n \geq 52$.

Exercise 1.2. *Consider the following integral* $I = \displaystyle\int_0^{\pi} \sin(x)dx.$
1. Calculate the exact value of I.
2. Using the trapezoidal and Simpson methods with $h = \frac{\pi}{4}$*:*

a- Approximate the value of the integral I.

b- Estimate the calculation error.

c- Evaluate the absolute error.

3. Find the value of steplenght h *and the number of required subdivisions so that the error obtained by the trapezoidal (resp. Simpson) method is less than* 5×10^{-4}.

Solution.

1. $I = \displaystyle\int_0^{\pi} \sin(x)dx = 2.$

2. I- Approximating the integral using the trapezoidal method:

a-

$$\begin{aligned} I(f) &= \frac{h}{2}\left[f(x_0) + f(x_n) + 2\sum_{i=1}^{3} f(x_i)\right] \\ &= \frac{\pi}{8}(f(0) + f(\pi) + 2(f(\pi/4) + f(\pi/2) + f(3\pi/4))) \\ &\simeq 1.896. \end{aligned}$$

b- We have

$$\begin{aligned} \mathcal{R}_n(f) &\leq \frac{(b-a)^3}{12n^2} \max_{x\in[a,b]} \left|f''(x)\right| = \frac{b-a}{12}h^2 \max_{x\in[a,b]} \left|f''(x)\right| \\ &\leq \frac{\pi^3}{192} \max_{x\in[0,\pi]} \left|\sin(x)\right| \leq \frac{\pi^3}{192} \simeq 0.16149. \end{aligned}$$

c- $\left|I(f) - \displaystyle\int_0^{\pi} \sin(x)dx\right| = 0.1038$

2. II- The Simpson method:

a-

$$\begin{aligned} I(f) =& \tfrac{h}{3}\left[f(a) + f(b) + 2\sum_{i=1}^{k-1} f(x_{2i}) + 4\sum_{i=1}^{k} f(x_{2i-1})\right], \text{ with } k = 2 \\ =& \frac{\pi}{12}(f(0) + f(\pi) + 2f(\frac{\pi}{2}) + 4(f(\frac{\pi}{4}) + f(\frac{3\pi}{4}))) \simeq 2.04. \end{aligned}$$

b- We have

$$\begin{aligned} \mathcal{R}_n(f) &\leq \frac{(b-a)^5}{180n^4} \max_{x\in[a,b]} \left|f^{(4)}(x)\right| \\ &\leq \frac{\pi^5}{180\times 4^4} \max_{x\in[0,\pi]} \left|\sin(x)\right| \leq \frac{\pi^5}{180\times 4^4} \simeq 0.0066 \end{aligned}$$

c- $|I(f) - \int_0^\pi \sin(x)dx| = 0.004$

3. Let's calculate the value of the steplenght h and the number of required subdivisions n so that the error is less than $\varepsilon = 5 \times 10^{-4}$.

3.I- The trapezoidal method:

We have

$$n \geq \sqrt{\frac{(b-a)^3 \max\limits_{x\in[a,b]} |f''(x)|}{12\,\varepsilon}}$$
$$\geq \sqrt{\frac{\pi^3 \max\limits_{x\in[0,\pi]} |\sin(x)|}{12 \times 5 \times 10^{-4}}} \simeq 71.8$$

Thus, the number of required subdivisions is $n \geq 72$.

3.II- The Simpson method:

We have

$$n \geq \sqrt[4]{\frac{(b-a)^5 \max\limits_{x\in[a,b]} |f^{(4)}(x)|}{180\varepsilon}}$$
$$\geq \sqrt[4]{\frac{\pi^5 \max\limits_{x\in[0,\pi]} |\sin(x)|}{180 \times 5 \times 10^{-4}}} \simeq 7.64116$$

Thus, the number of required subdivisions using Simpson method is $n \geq 8$.

Exercise 1.3. *A rocket is launched vertically from the ground, and its acceleration γ is measured during the first 80 seconds:*

t in (s)	*0*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
γ *in* m/s^2	*30*	*31.63*	*33.44*	*35.47*	*37.75*	*40.33*	*43.29*	*46.70*	*50.67*

- Calculate the velocity V of the rocket at $t = 80s$*, using the trapezoidal and Simpson methods.*

Solution.

We know that the acceleration γ is the derivative of the velocity V, so

$$V(t) = V(0) + \int_0^t \gamma(t)dt = 0 + \int_0^{80} \gamma(t)dt$$

- First, lets calculate $V(80)$ using the trapezoidal method. According to the previous table of values, we have $h = 10$ and $n = 8$. Therefore,

$$\begin{aligned} V(80) &= \frac{h}{2}\left(\gamma(t_0) + \gamma(t_2) + 2(\gamma(t_1)\cdots + \gamma(t_7))\right) \\ &= 5(30 + 50.67 + 2(31.63 + 33.44 + 35.47 + 37.75 + \\ &40.33 + 43.29 + 46.70)) = 3089 \text{ m/s}. \end{aligned}$$

- Next, we calculate $V(80)$ using Simpson method:

$$\begin{aligned} V(80) &= \frac{h}{3}\left[\gamma(t_0) + \gamma(t_k) + 2\sum_{i=1}^{3}\gamma(t_{2i}) + 4\sum_{i=1}^{4}\gamma(t_{2i-1})\right] \\ &= \frac{10}{3}(30 + 50.67 + 2(33.44 + 37.75 + 43.29) + \\ &4(31.63 + 35.47 + 40.33 + 46.70)) = 3087 \text{ m/s} \end{aligned}$$

Exercise 1.4.

a- Using 4 subintervals, determine by the trapezoidal method an approximate value of $\int_0^2 \sin^2(x)dx$*, and estimate the calculation error.*

b- What is the number of required subintervals to achieve a precision of 10^{-2} *by the trapezoidal method?*

Solution.

a- Approximating the integral using the trapezoidal method:

$$\begin{aligned}\int_0^2 f(x)dx \simeq I(f) &= \frac{h}{2}\left[f(x_0)+f(x_n)+2\sum_{i=1}^{n-1}(f(x_i))\right]\\ &= \tfrac{0.5}{2}[f(0)+f(2)+2(f(0.5)+f(1)+f(1.5))]\\ &= 1.173\end{aligned}$$

- Evaluating the error by the trapezoidal method with $n = 4$: We have $f''(x) = 2(\cos^2(x) - \sin^2(x)) = 2(1 - 2\sin^2(x))$ and $\max\limits_{x\in[0,2]} \left|f''(x)\right| = f''(0) = f''(\frac{\pi}{2}) = 2$, thus:

$$\begin{aligned}\mathcal{R}_n(f) &\leq \frac{(b-a)^3}{12n^2}\max_{x\in[a,b]}\left|f''(x)\right|\\ &\leq \frac{8}{12\times 4^2}2 = 0.0833\end{aligned}$$

b- The number of required subdivisions to achieve a precision of 10^{-3} with the trapezoidal method.
We have:

$$\begin{aligned}n &\geq \sqrt{\frac{(b-a)^3\max\limits_{x\in[a,b]}\left|f''(x)\right|}{12\,\varepsilon}}\\ &\geq \sqrt{\frac{2\times 2^3}{12\times 10^{-3}}} \simeq 36.51\end{aligned}$$

Thus, the number of required subdivisions is $n \geq 37$.

1.5.1 Supplementary Exercises

Exercise 1.5. *We consider the following integral:*

$$I = \int_0^1 \frac{dx}{1+x^2}dx.$$

1. Calculate the exact value of this integral.

2. Approximate the value of this integral numerically using:

- Midpoint method with 5 intervals.

- Trapezoidal method with 4 intervals.

- Simpson method with 2 intervals.

Result.

1. $I = \int_0^1 \frac{dx}{1+x^2} = 0.7854.$

2.

- By the midpoint rule, we obtain: $I(f) = 0.8387$

- By the trapezoidal method, we obtain: $I(f) = 0.7828$

- By Simpson method, we obtain: $I(f) = 0.7854$

Exercise 1.6. *Determine the required number of subdivisions to approximate the integral* $\int_0^1 xe^{-x}dx$ *with a precision of* 10^{-8} *using:*

1. Trapezoidal method.

2. Simpson method.

Result.

1. $n \geq 4083$

2. $n \geq 40$

Exercise 1.7. *Determine the number of required subdivisions to approximate the value of the integral* $\int_{-\pi}^{\pi} \cos(x)dx$ *with a precision of* 5×10^{-4} *using Simpson method.*

Result

- $n \geq 20$

Exercise 1.8. *Consider the following integral:*

$$I = \int_{\frac{-\pi}{3}}^{\frac{\pi}{3}} (e^x + \sin(x))\, dx$$

1- *Compute the exact value of I.*

2- *Give an approximation of the integral I with a precision of* $\varepsilon = 2 \times 10^{-4}$ *using Simpson's method.*

3- *Using the trapezoidal method with step size* $h = \frac{\pi}{6}$*:*

i) Provide an approximation of the integral I.

ii) Evaluate the error between the exact value and the approximate value of the integral.

Exercise 1.9. *Let* f *be a function defined on* $]0, +\infty[$ *by* $f(x) = e^{-x} - \ln(x)$.

1- Determine the analytical expression of $\int f(x)dx$.

2- Give the iterative scheme of the Simpson's algorithm to approximate a limited integral.

3- Using the Simpson's algorithm, approximate the value of the integral $\int_2^3 f(x)\, dx$ *with a precision of* 10^{-4}.

4- Using the trapezoidal method, determine the required number of subdivisions to approximate the integral $\int_2^3 f(x)\, dx$ *with a precision of* 10^{-4}.

2

Numerical Solution of Nonlinear Equations

In mathematics and applied sciences, nonlinear equations play a crucial role in modeling complex phenomena across various disciplines. Unlike linear equations, which can be solved analytically using algebraic methods, finding exact solutions for nonlinear equations is often difficult or even impossible.

For example, consider the equation:

$$\cos(x^3)\sin(2x^2 - 3) + 0.5 = 0.$$

It is evident that solving this equation analytically would be extremely time-consuming, if not impossible. These types of equations, known as nonlinear (transcendental) equations, can instead be solved numerically using methods that allow us to compute approximate roots with a specified level of precision.

In this chapter, we will explore three numerical methods for solving nonlinear univariate equations of the form $f(x) = 0$.

Definition 2.1. *Any number ξ that satisfies $f(\xi) = 0$ is called a solution (or root) of the equation $f(x) = 0$. Geometrically, ξ represents the x-coordinate of the point where the graph of the function $f(x)$ intersects the x-axis.*

Definition 2.2. *If the equation $f(x) = 0$ can be written in the form*

$$f(x) = (x - \xi)^m g(x) = 0$$

where $g(x) \neq 0$, then ξ is called a root of order m. If $m = 1$, ξ is called a simple root of the equation $f(x) = 0$.

In all iterative methods, it is necessary, to avoid divergence of the solution, to determine an interval containing the root being sought and to carefully choose the initial values.

2.1 Separation of Roots

Most numerical methods assume the existence of the desired root within a given interval $[a, b]$. In this case, the root is said to be localized or separated from any other potential roots.

Definition 2.3. *We say that a root ξ of an equation $f(x) = 0$ is separable if we can find an interval $[a, b]$ such that ξ is the only root of this equation in $[a, b]$. The root ξ is then called separated or localized.*

The two most classical techniques for localizing or separating roots are:

Analytical method

In this case, we rely on the Intermediate Value Theorem (IVT):

Theorem 2.1. *Let $[a, b] \subset \mathbb{R}$ and let f be a continuous function from $[a, b]$ to $\mathbb{R}$ such that $f(a)f(b) < 0$. Then there exists $\xi \in (a, b)$ such that $f(\xi) = 0$.*

Example 2.1. *Let us determine the roots of the function $f(x) = x^4 - 4x - 1$. The variations of f are given in the following table:*

x	$-\infty$		1		$+\infty$
$f'(x)$		$-$	0	$+$	
$f(x)$	$+\infty$	$\searrow$	-4	$\nearrow$	$+\infty$

According to the table of variations, the function f is strictly monotonic on the intervals $[-1, 0] \cup [1, 2]$, with $f(-1) \cdot f(0) < 0$ and $f(1) \cdot f(2) < 0$. Therefore, there are two roots: $\xi_1 \in (-1, 0)$ and $\xi_2 \in (1, 2)$.

Graphical method (geometric)

Let us trace (experimentally or by studying the variations of f) the graph of the function f and look for its intersection with the Ox-axis. Alternatively, we can decompose f into two functions f_1 and f_2 that are easier to study, such that $f = f_1 - f_2$, and we search for the points of intersection of the graphs of f_1 and f_2, whose x-coordinates are exactly the roots of the equation $f(x) = 0$.

Remark 2.1. *The functions f_1 and f_2 are often chosen with well-known graphs.*

Example 2.2. *Consider the equation*

$$x \log x = 1, \quad x > 0. \tag{2.1}$$

This equation can also be written as: $\log x = \frac{1}{x}$. *Let us define* $f_1(x) = \log x$, $f_2(x) = \frac{1}{x}$, *and* $f(x) = f_1(x) - f_2(x) = \log x - \frac{1}{x}$. *The variations of the functions f_1 and f_2 are given by the curves below (Figure 2.1). The x-coordinate of the point of intersection of the two curves allows us to localize the solution of the equation* (2.1) *and even provides a (first) approximation of it.*

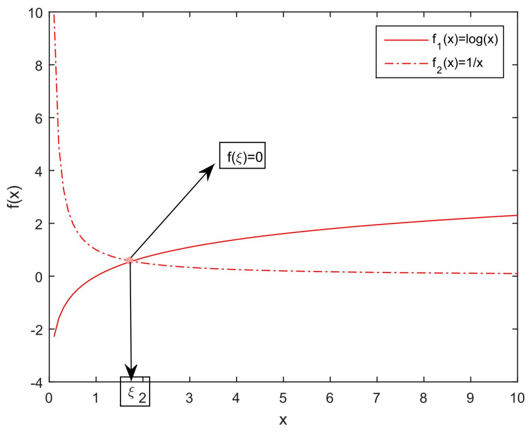


Figure 2.1: Graphical separation of the root.

2.2 Numerical methods

2.2.1 Bisection Method

The Bisection method (or dichotomy method) assumes that the function f is continuous on an interval $[a, b]$, has only one root $\xi \in (a, b)$, and satisfies $f(a)f(b) < 0$.

The principle is as follows: we set $a_0 = a$, $b_0 = b$, and define $x_0 = \frac{(a_0+b_0)}{2}$ as the midpoint of the initial interval and evaluate the function f at this point. If $f(x_0) = 0$, the point x_0 is the root of f, and the problem is solved. Otherwise, if $f(a_0)f(x_0) < 0$, the root ξ is contained within the interval (a_0, x_0), while it belongs to (x_0, b_0) if $f(x_0)f(b_0) < 0$. This process is then repeated on the new interval $[a_1, b_1]$, with $a_1 = a_0$ and $b_1 = x_0$ in the first case, or $a_1 = x_0$ and $b_1 = b_0$ in the second, and so on. In this way, we recursively construct three sequences $\{a_n\}_{n\in\mathbb{N}}$, $\{b_n\}_{n\in\mathbb{N}}$, and $\{x_n\}_{n\in\mathbb{N}}$ with:

- $x_n = \frac{a_n+b_n}{2}$
- $a_{n+1} = a_n$ and $b_{n+1} = x_n$ if $f(a_n)f(x_n) < 0$

- $a_{n+1} = x_n$ and $b_{n+1} = b_n$ if $f(x_n)f(b_n) < 0$

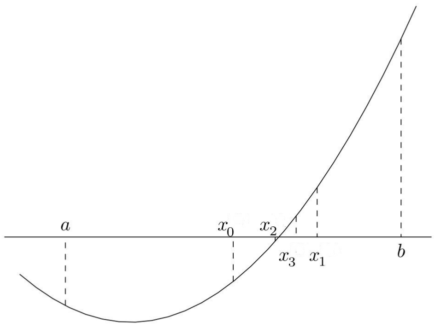


Figure 2.2: Construction of the first three iterations of the method.

Proposition 2.1. *Let f be a continuous function on the interval $[a, b]$ satisfying $f(a)f(b) < 0$. Let $\xi \in (a, b)$ be the unique solution of the equation $f(x) = 0$. Then, the sequence $\{x_n\}_{n\in\mathbb{N}}$ generated by the method converges to ξ with a precision given by*

$$|x_n - \xi| \leq \frac{b-a}{2^{n+1}}, \quad \forall n \in \mathbb{N}.$$

Remark 2.2. *From this inequality, if the precision ε is known, the required number of iterations n can be calculated. Indeed:*

$$\frac{b-a}{2^{n+1}} \leq \varepsilon \Longrightarrow n \geq \frac{\ln\left(\frac{b-a}{2\varepsilon}\right)}{\ln 2}$$

Example 2.3. *Apply the method to calculate the root of the equation $x^3 + 4x^2 - 10 = 0$ with a precision $\varepsilon = 10^{-2}$.*
The table of variations of f is as follows:

x	$-\infty$		$-8/3$		0		$+\infty$
$f'(x)$		$+$	0	$-$	0	$+$	
$f(x)$	$-\infty$	$\nearrow$	-	$\searrow$	-10	$\nearrow$	$+\infty$

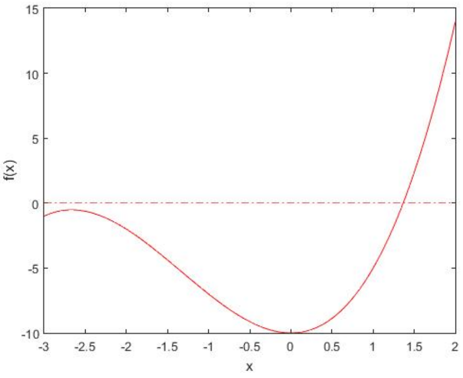


Figure 2.3: Graph of f.

From the table of variation and Figure 2.3, it follows that $\exists!\xi \in (1,2)$ *such that* $f(\xi) = 0$*. Hence the required number of iterations to reach a precision of* 10^{-2} *is*

$$n \geq \frac{\ln\left(\frac{2-1}{2\times 10^{-2}}\right)}{\ln 2} \simeq 5.64$$

Then $n = 6$ and the following table summarizes the evaluated points.

n	a_n	b_n	x_n	$f(x_n)$	*sign:* $f(a_n).f(x_n)$	$\delta_n = \frac{b-a}{2^{n+1}}$
0	1	2	1.5	2.375	-	0.5
1	1	1.5	-1.25	-1.789	+	0.25
2	1.25	1.5	1.375	0.1621	-	0.125
3	1.25	1.375	1.3125	-0.848	+	0.0625
4	1.3125	1.375	1.3437	-0.3509	-	0.03125
5	1.3437	1.375	1.3593	-0.0964	+	0.015625
6	1.35937	1.375	1.36718	0.0322	+	0.0078125

Example 2.4. *Let's calculate the first root of the equation* $\ln(x) - x^2 + 2 = 0$ *that lies in the interval* $[0.1, 0.5]$ *with a precision of* $\varepsilon = 0.01$. *First, we calculate the number of subdivisions* n *to perform:*

$$n \geq \frac{\ln\left(\frac{0.5-0.1}{2\times10^{-2}}\right)}{\ln 2} \simeq 4.32 \Longrightarrow n = 5.$$

The following table summarizes the steps of the method.

n	a_n	b_n	x_n	$f(x_n)$	*sign:* $f(a_n).f(x_n)$	$\delta_n = \frac{b-a}{2^{n+1}}$
0	0.1	0.5	0.3	0.706	-	0.2
1	0.1	0.3	0.2	0.351	-	0.1
2	0.1	0.2	0.15	0.08	-	0.05
3	0.1	0.15	0.125	-0.095	+	0.025
4	0.125	0.15	0.1375	-0.030	+	0.0125
5	0.1375	0.15	0.14375	0.0393	-	0.0062

2.2.2 Lagrange Method

Lagrange method, also known as the method of false position, is a technique for finding an approximate value of the solution of an equation $f(x) = 0$. It can be described as follows: suppose that the function f is continuous on $[a, b]$ with $f(a)f(b) < 0$. Consider the points $A(a, f(a))$ and $B(b, f(b))$ located on the curve $\mathcal{C}_f$ of f. We construct a sequence $\{x_n\}_{n\in\mathbb{N}}$ of real numbers using the points A_n on $\mathcal{C}_f$. To do this, we set $A_0 = A$ and construct A_{n+1} by drawing the line (A_nB) that intersects the x-axis at a point with abscissa x_{n+1}. The point A_{n+1}

is the point on $\mathcal{C}_f$ with abscissa x_{n+1}.
The iterative scheme of Lagrange algorithm is given as follows: Choose a starting point x_0 that satisfies the condition $f(x_0)f''(x_0) < 0$ and for each iteration we set

$$\begin{cases} -\text{Select } x_0 = a \text{ if } f(a)f''(a) < 0 \\ -\text{Set } x_{n+1} = x_n - f(x_n)\frac{x_n - b}{f(x_n) - f(b)}. \end{cases}$$

and

$$\begin{cases} -\text{Select } x_0 = b \text{ if } f(b)f''(b) < 0 \\ -\text{Set } x_{n+1} = x_n - f(x_n)\frac{x_n - a}{f(x_n) - f(a)}. \end{cases}$$

Exemple. Consider the equation $f(x) = x^3 - 20 = 0$. Since the function f is contentiously decreasing on the interval $[0.75, 4.5]$ with $f(0.75)f(4.5) < 0$, we can apply Lagrange method in the interval $[0.75, 4.5]$ by choosing $x_0 = 0.75$ as the starting point. The construction of the first iterates of Lagrange method is illustrated in Figure 2.4.

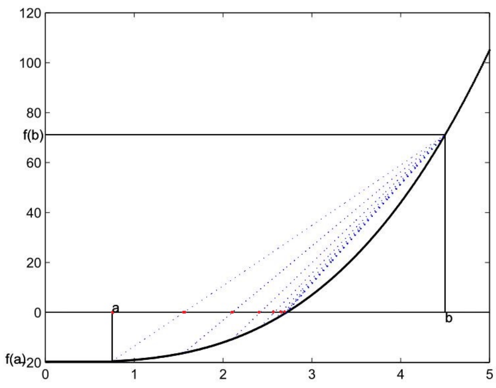


Figure 2.4: Construction of the iterates using Lagrange method

Proposition 2.2. *Let f be a continuous function on the interval $[a, b]$, satisfying $f(a)f(b) < 0$, and let $\xi \in (a, b)$ be the unique solution of the equation $f(x) = 0$. If $f \in \mathcal{C}^2([a, b])$ such that $\forall x \in [a, b]$, $f'(x)f''(x) \neq 0$, then the sequence $\{x_n\}_{n\in\mathbb{N}}$ constructed by Lagrange method converges to ξ with a precision given by*

$$|x_n - \xi| \leq \frac{M_1 - m_1}{m_1}|x_n - x_{n-1}|$$

where

$$M_1 = \max_{[a,b]}\{|f'(x)|\}, \quad m_1 = \min_{[a,b]}\{|f'(x)|\}.$$

Example 2.5. *Find the root of the function $f(x) = x^3 - x - 4$ in the interval $[1, 2]$ within a precision of $\varepsilon = 10^{-2}$, using the Lagrange method. We have*

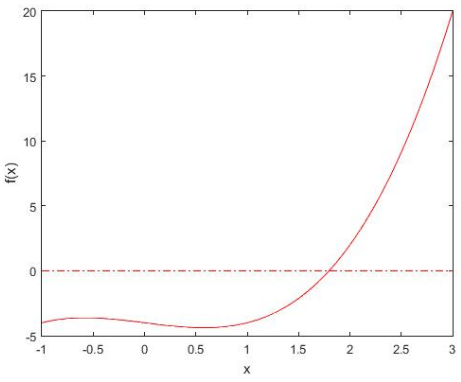


Figure 2.5: Graph of f

$$M_1 = \max_{[1,2]}\{|f'(x)|\} = |f'(2)| = 11$$

$$m_1 = \min_{[1,2]}\{|f'(x)|\} = |f'(1)| = 2,$$

and for all $x \in [1, 2]$, *we have*

$$f'(x) = 3x^2 - 1 > 0,$$
$$f''(x) = 6x > 0,$$

and $f(1)f''(1) < 0$, *so we take* $x_0 = 1$, *and for all* $n \in \mathbb{N}$,

$$x_{n+1} = x_n - f(x_n)\frac{x_n - 2}{f(x_n) - f(2)}.$$

By following the iterative scheme of the Lagrange algorithm, we obtain

$$* \, x_1 = x_0 - f(x_0)\frac{x_0 - 2}{f(x_0) - f(2)} = 1.666 \textit{ and } f(x_1) = -1,0368$$
$$|x_1 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_1 - x_0| = \frac{11 - 2}{2}|1,6667 - 1| = 3.$$
$$* \, x_2 = x_1 - f(x_1)\frac{x_1 - 2}{f(x_1) - f(2)} = 1.7805$$
$$|x_2 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_2 - x_1| = 0.05$$
$$* \, x_3 = x_2 - f(x_2)\frac{x_2 - 2}{f(x_2) - f(2)} = 1.7945$$
$$|x_3 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_3 - x_2| = 0.034$$
$$* \, x_4 = x_3 - f(x_3)\frac{x_3 - 2}{f(x_3) - f(2)} = 1.7961$$
$$|x_4 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_4 - x_3| = 0.009$$

Hence $\xi = 1.7961 \pm 0.009$

2.2.3 Newton-Raphson Method

This method is the most used for finding roots in one-dimensional problems. However, it requires the evaluation of $f(x)$ and $f'(x)$.

Let ξ be a unique root of the equation $f(x) = 0$ on the interval $[a, b]$, such that f is continuous and satisfies:

$$f'(x) \neq 0, \ \forall x \in [a, b], \tag{2.2}$$

$$f''(x) \neq 0, \ \forall x \in [a, b]. \tag{2.3}$$

The main idea of this method is to replace, at each iteration k, the arc of the curve of the function $y = f(x)$ on $[a, b]$ with the tangent to this arc at the point $(x_n, f(x_n))$: The abscissa x_{n+1} of the intersection of the tangent equation with the Ox-axis is an approximation of the unique solution ξ in $[a, b]$ for the equation $f(x) = 0$ (see Figure 2.6).

The equation of the tangent is:

$$y = f(x_n) + f'(x_n)(x - x_n),$$

which intersects the Ox-axis at the point $(x_{n+1}, 0)$, from which we get:

$$f(x_n) + f'(x_n)(x_{n+1} - x_n) = 0,$$

which gives the following iterative scheme (Newton-Raphson):

$$\begin{cases} -\text{Select a starting point } x_0 \in [a, b] \text{ with} f(x_0).f''(x_0) > 0. \\ -\text{Set } x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \end{cases}$$

Proposition 2.3. *Let f be a continuous function on the interval $[a, b]$, satisfying $f(a)f(b) < 0$, and let $\xi \in (a, b)$ be the unique solution of the equation $f(x) = 0$. If $f \in \mathcal{C}^2([a, b])$ such that for all $x \in [a, b]$, $f'(x) \cdot f''(x) \neq 0$, then the sequence $\{x_n\}_{n \in \mathbb{N}}$ constructed by the Newton-Raphson method converges to ξ with a precision given by:*

$$|x_n - \xi| \leq \frac{M_2}{2m_1}(x_n - x_{n-1})^2$$

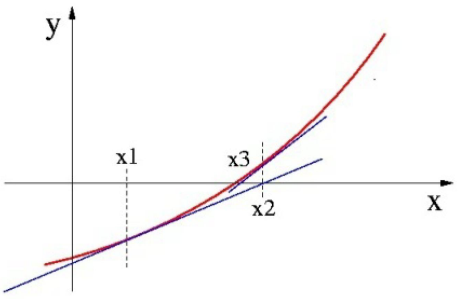


Figure 2.6: Construction of the first three iterates using Newton-Raphson method.

where

$$M_2 = \max_{[a,b]}\{|f''(x)|\}, \quad m_1 = \min_{[a,b]}\{|f'(x)|\}.$$

Example 2.6. *Let's calculate the root of the function* $f(x) = x^3 - x - 4$ *in* $[1,2]$*, within a precision of* 10^{-2}*, using the Newton-Raphson method. We have* $f(1)\cdot f(2) < 0$*, and for all* $x \in [1,2]$*,* $f'(x) = 3x^2 - 1 > 0$ *and* $f''(x) = 6x > 0$*. We have*

Applying the iterative scheme of the Newton-Raphson algorithm starting from $x_0 = 2$ *with* $f(2)\cdot f''(2) > 0$*, we get:*

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 2 - \frac{f(2)}{f'(2)} = 1.8181,$$
$$|\xi - x_1| \le \frac{M_2}{2m_1}(x_1 - x_0)^2 = \frac{12}{2\times 2}(1.818 - 2)^2 \approx 0.01,$$

where

$M_2 = \max_{[1,2]}\{|f''(x)|\} = f''(2) = 12$ *and* $m_1 = \min_{[1,2]}\{|f'(x)|\} = f'(1) = 2,$

Next, for x_2:

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 1.8181 - \frac{f(1.8181)}{f'(1.8181)} = 1.7966,$$
$$|\xi - x_2| \leq \frac{M_2}{2m_1}(x_2 - x_1)^2 = \frac{12}{4}(1.7966 - 1.818)^2 \approx 0.001.$$

Thus, $\xi = 1.7966 \pm 0.001.$

In some situations, the derivative f' can be quite complicated, or even impossible to calculate. In this case, we approximate the derivative of f using a rate of change. This method is called the **secant method**:

$$\begin{cases} -\text{Select } x_0, x_1 \in [a, b] \text{ close to } \xi. \\ -\text{Set } x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}. \end{cases}$$

Here, x_{n+1} depends on both x_n and x_{n-1}: we say that it is a **two-step method**; indeed, we need two initial iterates, x_0 and x_1. The advantage of this method is that it does not require the calculation of the derivative f'. The drawback is that the convergence is no longer as fast.

2.3 Matlab codes

2.3.1 Pseudo-code of Bisection method

```
clc;clear all;close all
f=@(x) x^3-x-2; % Example: f(x)=x^3-x-2
% Define the interval [a,b]
a=1; % Left endpoint
b=2; % Right endpoint
tol=1e-6; % Desired accuracy
max_iter=100; % Maximum number of
   iterations
```

```
% Check if the function has opposite signs
    at the endpoints
if f(a)*f(b)>= 0
    error('The function must have opposite
        signs at a and b.');
end
% Initialize variables
iter=0; % Iteration counter
fprintf('Iter\t a\t\t b\t\t c\t\t f(c)\n')
  ;
while (b-a)/2>tol && iter<max_iter
    iter=iter+1;% Increment iteration
      counter
    c=(a+b)/2;% Compute the midpoint
    % Display current step
    fprintf('%d\t%.6f\t%.6f\t%.6f\t%.6f\n'
      ,iter,a,b,c,f(c));

    % Check if the root is found or narrow
        the interval
    if f(c)==0
        break; % c is the root
    elseif f(c)*f(a)<0
        b=c; % Root is in the left
           subinterval
    else
        a=c; % Root is in the right
           subinterval
    end
end
% Output the result
root=(a+b)/2;
```

```
fprintf('The root using Bissection method
   is approximately: %.6f\n',root);
fprintf('Number of iterations: %d\n',iter)
   ;
%
```

2.3.2 Pseudo-code of Lagrange method

```
clc;clear all;close all
f=@(x) x^3-x-2; % Example: f(x)=x^3-x-2
ddf=@(x)6*x;
% Define the interval [a,b]
a=1; % Left endpoint
b=2; % Right endpoint
max_iter=100; % Maximum number of
   iterations
tol=1e-6; %Required tolerance
% Check if the function has opposite signs
    at the endpoints
if f(a)*f(b)>= 0
    error('The function must have opposite
        signs at a and b.');
end
iter=0; % Iteration counter
%Select the starting point
if f(a)*ddf(a)<0
    x0=a;
    x_end=b;
else
    x0=b;
    x_end=a;
end
x=x0;
```

```
while iter < max_iter
    if abs(f(x))<tol
        root=x;
       break;
    end
    % Lagrange iteration formula
    x_new=x-f(x)*(x-x_end)/(f(x)-f(x_end))
       ;
    % Check for convergence
    if abs(x_new-x)<tol
        root=x_new;
        break
    end
    x=x_new;
    iter=iter+1;
end
root=x; % Return the final approximation
   if max_iter is reached
fprintf('The root using Lagrange method is
    approximately: %.6f\n',root)
fprintf('Number of iterations: %d\n',iter)
fprintf('Approximate root: %.6f\n', root)
```

2.3.3 Pseudo-code of Newton-Raphson method

```
clc;clear all;close all
f=@(x) x^3-x-2; % Example: f(x)=x^3-x-2
df=@(x)3*x^2-1;
ddf=@(x)6*x;
% Define the interval [a,b]
a=1; % Left endpoint
```

```
b=2; % Right endpoint
max_iter=100; % Maximum number of
   iterations
tol=1e-6; %Required tolerance
% Check if the function has opposite signs
    at the endpoints
if f(a)*f(b)>= 0
    error('The function must have opposite
        signs at a and b.');
end
iter=0; % Iteration counter
%Select the starting point
if f(a)*ddf(a)>0
    x0=a;
else
    x0=b;
end
x=x0;
while iter < max_iter
    if abs(f(x))<tol
        root=x;
       break;
    end
    % Lagrange iteration formula
    x_new=x-f(x)/df(x);
    % Check for convergence
    if abs(x_new-x)<tol
        root=x_new;
        break
    end
    x=x_new;
    iter=iter+1;
end
```

```
root=x; % Return the final approximation
   if max_iter is reached
fprintf('The root using Newton-Raphson
   method is approximately: %.6f\n',root)
fprintf('Number of iterations: %d\n',iter)
fprintf('Approximate root: %.6f\n', root)
```

2.4 Solved Exercises

Exercise 2.1. *Using the Newton-Raphson algorithm, find the square root of* 2 *in the interval* $[1,2]$ *with a precision of* $\varepsilon = 10^{-3}$, *using* $x_0 = 2$ *as a starting starting point.*

Solution

We seek the square root of 2 in the interval $[1,2]$, i.e., we find the root of the following equation

$$x^2 = 2 \Rightarrow f(x) = x^2 - 2 = 0.$$

For all $k \in \mathbb{N}$, we apply the Newton-Raphson iterative scheme :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},\ k = 0, 1, 2, \ldots$$

We have $f'(x) = 2x$ and $f''(x) = 2 > 0$, so

$$M_2 = \max_{[1,2]} |f''(x)| = 2 \quad \text{and} \quad m_1 = \min_{[1,2]} |f'(x)| = f'(1) = 2.$$

$$k = 1: \; x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 2 - \frac{f(2)}{f'(2)} = 1.5$$

$$|\xi - x_1| \leq \frac{M_2}{2m_1}(x_1 - x_0)^2 = \frac{2}{2 \times 2}(2 - 1.5)^2 = 0.125$$

$$k = 2: \; x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 1.5 - \frac{f(1.5)}{f'(1.5)} = 1.416$$

$$|\xi - x_2| \leq \frac{M_2}{2m_1}(x_2 - x_1)^2 = \frac{2}{2 \times 2}(1.5 - 1.416)^2 = 0.0035$$

$$k = 3: \; x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 1.416 - \frac{f(1.416)}{f'(1.416)} = 1.414$$

$$|\xi - x_3| \leq \frac{M_2}{2m_1}(x_3 - x_2)^2 = \frac{2}{2 \times 2}(1.416 - 1.414)^2 = 2 \times 10^{-6}$$

Thus, $x^* \approx 1.414 \pm 2 \times 10^{-6}$ is the approximated square root of 2.

Exercise 2.2.

a- Give the iterative scheme of the Newton-Raphson algorithm to solve a nonlinear equation $f(x) = 0$.

b- Using the Newton-Raphson algorithm, determine the root in the interval $[0, 1]$ *of the equation* $x^2 = e^{-2x}$ *with a precision of* 10^{-3}*, starting with an initial point* $x_0 = 1$.

Solution

b- Let's determine the root in the interval $[0, 1]$ of the equation $x^2 = e^{-2x}$ with a precision of 10^{-3} using the Newton-Raphson algorithm. We have

$$f'(x) = 2x + 2e^{-2x} \quad \text{and} \quad f''(x) = 2 - 4e^{-2x},$$

with

$$M_2 = \max_{[0,1]} |f''(x)| = f''(1) = 1.45 \quad \text{and} \quad m_1 = \min_{[0,1]} |f'(x)| = f'(0.346) = 1.69.$$

Proceeding as in the previous exercise, we get:

$$
\begin{aligned}
k = 1 : x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} &= 1 - \frac{f(1)}{f'(1)} = 0.6192 \\
|\xi - x_1| &\leq \frac{M_2}{2m_1}(x_1 - x_0)^2 = 0.0624 \\
k = 2 : x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} &= 0.6192 - \frac{f(0.6192)}{f'(0.6192)} = 0.5677 \\
|\xi - x_2| &\leq \frac{M_2}{2m_1}(x_2 - x_1)^2 = 0.0011 \\
k = 3 : x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} &= 0.5677 - \frac{f(0.5677)}{f'(0.5677)} = 0.5671 \\
|\xi - x_3| &\leq \frac{M_2}{2m_1}(x_3 - x_2)^2 = 1.55 \times 10^{-7} < \varepsilon
\end{aligned}
$$

Thus, $x^* \approx 0.5671 \pm 1.55 \times 10^{-7}$ is the approximated root.

Exercise 2.3. *Consider the equation* $f(x) = 2\tan(x) - x - 1 = 0$ *with* $x \in [-\pi, \pi]$.
a- Separate analytically the roots of this equation.
b- Calculate the number n *of required iterations to approximate this root with a precision of* 10^{-3} *using the method.*

Solution

a- We have $f(x) = 2\tan(x) - x - 1$, and $f'(x) = \frac{2}{\cos(x)^2} - 1$. The table of variations of f is given as follows:

x	$-\pi$		$-\pi/2$		$\pi/2$		π
$f'(x)$		$+$	$\|\|$	$+$	$\|\|$	$+$	
$f(x)$	2.14	$\nearrow$	$+\infty$ $\|\|$ $-\infty$	$\nearrow$	$+\infty$ $\|\|$ $-\infty$	$\nearrow$	-4.14

Thus, according to this table, there exists a single root in the interval $]-\frac{\pi}{2},\frac{\pi}{2}[$.

b- Let's calculate the required number of iterations n:

$$\begin{aligned} n &\geq \frac{\ln\left(\frac{b-a}{2\varepsilon}\right)}{\ln 2} \\ &\geq \frac{\ln\left(\frac{\pi}{2\times10^{-3}}\right)}{\ln 2} \simeq 10.6173 \end{aligned}$$

Therefore, to reach the root with a precision of 2×10^{-3}, we need at least $n\geq 11$.

Exercise 2.4. ***a***. *Approximate the smallest root of the function* $f(x)=x^4-2x-4$ *with a precision of* 5×10^{-3} *using Newton-Raphson and Lagrange methods.*
b. *Compare the two methods and draw a conclusion.*

Solution

According to Figure 2.7, this function $f(x)$ has two roots; let's find the negative root located in the interval $[-2,-1]$.

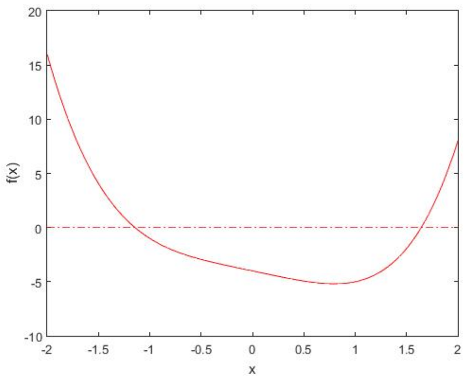


Figure 2.7: Graph of f.

Approximating using Newton-Raphson method:

We have

$$f'(x) = 4x^3 - 2 < 0, \forall x \in [-2,-1],$$
$$f''(x) = 12x^2 > 0, \forall x \in [-2,-1].$$

and

$$M_2 = \max_{[-2,-1]} \{|f''(x)|\} = |f''(-2)| = 48$$
$$m_1 = \min_{[-2,-1]} \{|f'(x)|\} = |f'(-1)| = 6$$

Since $f(-2).f''(-2) > 0$, we take $x_0 = -2$ as a starting point, and for all $k \in \mathbb{N}$, we set

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, k = 0, 1, 2, \ldots.$$

Following the iterative scheme of the Newton-Raphson algorithm, we obtain:

$$k = 1 : x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = -2 - \frac{16}{-34} = -1.53$$
$$|\xi - x_1| \leq \frac{M_2}{2m_1}(x_1 - x_0)^2 = \frac{48}{2 \times 6}(-1.53 + 2)^2 = 0.88$$

$$k = 2 : x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = -1.53 - \frac{-4.53}{-16.32} = -1.25$$

$$|\xi - x_2| \leq \frac{M_2}{2m_1}(x_2 - x_1)^2 = 0.31$$

$$k = 3 : x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = -1.25 - \frac{0.94}{-9.81} = -1.1542$$

$$|\xi - x_3| \leq \frac{M_2}{2m_1}(x_3 - x_2)^2 = 0.03$$

$$k = 4 : x_4 = x_3 - \frac{f(x_3)}{f'(x_3)} = -1.1542 - \frac{0.083}{-8.15} = -1.144$$

$$|\xi - x_4| \leq \frac{M_2}{2m_1}(x_4 - x_3)^2 = 0.004$$

Hence $\xi = -1.144 \pm 0.004$ is the prescribed solution.

Approximating using Lagrange method:

Since $f(-1).f''(-1) < 0$, we take $x_0 = -1$ as initial point,and for all $n \in \mathbb{N}$, we set

$$x_{n+1} = x_n - f(x_n)\frac{x_n + 2}{f(x_n) - f(-2)}$$

with $M_1 = \max\limits_{[-2,-1]}\{|f'(x)|\} = |f'(-2)| = 34$ and $m_1 = \min\limits_{[-2,-1]}\{|f'(x)|\} = |f'(-1)| = 6$

$$k = 1 : x_1 = x_0 - f(x_0)\frac{x_0 + 2}{f(x_0) - f(-2)} = -1.05$$

$$|x_1 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_1 - x_0| = 0.274$$

$$k = 2 : x_2 = x_1 - f(x_1)\frac{x_1 + 2}{f(x_1) - f(-2)} = -1.0941$$

$$|x_2 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_2 - x_1| = 0.164$$

$$k = 3 : * x_3 = x_2 - f(x_2)\frac{x_2 + 2}{f(x_2) - f(-2)} = -1.1149$$

$$|x_3 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_3 - x_2| = 0.097$$

$$k = 4 : x_4 = x_3 - f(x_3)\frac{x_3 + 2}{f(x_3) - f(-2)} = -1.127$$

$$|x_4 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_4 - x_3| = 0.0564$$

$$k = 5 : x_5 = x_4 - f(x_4)\frac{x_4 + 2}{f(x_4) - f(-2)} = -1.1341$$

$$|x_5 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_5 - x_4| = 0.033$$

$$k = 6 : x_6 = x_5 - f(x_5)\frac{x_5 + 2}{f(x_5) - f(-2)} = -1.1382$$

$$|x_6 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_6 - x_5| = 0.0191$$

$$k = 7 : x_7 = x_6 - f(x_6)\frac{x_6 + 2}{f(x_6) - f(-2)} = -1.1406$$

$$|x_7 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_7 - x_6| = 0.011$$

$$k = 8 : x_8 = x_7 - f(x_7)\frac{x_7 + 2}{f(x_7) - f(-2)} = -1.1419$$

$$|x_8 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_5 - x_4| = 0.006$$

$$k = 9 : x_9 = x_8 - f(x_8)\frac{x_8 + 2}{f(x_8) - f(-2)} = -1.14275$$

$$|x_9 - \xi| \leq \frac{M_1 - m_1}{m_1}|x_9 - x_8| = 0.004$$

Hence$\xi = -1.14275 \pm 0.004$.

To achieve a precision of 5×10^{-3}, it would require 9 iterations using the Lagrange method, whereas the Newton method requires only 4 iterations. The Newton method converges faster than the Lagrange method.

Exercise 2.5. *We consider the equation $f(x) = 0$, with $f(x) = \ln(x) - x + 2$.*
1.a. Write the equation $f(x) = 0$ in the form $f_1(x) = f_2(x)$ with $f_1(x) = \ln(x)$.
b. Plot the graphs of f_1 and f_2. What can be said about this equation?
2.a. Perform 4 iterations of the method to approximate the solution in the interval $[3, 4]$. At which iteration we obtain the best result? Justify and conclude.
b. Determine the number of n of required iterations to achieve a precision of 10^{-4}.
c. Give an estimate of the error after 25 iterations.
3. Approximate the root with a precision of 10^{-4} using the Newton-Raphson method, starting from $x_0 = 3$.
4. Compare the two methods and draw a conclusion.

Solution

1-a.

$$\begin{aligned} f(x) = 0 &\Leftrightarrow \ln(x) - x + 2 = 0 \\ &\Leftrightarrow \ln(x) = x - 2 \\ &\Leftrightarrow f_1(x) = f_2(x) \text{ avec } f_1(x) = \ln(x) \text{ et } f_2(x) = x - 2 \end{aligned}$$

1-b. According to Figure 2.8, the graphs of f_1 and f_2 have two intersection points, so this equation has two roots $\xi_1 \in]0, 1[$ and $\xi_2 \in]3, 4[$.
2-a. Proceeding as in examples 1 and 2, we obtain $x_1 = 3.5$, $x_2 = 3.25$, $x_3 = 3.125$, and $x_4 = 3.1875$, with x_3 the best result obtained since $f(x_3) \min\{f(x_i), i = 1, 2, 3, 4\}$. We conclude that, although the convergence of the method towards the root is guaranteed, it is not monotonic.

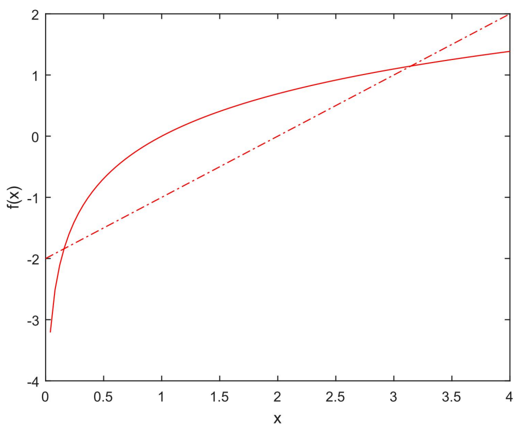


Figure 2.8: Graphical separation of the roots.

2-b.

$$\begin{aligned} n &\geq \frac{\ln(\frac{b-a}{2\varepsilon})}{\ln 2} \\ &\geq \frac{\ln(\frac{1}{10^{-4}})}{\ln 2} \simeq 13.29\,, \end{aligned}$$

hence $n \geq 14$.

2-c

$$\begin{aligned} |x_n - \xi| &\leq \frac{b-a}{2^{n+1}} \\ &= \frac{4-3}{2^{26}} = 1.4901 \times 10^{-8} \end{aligned}$$

3. By applying the Newton algorithm starting from the point $x_0 = 3$, and after 3 iterations, the algorithm reaches the root within 10^{-4}. The generated points are $x_1 = 3.1479$, $x_2 = 3.1462$, and $x_3 = 3.1462$.
4. To achieve a precision of 10^{-4}, it would require 14 iterations using the method, whereas the Newton method only requires 3 iterations. The Newton-Raphson method converges quicker than the method.

2.4.1 Suplementary Exercicises

Exercise 2.6. *Consider the function $f(x) = x^2 - \xi$ defined on $\mathbb{R}$, with $\xi \in \mathbb{R}_+$.*

1- *Prove that the iterative scheme of the Newton-Raphson method to find a root of f can be given by the following form:*

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{\xi}{x_k}\right), \quad k \in \mathbb{N}. \tag{2.4}$$

2- *Let $\xi = 5$.*

i) *Determine an interval of the form $[a, a+1]$, with $a \in \mathbb{N}$, where we can approximate a value of $\sqrt{\xi}$.*

ii) *Provide an approximation of the value $\sqrt{\xi}$ using the recurrent formula (2.4), choosing $x_0 = a+1$ as the initial point and using the stopping criterion $\Delta_k = \frac{M}{2m}(x_k - x_{k-1})^2 \leq \varepsilon$, with $M = \max\limits_{[a,a+1]} |f''(x)|$, $m = \min\limits_{[a,a+1]} |f'(x)|$, and $\varepsilon = 10^{-4}$ (where k is the number of iterations).*

3- *Determine the number of iterations n required by the Bisection method to obtain an approximate solution with the same precision $\varepsilon = 10^{-4}$.*

Exercise 2.7. *Let f be a function defined on $]0, +\infty[$ by $f(x) = e^{-x} - \ln(x)$.*

1- *Prove that there exists a unique root ξ of the function f, located between two consecutive integers $\mathbf{a}$ and $\mathbf{a+1}$.*

2- *Calculate the number n of required iterations to reach an error less than 10^{-4} using Bisection method.*

3- *Give the iterative scheme of the Newton-Raphson algorithm.*

4- *Approximate the value of* ξ *with a precision of* 10^{-4} *using the Newton-Raphson algorithm, justifying the choice of the initial point* x_0 *between* $\mathbf{a}$ *and* $\mathbf{a+1}$.

Exercise 2.8. *Let* f *be a function defined on* $\mathbb{R}$ *by* $f(x) = e^{\frac{x}{4}} - 3/2$.

1- *Prove that there exists a unique root* ξ *of the function* f, *located between two consecutive integers* $\mathbf{a}$ *and* $\mathbf{a+1}$.

2- *Calculate the number* n *of required iterations to reach an error less than* 10^{-4} *using Bisection method.*

3- *Approximate the value of* ξ *with a precision of* 10^{-4} *using the Newton-Raphson algorithm, justifying the choice of the initial point* x_0 *between* $\mathbf{a}$ *and* $\mathbf{a+1}$.

3
Numerical Solution of Ordinary Differential Equations of First Order

Many challenges in science and engineering can be reduced to the task of solving differential equations while satisfying certain predefined conditions. Traditional analytical techniques, which are assumed to be understood by the reader, are suitable for solving only a subset of these equations. However, the differential equations that govern the behavior of physical systems often lack closed-form solutions. Therefore, it is crucial to use numerical methods to solve these problems.

Definition 3.1. *An ordinary differential equation* (ODE) *of order* $n, n \in \mathbb{N}^*$ *is any relation of the type*

$$f(t, y(t), y'(t), \ldots, y^{(n)}(t)) = 0 \tag{3.1}$$

which we write in the canonical form as

$$y^{(n)}(t) = f(t, y(t), y'(t), \ldots, y^{(n-1)}(t)) \tag{3.2}$$

where y is a function of the variable t, and for $i = 1, \dots, n$, $y^{(i)}$ is the derivative of y with respect to t of order i.

The general solution of equations (3.1) and (3.2) is given by a relation between t and y with a number of constants (equal to the degree of the equation). This relation can be implicit:

$$W(t, y(t), c_1, \dots, c_n) = 0$$

or explicit

$$y(t) = V(t, c_1, \dots, c_n).$$

To determine the constants $c_i, i = 1, \dots, n$, we need initial or boundary conditions on y.

Definition 3.2. *A differential equation is said to be of order 1 if it is of the form: $y'(t) = f(t, y(t))$ with $t \in [a, b]$ and f a function is defined on $[a, b] \times \mathbb{R} \to \mathbb{R}$.*

In this chapter, we present some numerical methods that aim to approximate solutions of ordinary differential equations of first order.

3.1 The Basic Principles of Initial-Value Problems

Definitions and results from ordinary differential equations theory are necessary before delving into methods for approximating solutions of initial-value problems.

3.1.1 Cauchy problem

The goal is to find a differentiable function $y(t) : I = [a, b] \to \mathbb{R}$ such that

$$(P) : \begin{cases} y'(t) = f(t, y(t)), t \in I \\ y(t_0) = y_0 \text{ (Initial condition)} \end{cases}$$

3.1.2 Existence and uniqueness of the solution

Theorem 3.1. *If $f(t,y)$ is a continuous function on $I\times\mathbb{R}$, then the problem (P) admits a solution. The uniqueness of the solution is guaranteed under one of the following conditions:*

a- $f(t,y)$ satisfies the Lipschitz condition with respect to y, i.e.,

$$\exists L>0,\forall y_1,y_2\in\mathbb{R}:|f(t,y_1)-f(t,y_2)|\leq L|y_1-y_2|$$

b- The partial derivative $\frac{\partial f}{\partial y}(t,y)$ is continuous and bounded on $I\times\mathbb{R}$.

Example 3.1.

$$(P_1):\begin{cases} y'(t)=\frac{-y}{t\ln t}+\frac{1}{\ln t}, & t\in[e,5]\\ y(e)=e\end{cases}$$

We have $f(t,y)=\frac{-y}{t\ln t}+\frac{1}{\ln t}$, which is continuous, and $|\frac{\partial f}{\partial y}(t,y(t))|=|\frac{-1}{t\ln t}|\leq\frac{1}{e}$, so $\frac{\partial f}{\partial y}$ is continuous and bounded on $[e,5]\times\mathbb{R}$. Therefore, the problem (P_1) admits a unique solution $y(t)=\frac{t}{\ln t}$.

Example 3.2.

$$(P_2):\begin{cases} y'(t)=1+t\sin(ty(t)), & t\in[0,2]\\ y(0)=0\end{cases}$$

We have $f(t,y)=1+t\sin(ty(t))$, which is continuous, and $\frac{\partial f}{\partial y}(t,y(t))=t^2\cos(ty(t))\leq t^2\leq 4$, so $\frac{\partial f}{\partial y}$ is bounded. Therefore, the problem (P_2) admits a unique solution.

Example 3.3. *Consider the following IVP:*

$$(P_3):\begin{cases} y'(t)=-\frac{y}{t\ln t}+\frac{1}{\ln t}, & t\in[e,5]\\ y(e)=e\end{cases}$$

The function $f(t,y) = -\frac{y}{t\ln t} + \frac{1}{\ln t}$ is continuous on $D = [e,5] \times \mathbb{R}$. Additionally, we have:

$$\left|\frac{\partial f}{\partial y}(t,y)\right| = \left|\frac{-1}{t\ln t}\right| \leq \frac{1}{e}.$$

Therefore, the problem (P_3) possesses a unique solution $y(t) = \frac{t}{ln(t)}$.

Example 3.4. *Consider the following IVP:*

$$(P_4) : \begin{cases} y' = 1 + t\sin(ty), & t \in [0,2] \\ y(0) = 0. \end{cases}$$

The function $f(t,y) = 1 + t\sin(ty)$ is continuous on $D = [0,2] \times \mathbb{R}$,

$$\frac{\partial f}{\partial y}(t,y) = t^2\cos(ty) \leq 4.$$

Hence, the problem (P_4) has a unique solution.

3.2 Picard's Method of Succssive Approximations

Upon integrating the an IVP, we integrate the following integral equation

$$y = y_0 + \int_{t_0}^{t} f(t,y)dt. \tag{3.3}$$

Equation (3.3), wherein the unknown function y appears under the integral sign, is termed an integral equation. Such an equation can be resolved through the method of successive approximations, wherein the initial approximation to y is acquired by substituting y_0 for y on the right side of Eq. (3.3). Thus, we express:

$$y_1 = y_0 + \int_{t_0}^{t} f(t,y_0)\,dt.$$

The integral on the right can now be evaluated, yielding y_1, which is then substituted for y in the integrand of Eq. (3.3), resulting in the second approximation y_2:

$$y^{(2)} = y_0 + \int_{t_0}^{t} f\left(t, y^{(1)}\right) dt.$$

Continuing iteratively, we obtain $y_3, y_4, \ldots, y_{n-1}$ and y_n, where

$$y_n = y_0 + \int_{t_0}^{t} f\left(t, y_{n-1}\right) dt. \tag{3.4}$$

Thus, this method provides a sequence of approximations $y_1, y_2, \ldots, y_n$, and it can be demonstrated that if the function $f(t, y)$ remains bounded in a certain region around the point (t_0, y_0) and if $f(t, y)$ satisfies the Lipschitz condition, then the sequence $y_1, y_2, \ldots$ converges to the solution of Problem (P).

Example 3.5. *Consider the following IVP*

$$y' = t + y^2, \quad t \in [0, +\infty[, \quad y(0) = 1.$$

Starting with $y_0 = 1$*, the first approximation is*

$$y^{(1)} = 1 + \int_0^t (t+1) dt = 1 + t + \frac{1}{2}t^2.$$

The second approximation is

$$\begin{aligned} y^{(2)} &= 1 + \int_0^t \left[t + \left(1 + t + \frac{1}{2}t^2\right)^2 \right] dt \\ &= 1 + t + \frac{3}{2}t^2 + \frac{2}{3}x^3 + \frac{1}{4}t^4 + \frac{1}{20}t^5. \end{aligned}$$

It is evident that as we proceed to higher approximations, the integrations may become increasingly challenging.

3.3 Numerical Methods

3.3.1 Euler Method

The Euler method is the simplest numerical method that allows approximating a solution of first-order ordinary differential equations with initial conditions. To numerically solve the Cauchy problem (P), we begin by partitioning the interval $I = [a, b]$, i.e., we choose points $t_0, t_1, \dots, t_n$ such that $a = t_0 < t_1 < \cdots < t_n = b$, with $t_{i+1} = t_i + h, h = \frac{b-a}{n}$ (the step size) with n is the number of evaluated points. The tangent to the curve $y = y(t)$ at $t = t_0$ has the equation:

$$\tilde{y}(t) = y(t_0) + (t - t_0)y'(t_0)$$

where

$$\tilde{y}(t) = y(t_0) + (t - t_0)f(t_0, y(t_0)).$$

At the point $t = t_1$, we get (see Figure 3.1):

$$y(t_1) \simeq \tilde{y}(t_1) = y(t_0) + (t_1 - t_0)f(t_0, y(t_0)),$$

by setting $h = t_1 - t_0$, it become

$$y(t_1) \simeq \tilde{y}(t_1) = y(t_0) + hf(t_0, y(t_0)).$$

Let $y_0 = \tilde{y}(t_0), y_1 = \tilde{y}(t_1)$, and then repeat the same procedure in the interval $[t_1, t_2]$, we obtain:

$$y(t_2) \simeq y_2 = y_1 + hf(t_1, y_1).$$

Thus, continuing in this way, we construct the following Euler algorithm:

$$\begin{cases} y_0 = y(t_0), t_0 = a \\ y_{i+1} = y_i + hf(t_i, y_i), i = 1, \dots, n-1 \end{cases}$$

where $h = \frac{b-a}{n}$, and $t_{i+1} = t_i + h$.

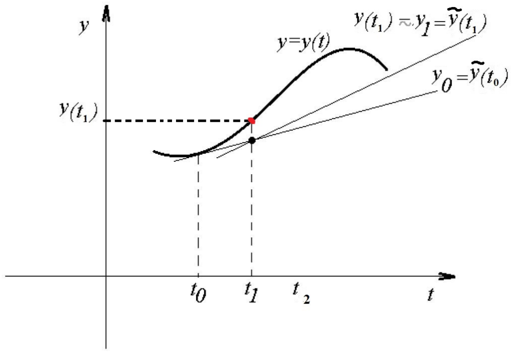


Figure 3.1: Construction of the first iterates of the Euler method.

Definition 3.3. *A numerical method that approximates $y(t_i)$ by y_i with an error $e_i = |y(t_i) - y_i|$ with*

$$e_i \leq kh^p$$

is said of order p, where k is a constant independent of i and h, and $y(t_i)$ is the exact value of the solution of the Cauchy problem at the point $t_i = t_0 + ih$.

Theorem 3.2. *Let $f(t, y)$ be a continuous function on $[a, b] \times \mathbb{R}$ and L-Lipschitz continuous with respect to the variable y, and let $y(t) \in \mathcal{C}^2[a, b]$. Then we have*

$$e_i \leq (e^{L(b-a)} - 1)\frac{M_2}{2L}h$$

where $M_2 = \max_{t\in[a,b]} |y''(t)|$, and e_i is the error made at the point (t_i, y_i), i.e., $e_i = |y(t_i) - y_i|$.

Remark 3.1. *This result can be expressed in the form $e_i \leq kh$, meaning that the Euler method is of order 1.*

Example 3.6. *Consider the following Cauchy problem be given:*

$$\begin{cases} y'(t) = ty^{1/3} \\ y(1) = 1 \end{cases}$$

Let's calculate $y(1.01), y(1.02), y(1.03)$ *using the Euler method.*
We take $y_0 = 1, t_0 = 1$ *with* $y_{i+1} = y_i + h(t_i y_i^{1/3})$ *and* $h = 0.01$*, the it results that:*

$$y(1.01) \simeq y_1 = y_0 + 001 \times t_0 \times y_0^{1/3} = 1 + 0.01 \times 1 \times 1^{1/3} = 1.01.$$
$$y(1.02) \simeq y_2 = y_1 + 0.01 \times 1.01 \times (1.01)^{1/3} = 1.0201$$
$$y(1.03) \simeq y_3 = y_2 + 0.01 \times 1.0201 \times (1.0201)^{1/3} = 1.0304.$$

Example 3.7. *Solve the following Cauchy problem using Euler method with a step size* $h = 0.25$*.*

$$\begin{cases} y'(t) = 2 - ty^2,\ t \in [0,1] \\ y(0) = 1 \end{cases}$$

The points t_i *to evaluate for* $h = 0.25$ *are* $t_0 = 0, t_1 = 0.25, t_2 = 0.5, t_3 = 0.75, t_4 = 1$*. Following the same scheme as in the previous example, we obtain:*

$$\begin{aligned} y(0.25) \simeq y_1 &= y_0 + 0.25 \times f(t_0, y_0) \\ &= 1 + 0.25(2 - 0 \times 1^2) = 1.5 \\ y(0.50) \simeq y_2 &= y_1 + 0.25 \times f(t_1, y_1) \\ &= 1.5 + 0.25(2 - 0.25 \times 1.5^2) = 1.8594 \\ y(0.75) \simeq y_3 &= y_2 + 0.25 \times f(t_2, y_2) \\ &= 1.859 + 0.25(2 - 0.5 \times 1.859^2) = 1.927 \\ y(1.00) \simeq y_4 &= y_3 + 0.25 \times f(t_3, y_3) \\ &= 1.927 + 0.25(2 - 0.75 \times 1.927^2) = 1.7308. \end{aligned}$$

Example 3.8. *Consider the following Cauchy problem:*

$$\begin{cases} y'(t) = t + y, t \in [0,1] \\ y(0) = 1. \end{cases}$$

We want to approximate the solution of this problem at $t = 1$ *using Euler method, by subdividing the interval* $[0,1]$ *into ten equal parts. Following the same procedure, we obtain the values* $\{t_i, y_i\}$ *as listed below:*

i	0	1	2	3	4	5	6	7	8	9	10
t_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y_i	1	1,1	1,22	1,362	1,5282	1,7210	1,9431	2,1974	2,4871	2,8158	3,1874

From this table, we obtain $y(1) \simeq y_{10} = 3.187$*. This approximation is quite rough because the exact solution to this problem is given by* $y(t) = 2e^t - t - 1$*, so the exact value is* $y(1) = 3.437$*.*

3.3.2 Improved Euler Method

This method is more precise than the previous one; it consists of replacing, in Euler method, the slope of the tangent at (x_n, y_n) with the corrected value at the midpoint of the interval $[x_n, x_{n+1}]$, whose algorithm is:

$$\begin{cases} y_0 = y(t_0), t_0 = a \\ y_{i+1} = y_i + hf(t_i + \frac{h}{2}, y_i + \frac{h}{2}K_1), i = 1, \ldots, n-1 \\ K_1 = f(x_i, y_i) \end{cases}$$

Example 3.9. *Consider the following Cauchy problem:*

$$\begin{cases} y'(t) = y(t) - t + 2, t \in [0,1] \\ y(0) = 2 \end{cases}$$

Using the Improved Euler method with a step size of $h = 0.1$, *we obtain*

$$\begin{cases} y_0 = y(0) = 2, h = 0.1 \\ y_1 = y(0.1) = y_0 + hf(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_1), \\ K_1 = f(t_0, y_0) = f(0.2) = 4 \\ y(0.1) \simeq y_1 = 2 + \frac{0.1}{2}f(0.05, 2.2) = 2.415. \end{cases}$$

Proceeding the same process, we obtain the results in the following table:

i	0	1	2	3	4	5	6	7	8	9	10
t_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y_i	2	2.415	2.8465	3.3111	3.8122	4.3535	4.9388	5.5727	6.2599	7.0059	7.8165

3.3.3 Second-order Runge-Kutta (Heun's) Method

Runge-Kutta methods approximate the solution with higher accuracy (they generate numerical solutions that are closer to the analytical solutions) than the Euler method. The second-order Runge-Kutta method (RK_2) is an amelioration of the Euler method. Indeed, the Euler method relies on a first-order Taylor expansion. However, it is clear that more efficient methods can be obtained by considering expansions of higher order than 1. Thus, if the function f is sufficiently differentiable, we can write:

$$y_{i+1} = y_i + h \times y'(t_i) + \frac{h^2}{2}y''(t_i)$$

with,

$$y'(t) = f(t, y) \text{ and } y''(t) = \frac{\delta f}{\delta t}(t, y) + f(t, y).\frac{\delta f}{\delta y}(t, y).$$

Hence,

$$y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2}\left(\frac{\delta f}{\delta t}(t_i, y_i) + f(t_i, y_i).\frac{\delta f}{\delta y}(t_i, y_i)\right),$$

since we have:

$$f(t_i+h, y_i+hf(t_i,y_i)) = f(t_i,y_i) + h\left(\tfrac{\delta f}{\delta t}(t_i,y_i) + f(t_i,y_i).\tfrac{\delta f}{\delta y}(t_i,y_i)\right),$$

it results that

$$y(t_{i+1}) = y(t_i) + \frac{h}{2}f(t_i,y_i) + \frac{h}{2}f(t_i+h, y_i+hf(t_i,y_i)).$$

Thus, we obtain the second-order Runge-Kutta algorithm:

$$(RK_2)\begin{cases} y_0 = y(t_0), t_0 = a \text{ and } h = \frac{b-a}{n} \\ y_{i+1} = y_i + \frac{h}{2}(K_1+K_2), i = 1,\dots,n-1 \\ K_1 = f(t_i,y_i) \\ K_2 = f(t_i+h, y_i+hK_1) \end{cases}$$

3.3.4 Fourth-order Runge–Kutta Method

This is the most accurate and widely used method in practice, with an error of order four. It calculates the value of the function at four intermediate points. Its iterative scheme is given as follows:

$$(RK_4)\begin{cases} y_0 = y(t_0), t_0 = a \text{ and } h = \frac{b-a}{n} \\ y_{i+1} = y_i + \frac{h}{6}(K_1+2K_2+2K_3+K_4), i = 1,\dots,n-1 \\ K_1 = f(t_i,y_i) \\ K_2 = f(t_i+\frac{h}{2}, y_i+\frac{h}{2}K_1) \\ K_3 = f(t_i+\frac{h}{2}, y_i+\frac{h}{2}K_2) \\ K_4 = f(t_i+h, y_i+hK_3) \end{cases}$$

Note that the number of terms retained in the Taylor series defines the order of the Runge-Kutta method. The Runge-Kutta method of order 4 truncates the Taylor series at the term $O(h^4)$.

Example 3.10. *Consider the following Cauchy problem:*

$$\begin{cases} y'(t) = y - \frac{2t}{y}, t \in [0,1] \\ y(0) = 1. \end{cases}$$

The exact solution of this problem is: $y(t) = \sqrt{2t+1}$*.*
- Compute an approximate value of $y(0.2)$ *using the* RK_2 *and* RK_4 *methods with a step size* $h = 0.2$*.*
- Evaluate the obtained results by comparing them with the exact solution.
Runge-Kutta Method of Order 2:

$$(RK_2)\begin{cases} y_0 = y(0) = 1, h = 0.2 \\ y_1 = y(0.2) = y_0 + \frac{h}{2}(K_1 + K_2), \\ K_1 = f(t_0, y_0) = f(0,1) = 1 \\ K_2 = f(t_0 + h, y_0 + hK_1) = f(0.2, 1.2) = 0.866 \\ y_1 = y(0.2) = 1 + \frac{0.2}{2}(1 + 0.866) = 1.1866. \end{cases}$$

$e_{RK_2} = |\sqrt{2 \times (0.2) + 1} - 1.1866| = 3.450709 \times 10^{-3}$.

Runge-Kutta Method of Order 4:

$$(RK_4)\begin{cases} y_0 = y(0) = 1, h = 0.2 \\ y_1 = y_0 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \\ K_1 = f(t_0, y_0) = 1 \\ K_2 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_1) = f(0.1, 1.1) = 0.918182 \\ K_3 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_2) = f(0.1, 1.091818) = 0.908637 \\ K_4 = f(t_0 + h, y_i + hK_3) = f(0.2, 1.181727) = 0.843239 \\ y_1 = 1 + \frac{0.2}{6}(K_1 + 2K_2 + 2K_3 + K_4) = 1.1832292 \end{cases}$$

$e_{RK_4} = |\sqrt{2 \times (0.2) + 1} - 1.1832292| = 1.32 \times 10^{-5}$. *Hence* $e_{RK_4} \ll e_{RK_2}$.

Example 3.11. *Give an approximate solution of the following Cauchy problem using the RK_4 method with a step size of $h = 0.25$.*

$$\begin{cases} y'(t) = 2 - ty^2, t \in [0,1] \\ y(0) = 1. \end{cases}$$

For the first step, we have

$$(RK_4)\begin{cases} y_0 = y(0) = 1, h = 0.25 \\ y_1 = y_0 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), i = 1, \dots, n-1 \\ K_1 = f(t_0, y_0) = 2 \\ K_2 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_1) = 1.8047 \\ K_3 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_2) = 1.8122 \\ K_4 = f(t_0 + h, y_i + hK_3) = 1.4722 \\ y_1 = 1 + \frac{0.25}{6}(K_1 + 2K_2 + 2K_3 + K_4) = 1.4461 \end{cases}$$

By proceeding the same process as in step 1, we obtain: $y_2 = 1.7028$, $y_3 = 1.7317$ *and* $y_4 = 1.6147$.

3.4 Matlab codes

3.4.1 Pseudo-code of Euler method

```
% Euler Method for Solving a Cauchy
  Problem
clear;clc;close all;

% Define the function f(x, y) = dy/dx
f=@(x,y)-2*x*y;  % Example: dy/dx = -2xy

% Initial conditions
x0=0;      % Initial x-value
y0=1;      % Initial y-value
```

```
h=0.1;    % Step size
x_end=2; % Final x-value

% Number of iterations
N=(x_end-x0)/h;

% Initialize arrays for x and y
x=zeros(1,N+1);
y=zeros(1,N+1);

% Set initial values
x(1)=x0;
y(1)=y0;

% Euler Method Iteration
for i=1:N
    y(i+1)=y(i)+h*f(x(i),y(i));
    x(i+1)=x(i)+h;
end

% Display results
disp('x-values:');
disp(x);
disp('y-values:');
disp(y);

% Plot the numerical solution
plot(x,y,'bo-','LineWidth',2,'MarkerSize'
   ,6);
hold on;
xlabel('x');
ylabel('y');
```

```
title('Euler Method for Solving Cauchy
   Problem');
grid on;
legend('Euler Approximation');
```

3.4.2 Pseudo-code of improved Euler method

```
% Improved Euler Method for Solving a
   Cauchy Problem
clear;clc;close all;

% Define the function f(x, y) = dy/dx
f = @(x,y)-2*x*y;  % Example: dy/dx = -2xy

% Initial conditions
x0=0;     % Initial x-value
y0=1;     % Initial y-value
h=0.1;    % Step size
x_end=2; % Final x-value

% Number of iterations
N=(x_end-x0)/h;

% Initialize arrays for x and y
x=zeros(1,N+1);
y=zeros(1, N+1);

% Set initial values
x(1)=x0;
y(1)=y0;
```

```
% Improved Euler Method Iteration (Heun's
   Method)
for i=1:N
    k1=f(x(i),y(i));
    k2=f(x(i)+h/2,y(i)+(h/2)*k1);
    y(i+1)=y(i)+h*k2;
    x(i+1)=x(i)+h;
end

% Display results
disp('x-values:');
disp(x);
disp('y-values:');
disp(y);

% Plot the numerical solution
plot(x,y,'bo-','LineWidth',2,'MarkerSize'
   ,6);
hold on;
xlabel('x');
ylabel('y');
title('Improved Euler Method for Solving
   Cauchy Problem');
grid on;
legend('Improved Euler Approximation');
```

3.4.3 Pseudo-code of Second-order Runge-Kutta (RK2) method

```
% Runge-Kutta 2nd Order Method (RK2) for
   Solving a Cauchy Problem
clear;clc;close all;
```

```
% Define the function f(x, y)=dy/dx
f = @(x, y)-2*x*y;  % Example: dy/dx=-2xy

% Initial conditions
x0=0;     % Initial x-value
y0=1;     % Initial y-value
h=0.1;    % Step size
x_end=2; % Final x-value

% Number of iterations
N=(x_end-x0)/h;

% Initialize arrays for x and y
x=zeros(1,N+1);
y=zeros(1,N+1);

% Set initial values
x(1)=x0;
y(1)=y0;

% RK2 Method Iteration
for i=1:N
    k1=f(x(i),y(i));
    y_predictor=y(i)+h*k1;
    k2=f(x(i)+h,y_predictor);
    y(i+1)=y(i)+(h/2)*(k1+k2);
    x(i+1)=x(i)+h;
end

% Display results
disp('x-values:');
disp(x);
disp('y-values:');
disp(y);

% Plot the numerical solution
plot(x,y,'bo-','LineWidth',2,'MarkerSize'
   ,6);
hold on;
xlabel('x');
ylabel('y');
title('Runge-Kutta 2nd Order Method (RK2)
   for Solving Cauchy Problem');
grid on;
legend('RK2 Approximation');
```

3.4.4 Pseudo-code of Fourth-order Runge-Kutta (RK4) method

```
% Fourth-Order Runge-Kutta (RK4) Method
   for Solving a Cauchy Problem
clear;clc;close all;

% Define the function f(x,y)=dy/dx
f = @(x,y)-2*x*y;  % Example: dy/dx=-2xy

% Initial conditions
x0=0;     % Initial x-value
y0=1;     % Initial y-value
h=0.1;    % Step size
x_end=2;  % Final x-value

% Number of iterations
```

```
N=(x_end-x0)/h;

% Initialize arrays for x and y
x=zeros(1,N+1);
y=zeros(1,N+1);

% Set initial values
x(1)=x0;
y(1)=y0;

% RK4 Method Iteration
for i=1:N
    k1=f(x(i), y(i));
    k2=f(x(i)+h/2,y(i)+(h/2)*k1);
    k3=f(x(i)+h/2,y(i)+(h/2)*k2);
    k4=f(x(i)+h,y(i)+h*k3);

    y(i+1)=y(i)+(h/6)*(k1+2*k2+2*k3+k4);
    x(i+1)=x(i)+h;
end

% Display results
disp('x-values:');
disp(x);
disp('y-values:');
disp(y);

% Plot the numerical solution
plot(x,y,'bo-','LineWidth',2,'MarkerSize'
   ,6);
hold on;
xlabel('x');
ylabel('y');
```

```
title('Fourth-Order Runge-Kutta (RK4)
   Method for Solving Cauchy Problem');
grid on;
legend('RK4 Approximation');
```

3.5 Solved Exercises

Exercise 3.1. *Consider the following Cauchy problem:*

$$\begin{cases} y'(t) & = 2t - y(t) \Big| t \in [0,1] \\ y(0) & = 1. \end{cases} \qquad (P)$$

a- Prove that the problem (P) admits a unique solution.

b- Verify that the problem (P) admits the equation (3.5) as a particular solution.

$$y(t) = 2t - 2 + 3e^{-t}. \qquad (3.5)$$

c- Provide the iterative scheme of the fourth-order Runge–Kutta algorithm to solve the problem (P).

d- Apply the fourth-order Runge–Kutta algorithm (RK4) for this problem with $h = 0.1$ to evaluate the solution at $t = 0.3$. Compare the obtained solution with the exact solution.

Solution

a- We have $\frac{\delta f}{\delta y} = 1$, which is a continuous and bounded function, so this problem admits a unique solution.

b- We have, according to (3.5),

$$\begin{aligned} y'(t) =& 2 - 3e^{-t} \\ =& 2 - 3e^{-t} - 2t + 2t \\ =& - y(t) + 2t. \end{aligned}$$

On the other hand, we have $y(0) = -2+3 = 1$, from which we deduce that the equation (3.5) is a particular solution.

c-

$$(RK_4)\begin{cases} y_0 = y(t_0), t_0 = a \text{ et } h = \frac{b-a}{n} \\ y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), i = 1, \dots, n-1 \\ K_1 = f(t_i, y_i) \\ K_2 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}K_1) \\ K_3 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}K_2) \\ K_4 = f(t_i + h, y_i + hK_3) \end{cases}$$

d- Apply the fourth-order Runge-Kutta method algorithm RK_4 with $h = 0.1$:

$$(RK_4)\begin{cases} y_0 = y(0) = 1, h = 0.1 \\ y_1 = y_0 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), i = 1, \dots, n-1 \\ K_1 = f(t_0, y_0) = f(0, 1) - 1 \\ K_2 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_1) = f(0.05, 1.05) = -0.95 \\ K_3 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_2) = f(0.05, 0.955) = -0.852 \\ K_4 = f(t_0 + h, y_0 + hK_3) = f(0.05, 0.914) = -0.814 \\ y_1 = 1 + \frac{0.1}{6}(K_1 + 2K_2 + 2K_3 + K_4) = 0.943 \end{cases}$$

hence $y(0.1) \simeq y_1 = 0.943$.

$$(RK_4)\begin{cases} y_2 = y_1 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), i = 1, \dots, n-1 \\ K_1 = f(t_1, y_1) = f(0.1, 0.9430) = -0.743 \\ K_2 = f(t_1 + \frac{h}{2}, y_1 + \frac{h}{2}K_1) = f(0.15, 0.905) = -0.605 \\ K_3 = f(t_1 + \frac{h}{2}, y_1 + \frac{h}{2}K_2) = f(0.15, 0.9127) = -0.6127 \\ K_4 = f(t_1 + h, y_1 + hK_3) = f(0.2, 0.8818) = -0.4818 \\ y_1 = 0.943 + \frac{0.1}{6}(K_1 + 2K_2 + 2K_3 + K_4) = 0.882 \end{cases}$$

then $y(0.2) \simeq y_2 = 0.882$.

By repeating the same process, we obtain: $y(0.3) \simeq y_3 = 0.8436$

Comparison: The exact value at $t = 0.3$ is $y(0.3) = 0.8225$, then

$$err = |0.8225 - 0.8436| = 0.0216$$

Exercise 3.2.

a- Provide the iterative scheme of the Euler algorithm to solve the problem (P) *of Exercise 1.*

b- Apply the Euler algorithm for this problem with $h = 0.1$ *to evaluate the solution at* $t = 0.3$*. Compare the obtained solution with the exact solution.*

Solution

a-

$$\begin{cases} y_0 = y(t_0), t_0 = a \\ y_{i+1} = y_i + hf(t_i, y_i), i = 1, \dots, n-1 \end{cases}$$

with $h = \frac{b-a}{n}$, et $t_{i+1} = t_i + h$.

b-

$$\begin{aligned} y(0.1) &\simeq y_1 = y_0 + 0.1(2t_0 - y(t_0)) = 0.92. \\ y(0.2) &\simeq y_2 = y_1 + 0.1(2t_1 - y(t_1)) = 0.868 \\ y(0.3) &\simeq y_3 = y_2 + 0.1(2t_2 - y(t_2)) = 0.8412 \end{aligned}$$

Then $y(0.3) \simeq y_3 = 0.8412$.

Comparison: The exact value at $t = 0.3$ is $y(0.3) = 0.8225$, so the error made when applying the Euler algorithm is

$$err = |0.8225 - 0.8412| = 0.019.$$

The theoretical error is given by

$$e_t \leq (e^{L(b-a)} - 1)\frac{M_2}{2L}h,$$

where $M_2 = \max_{t\in[0,1]} |y''(t)|$ and L is the Lipschitz constant of f with respect to y, which is equal to 1.
In addition, we have

$$y''(t) = 3e^{-t}$$

Then $M_2 = \max\limits_{t\in[0,1]} |3e^{-t}| = 3$. Hence,

$$\begin{aligned} e_t &\leq (e^{L(b-a)} - 1)\frac{M_2}{2L}h \\ &\leq (e^{1(0.3-0)} - 1)\frac{3 \times 0.1}{2 \times 1} \\ &\leq 0.05247 \end{aligned}$$

It is clear that $err \leq e_t$, so the Euler method provides a good approximation for this Cauchy problem at $t = 1$.

Exercise 3.3. *Consider the following differential equation:*

$$\begin{cases} y'(t) &= y(t) + t, t \in [0,1] \\ y(0) &= 1. \end{cases}$$

The exact solution of this equation is $y(t) = -1 - t + 2e^t$.
- Approximate the solution of this equation at $t = 1$ using Euler method by subdividing the interval into 10 equal parts.
- Compare the obtained solution with the exact solution.

Solution
Let $f(t,y) = y(t) + t$, the points t_i to evaluate for $h = 0.1$ are $t_1 = 0.1, t_2 = 0.2, t_3 = 0.3, \ldots, t_{10} = 1$. By following the same procedure

as in the previous examples, we obtain:

$$\begin{aligned} y(0.1) &\simeq y_1 = y_0 + 0.1 \times f(t_0, y_0) = 1.1 \\ y(0.2) &\simeq y_2 = y_1 + 0.1 \times f(t_1, y_1) = 1.22 \\ y(0.3) &\simeq y_3 = y_2 + 0.1 \times f(t_2, y_2) = 1.362 \\ &\vdots \\ y(1) &\simeq y_{10} = y_9 + 0.1 \times f(t_9, y_9) = 3.1874\,. \end{aligned}$$

That is, the approximation at $t = 1$ of $y(t)$ is $y_{10} = 3.1874$.

- **Comparison of results:**
The exact value at $t = 1$ is $y(1) = -1 - 1 + 2e^1 = 3.4366$. Thus, the error made when applying Euler method is

$$err = |3.4366 - 3.1874| = 0.25.$$

Now, let's find the theoretical error, which is given by

$$e_t \leq (e^{L(b-a)} - 1)\frac{M_2}{2L}h$$

where $M_2 = \max\limits_{t\in[0,1]} |y''(t)|$ and L is the Lipschitz constant of f with respect to y.
We have,

$$|f(t, y_1) - f(t, y_2)| = |y_1 - y_2| \Rightarrow L = 1.$$

Furthermore, we have,

$$\begin{aligned} y''(t) &= y'(y) + 1 = y(t) + t + 1 \\ &= 1 + t + (-1 - t + 2e^t) \\ &= 2e^t \end{aligned}$$

$M_2 = \max\limits_{t\in[0,1]} |2e^t| = 2e$. Hence

$$\begin{aligned} e_t &\leq (e^{L(b-a)} - 1)\frac{M_2}{2L}h \\ &\leq (e^{1(1-0)} - 1)\frac{2e \times 0.1}{2 \times 1} \\ &\leq 0.4673\,. \end{aligned}$$

It is clear that $err \leq e_t$, so the Euler method provides a good approximation for this problem at $t = 1$.

Exercise 3.4. *Determine an approximate solution of the following Cauchy problem using the fourth-order Runge-Kutta method with a step size of* $h = 0.1$.

$$\begin{cases} y'(t) = y(t) - t + 2, t \in [0,1] \\ y(0) = 2. \end{cases}$$

Solution

We have

$$(RK_4)\begin{cases} y_0 = y(0) = 2, h = 0.1 \\ y_1 = y_0 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), i = 1, \ldots, n-1 \\ K_1 = f(t_0, y_0) = 0.4 \\ K_2 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_1) = 0.4150 \\ K_3 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_2) = 0.4157 \\ K_4 = f(t_0 + h, y_i + hK_3) = 0.4365 \\ y_1 = 1 + \frac{0.25}{6}(K_1 + 2K_2 + 2K_3 + K_4) = 2.4163 \end{cases}$$

By repeating the same process for the other iterations, we obtain the results listed in the following table:

i	0	1	2	3	4	5	6	7	8	9	10
t_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y_i	2	2.4163	2.8659	3.5323	3.8793	4.4513	5.0728	5.7492	6.4863	7.2903	8.1684

3.5.1 Supplementary Exercises

Exercise 3.5. *Consider the following ordinary differential equation:*

$$\begin{cases} y'(t) &= t - \ln y \\ y(2) &= 3.4 \end{cases}$$

- Calculate $y(2.8)$ using the fourth-order Runge-Kutta method with $h = 0.8$ and then with $h = 0.4$.

Result

- $y(2.8)$ with $h = 0.8$ is $y(2.8) \simeq y_1 = 4.255952$.
- $y(2.8)$ with $h = 0.4$ is $y(2.8) \simeq y_2 = 4.255888$.

Exercise 3.6. *Consider the following ordinary differential equation:*

$$y'(t) = \frac{y^2}{t}$$
$$y(1) = 1$$

- Calculate $y(1.5)$ using the fourth-order Runge-Kutta method with a step size of $h = 0.5$.
- Recalculate $y(1.5)$ with $h = 0.25$.

Result

- $y(1.5)$ with $h = 0.5$ is $y(1.5) \simeq y_1 = 1.67985$
- $y(1.5)$ with $h = 0.25$ is $y(1.5) \simeq y_2 = 1.68178$

Exercise 3.7. *Consider the following ordinary differential equation:*

$$y'(t) = -y + t + 1 \Big| t \in [0, 1]$$
$$y(0) = 1.$$

a- *Calculate an approximation of $y(0.2)$ using the Euler method, with a step size of $h = 0.1$.*

b- Calculate an approximation of $y(0.2)$ using the improved Euler method, with a step size of $h = 0.1$.

c- For each method, calculate the error made by comparing the result obtained with the exact solution $y^*(0.2) = 1.018731$.

Result

a- $y(0.2) \simeq y_2 = 1.01$ and $|y_2 - y(0.2)| = 0,008731$.
b- $y(0.2) \simeq y_2 = 1.019025$ and $|y_2 - y(0.2)| = 0,000294$.

Exercise 3.8. *Consider the following ordinary differential equation:*

$$\frac{dy}{5y-3} = dt$$

1- Determine the expression of the general solution of the given ODE.

2- Given that $y(0) = 1$, give an approximation of $y(0.5)$ using RK4 with a steplength $h = 0.25$. Compute the error between $y(0.5)$ and its approximated value.

Exercise 3.9. *Consider the following Cauchy problem:*

$$(P)\begin{cases} (3t+1)dy = y\,dt, \quad t \in [0,1] \\ y(0) = 1 \end{cases}$$

Let $h = 0.1$ be the step size for subdividing the interval $[0,1]$.

1- *Give the expression of the function f.*

2- *Determine an approximate value of $y(0.2)$ using Euler's method and then the Second-order Runge-Kutta Method (RK2).*

3- *Verify that $y(t) = (3t+1)^{1/3}$ is the exact solution of Problem (P).*

4- *Evaluate the results obtained in Question 2).*

4 Numerical Solution of Systems of Linear Equations

In practice, engineers often encounter problems whose solution requires solving a system of linear equations that models the problem under consideration. For example, determining currents and voltages in electrical networks requires solving a system of linear equations. That is, we seek the vector $X \in \mathbb{R}^n$, where $X = (x_1, x_2, \ldots, x_n)$, which is the solution of the following linear system:

$$AX = b \Longleftrightarrow \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{4.1}$$

This system admits a unique solution when the determinant of A is nonzero, which we will assume below. Solving this system using direct methods becomes impractical when n is relatively height. Therefore, it is preferable to use numerical methods that construct a sequence converging to the solution of the system.

In this chapter, we present two numerical methods that provide approximate solutions to systems of linear equations using a linear function f such that $X^{k+1} = f(X^k)$, $k \in \mathbb{N}$. These methods are easy to implement, require minimal memory, and produce results with the desired accuracy.

Given an arbitrary initial vector X^0, we construct a sequence of vectors

$$X^0, X^1, \ldots, X^k, \ldots$$

which converges to the solution X^* of the linear system $AX = b$.

We consider the linear system (4.1), where A is an $n \times n$ invertible matrix and b is a vector in $\mathbb{R}^n$. For any invertible square matrix M of order n, the system (4.1) is equivalent to

$$MX - (M - A)X = b.$$

By setting $N = M - A$, $B = M^{-1}N$, and $c = M^{-1}b$, we obtain

$$X = BX + c\,.$$

This allows us to define the following iterative formula:

$$\begin{cases} X^0 \in \mathbb{R}^n \text{initial vector} \\ X^{k+1} = BX^k + c. \end{cases} \tag{4.2}$$

Let X^* be the exact solution of (4.1). If we denote $e^k = \|X^k - X^*\|$ as the k-th error vector, we obtain

$$\begin{aligned} e_k &= \|X^k - X^*\| = \|(BX^{k-1} + c) - (BX^* + c)\| = B\|X^{k-1} - X^*\| \\ &= Be_{k-1} = B^k e_0. \end{aligned}$$

Remark 4.1. *In practice, if we impose a precision ε, we can estimate the error by:*

$$\|X^k - X^{k-1}\| \leq \varepsilon$$

This means that, for all $i \in \{1, \ldots, n\}$*, we have:*

$$|x_i^k - x_i^{k-1}| \leq \varepsilon.$$

Theorem 4.1. *The iterative method* (4.2) *converges if the sequence of vectors* $\{e^k\}_{k\in\mathbb{N}}$ *converges to zero independently of the initial vector* X^0*, if one of the three norms is less than 1:*

- $\|B\|_1 = \max_j(\sum_{i=1}^{n} |B_{ij}|)$

- $\|B\|_\infty = \max_i(\sum_{j=1}^{n} |B_{ij}|)$

- $\|B\|_2 = \sqrt{\rho(BB^t)}.$

Depending on the choices of the matrices M and N, we obtain different iterative methods. Let D be the matrix formed by the diagonal elements of A, E be the matrix formed by the $-a_{ij}$ when $i > j$, and F be the matrix formed by the $-a_{ij}$ when $i < j$, so that $A = D - (E + F)$.

- The matrix D is a diagonal matrix of A, given by:

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

- The matrix E is a lower triangular matrix of A with a zero diagonal.

$$E = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ -a_{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{pmatrix}$$

- The matrix F is an upper triangular matrix of A with a zero diagonal.

$$F = \begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

4.1 Jacobi Method

In the Jacobi iterative method, the matrix A of the system $AX = b$ is decomposed as $A = M - N$. The matrix M corresponds to the diagonal of A (with zeros outside the diagonal), so $M = D$, and the matrix N is the matrix A in which the diagonal elements are replaced by zeros, i.e., $N = E + F$. The matrix $J = M^{-1}N = D^{-1}(E + F) = I - D^{-1}A$ is called the Jacobi matrix. Starting from an initial vector $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)^t$, at each step, we compute X^k using the following formula:

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^k \right), i = 1, 2 \ldots, n\,. \qquad (4.3)$$

Remark 4.2. *The Jacobi method does not always converge. If A is a positive definite matrix, the Jacobi method converges. Similarly, if A is a strictly diagonally dominant matrix, i.e., $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$, then the Jacobi method is convergent.*

Example 4.1. *Consider the following system*

$$\begin{cases} 4x_1 + 2x_2 + x_3 & = 4 \\ -x_1 + 2x_2 & = 2 \\ 2x_1 + x_2 + 4x_3 & = 9\,. \end{cases}$$

Let $X^0 = (0,0,0)^t$ *be the initial vector. By calculating the first five iterations, we obtain:*

$$X^1 = \begin{pmatrix} 1 \\ 1 \\ 9/4 \end{pmatrix}, X^2 = \begin{pmatrix} -1/16 \\ 3/2 \\ 3/2 \end{pmatrix}, X^3 = \begin{pmatrix} -1/8 \\ -1/32 \\ 61/32 \end{pmatrix}, X^4 = \begin{pmatrix} 5/128 \\ 15/16 \\ 265/128 \end{pmatrix} \text{ and } X^5 = \begin{pmatrix} 7/512 \\ 261/256 \\ 511/256 \end{pmatrix}$$

Example 4.2. *Let us solve the following system using the Jacobi method:*

$$\begin{cases} 3x_1 + x_2 - x_3 = 2 \\ x_1 + 5x_2 + 2x_3 = 17 \\ 2x_1 - x_2 - 6x_3 = -18 \end{cases}$$

The iterative scheme is

$$\begin{cases} i = 1, & x_1^{k+1} & = \frac{1}{3}\left(2 - x_2^k + x_3^k\right) \\ i = 2, & x_2^{k+1} & = \frac{1}{5}\left(17 - x_1^k - 2x_3^k\right) \\ i = 3, & x_3^{k+1} & = \frac{-1}{6}\left(-18 - 2x_1^k + x_2^k\right) \end{cases}$$

Let $X^0 = (0,0,0)^t$ *be the initial vector, we obtain:* $X_1 = (2/3, 17/5, 3)^t$, $X_2 = (8/15, 31/15, 2.6555)^t$.
After 10 *iterations, we obtain the following table of results:*

k	x_1^k	x_2^k	x_3^k
0	0	0	0
1	0,666666	3,4	3
2	0,533333	2,066667	2,655556
3	0,862963	2,231111	2,833333
4	0,867407	2,094074	2,915802
5	0,940576	2,0601198	2,970123
6	0,959975	2,035835	2,970159
7	0,978108	2,019941	2,980686
8	0,986915	2,012104	2,989379
9	0,992425	2,006865	2,993621
10	0,995585	2,004067	2,996331

From this table, we notice that the sequence of points converge towards the solution $X = (1, 2, 3)^t$.

4.2 Gauss-Seidel Method

The Gauss-Seidel method is an amelioration of the Jacobi method because, it makes the iterative process faster. In the Jacobi method, the generated vectors $X^1, X^2, \dots, X^k, \dots$ convergences to the solution X^*, which means that each new vector is better than the previous one. However, to calculate the component x_2^2 of the vector X^2, in the Jacobi, we use those of X^1 even though x_1^2 is already calculated and it is better than x_1^1. To overcome this drawback, the principle of the Gauss-Seidel method, is to use each component as soon as it is calculated. Thus, to compute the component x_i^{k+1}, we use all the components from x_1^{k+1} to x_{i-1}^{k+1} already determined at iteration $(k+1)$ as well as the components x_{i+1}^k to x_n^k that are still at iteration k.

The matrix A is decomposed as $A = M - N$. We take:

$$M = D - E, \quad N = F.$$

This modifies relation (4.3) as follows: for $k \geq 0$ (assuming again that $a_{ii} \neq 0$ for $i = 1, \dots, n$).

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right), i = 1, 2 \dots, n\,. \tag{4.4}$$

Remark 4.3. *The Gauss-Seidel method does not always converge. If A is a positive definite matrix, the iterative method converges. Similarly, if A is a diagonally dominant matrix, i.e., if*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|,$$

then the Gauss-Seidel method converges.

Example 4.3. *Solve the following system using the Gauss-Seidel method using 3 iterations and* $X^0 = (0,0,0)^t$ *as a starting point.*

$$\begin{cases} -x_1 + x_2 + 3x_3 & = -1 \\ x_1 + 2x_2 & = 2 \\ 3x_1 + x_2 - x_3 & = 1 \end{cases}$$

This system can be written in following form:

$$\begin{cases} i=1, & x^1_{k+1} & = 1 + x^2_k + 3x^3_k \\ i=2, & x^2_{k+1} & = 1 - \frac{1}{2}x^1_{k+1} \\ i=3, & x^3_{k+1} & = -1 + 3x^1_{k+1} - x^2_{k+1} \end{cases}$$

- First iteration, we obtain $X^1 = (1, 0.5, 1.5)^t$*,*
- Second iteration, we obtain $X^2 = (6, -2, 19)^t$ *,*
- Third iteration, we obtain $X^3 = (56, -27, 194)^t$ *.*
Hence after three iterations, $X^* \simeq (56, -27, 194)^t$

Example 4.4. *Let us solve the linear system of Example 2 using the Gauss-Seidel method. The iterative scheme, in this case, is written as follows:*

$$\begin{cases} i=1, & x_1^{k+1} & = \frac{1}{3}\left(2 - x_2^k + x_3^k\right) \\ i=2, & x_2^{k+1} & = \frac{1}{5}\left(17 - x_1^{k+1} - 2x_3^k\right) \\ i=3, & x_3^{k+1} & = \frac{-1}{6}\left(-18 - 2x_1^{k+1} + x_2^{k+1}\right) \end{cases}$$

Starting from $X^0 = (0,0,0)^t$*, we find* $X^1 = \left(\frac{2}{3}, \frac{49}{15}, \frac{241}{90}\right)^t$*. After 10 iterations, we obtain the following table of results:*

k	x_k^1	x_k^2	x_k^3
0	0	0	0
1	0.6666667	3.266667	2.677778
2	0.4703704	2,234815	2.784321
3	0.8498354	2.116305	2.930561
4	0.9380855	2.040158	2.972669
5	0.9775034	2.015432	2.989929
6	0.9914991	2.005729	2.996212
7	0.9968271	2.002150	2.998584
8	0.9988115	2.000804	2.999470
9	0.9995553	2.000301	2.999802
10	0.9998335	2.000113	2.999926

It can be observed that for the same number of iterations, the approximate solution obtained by the Gauss-Seidel method is more accurate. The Gauss-Seidel method generally converges more quickly than the Jacobi method, but not always.

4.3 Matlab codes

4.3.1 Pseudo-code of Jacobi method

```
clc; clear all;close all
% Jacobi Method to solve Ax = b
% Input matrix A and vector b

A = [5 -2 3;
    -3 9 1;
     2 -1 -7];
b = [-1; 2; 3];
x0 = [0; 0; 0]; % Initial guess
```

```
max_iter = 100; % Maximum number of
   iterations
tol = 1e-4; % Tolerance for convergence
n = length(b);% Size of the system

x = x0;% Initialize variables
x_new = x0;
fprintf('Iter\t\tx1\t\t\tx2\t\t\tx3\n');
% Jacobi iteration
for k = 1:max_iter
    for i = 1:n
        sigma = 0;
        for j = 1:n
            if j ~= i
                sigma = sigma + A(i,j) * x
                   (j);
            end
        end
        x_new(i) = (b(i) - sigma) / A(i,i)
           ;
    end

    % Print current iterate
    fprintf('%d\t\t%.6f\t%.6f\t%.6f\n', k,
        x_new(1), x_new(2), x_new(3));

    % Check for convergence
    if norm(x_new - x, inf) < tol
        fprintf('Jacobi method converged
           in %d iterations.\n', k);
        break;
    end
    x = x_new;
end
% Output result
disp('Solution x:');
disp(x_new);
```

4.3.2 Pseudo-code of Gauss-Seidel method

```
% Gauss-Seidel Method for solving AX = b
% Inputs:
clc;clear all;close all
A = [5 -2 3;
    -3 9 1;
    2 -1 -7];
b = [-1; 2; 3];
X = [0; 0; 0];% Initial guess
tol = 1e-4;% Tolerance
max_iter = 100;%maximum number of
   iterations
n = length(b);% Size of the system
iter = 0;
while iter < max_iter
    X_old = X;
    for i = 1:n
        sum1 = A(i, 1:i-1) * X(1:i-1);
        sum2 = A(i, i+1:n) * X_old(i+1:n);
        X(i) = (b(i) - sum1 - sum2) / A(i,
            i);
    end
    fprintf('%d\t\t%.6f\t%.6f\t%.6f\n',
       iter, X(1), X(2), X(3));
    % Check for convergence
    if norm(X - X_old, inf) < tol
        fprintf('The Gauss-Seidel method
           converged after %d iterations.\
           n', iter);
        break;
    end
    iter = iter + 1;
end
% Final result
disp('Solution x =');
disp(X);
```

4.4 Solved Exercises

Exercise 4.1. *Solve the following system using the Jacobi method and determine the number of iterations required to obtain an error* $\varepsilon = \|x^k - x^{k-1}\| \leq 10^{-4}$*, taking the initial vector* $X^0 = (0,0,0)^t$.

$$\begin{cases} 4x_1 + 1x_2 + x_3 &= 4 \\ -x_1 + 2x_2 &= 2 \\ 2x_1 + x_2 + 4x_3 &= 9 \end{cases}$$

Solution

For each iteration k, the iterative scheme of the Jacobi method is written in this case as follows:

$$\begin{cases} i=1, & x_1^{k+1} &= \frac{1}{4}\left(4 - 4x_2^k - x_3^k\right) \\ i=2, & x_2^{k+1} &= \frac{1}{2}\left(2 + x_1^k\right) \\ i=3, & x_3^{k+1} &= \frac{1}{4}\left(9 - 2x_1^k - x_2^k\right) \end{cases}$$

Starting from $X^0 = (0,0,0)^t$, to achieve the prescribed accuracy, we perform 12 iterations, the results of which are presented in the following table.

k	x_k^1	x_k^2	x_k^3
0	0	0	0
1	1	1	2.25
2	-0.0625	1.5	1.5
3	-0.125	0.9688	1.9063
4	0.0391	0.9375	2.0703
5	0.0137	1.0195	1.9961
6	-0.0088	1.0068	1.9883
7	-0.0005	0.9956	2.0027
8	0.0015	0.9998	2.0013
9	-0.0002	1.0008	1.9993
10	-0.0002	0.9999	1.9999
11	0.0001	0.9999	2.0001
12	0	1	2

Exercise 4.2. *Consider the following system*

$$\begin{cases} 2x_1 - x_2 + x_3 & = 3 \\ x_1 + 7x_2 - 3x_3 & = 6 \\ -x_1 + 3x_2 + 4x_3 & = 17 \end{cases}$$

a- Starting from $X^0 = (0,0,0)^t$, determine the first six iterations of the Jacobi and Gauss-Seidel methods.

b- Given that the exact solution is $X = (1,2,3)^t$, what can we conclude?

Solution

a- For each iteration k, the iterative scheme of the Jordan method is written in this case as follows:

$$\begin{cases} i=1, & x_1^{k+1} = \frac{1}{3}\left(2 - x_2^k - x_3^k\right) \\ i=2, & x_2^{k+1} = \frac{1}{5}\left(17 - x_1^k - 2x_3^k\right) \\ i=3, & x_3^{k+1} = -\frac{1}{6}\left(-18 - 2x_1^k + x_2^k\right) \end{cases}$$

Starting from $X^0 = (0,0,0)^t$, we obtain

$$\begin{aligned} X_1 &= (1.5000, 0.8571, 4.2500)^t \\ X_2 &= (-0.1964, 2.4643, 3.9821)^t \\ X_3 &= (0.7411, 2.5918, 2.3527)^t \\ X_4 &= (1.6196, 1.7596, 2.4914)^t \\ X_5 &= (1.1341, 1.6935, 3.3352)^t \\ X_6 &= (0.6791, 2.1245, 3.2634)^t \end{aligned}$$

with $\varepsilon = \|X_6 - X^*\| = 0.4334$.

- For each iteration k, the iterative scheme of the Gauss-Seidel method is written in this case as follows:

$$\begin{cases} i=1, & x_1^{k+1} = \frac{1}{3}\left(2 - x_2^k - x_3^k\right) \\ i=2, & x_2^{k+1} = \frac{1}{5}\left(17 - x_1^{k+1} - 2x_3^k\right) \\ i=3, & x_3^{k+1} = -\frac{1}{6}\left(-18 - 2x_1^{k+1} + x_2^{k+1}\right) \end{cases}$$

Starting from $X^0 = (0,0,0)^t$, we obtain

$$\begin{aligned} X_1 &= (1.5, 0.6429, 4.1429)^t \\ X_2 &= (-0.25, 2.6684, 2.1862)^t \\ X_3 &= (1.7411, 1.5454, 3.5262)^t \\ X_4 &= (0.5096, 2.2956, 2.6557)^t \\ X_5 &= (1.3199, 1.8067, 3.2249)^t \\ X_6 &= (0.7909, 2.1263, 2.8530)^t \end{aligned}$$

avec $\varepsilon = \|X_6 - x^*\| = 0.2851.$
b- We note that, for the same number of iterations, the approximate solution obtained by the Gauss-Seidel method is more precise.

Exercise 4.3. *Using the Gauss-Seidel method, approximate the solution of the following system of linear equation within a precision of* 10^{-3}

$$\begin{cases} 8x_1 + x_2 + x_3 & = 26 \\ x_1 + 5x_2 - x_3 & = 7 \\ x_1 - x_2 + 5x_3 & = 7 \end{cases}$$

Solution
For each iteration k, the Gauss-Seidel method is written in this case as follows:

$$\begin{cases} i = 1, & x_1^{k+1} & = \frac{1}{8}\left(26 - x_2^k - x_3^k\right) \\ i = 2, & x_2^{k+1} & = \frac{1}{5}\left(7 - x_1^{k+1} + x_3^k\right) \\ i = 3, & x_3^{k+1} & = \frac{1}{5}\left(7 - x_1^{k+1} + x_2^{k+1}\right) \end{cases}$$

Starting from $X^0 = (0, 0, 0)^t$, it results that

$$\begin{aligned} X_1 &= (3.25, 0.75, 0.9000)^t \\ X_2 &= (3.0438, 0.9712, 0.9855)^t \\ X_3 &= (3.0054, 0.996, 0.9981)^t \\ X_4 &= (3.0007, 0.9995, 0.9997)^t \\ X_5 &= (3.0001, 0.9999, 1)^t\,. \end{aligned}$$

The generated vectors of this system converge to $X^* = (3, 1, 1)^t$.

5
Polynomial Interpolation

In practice, we often encounter problems where the function f is not known explicitly but is only known at certain points $x_0, x_1, \ldots, x_n$, or can be evaluated only by calling expensive code. However, in many cases, we need to perform operations (differentiation, integration, minimization, etc.) on the function f. We therefore seek to reconstruct f using an approximating function f_r that is both simple to represent and efficient to evaluate, based solely on the discrete data of f. We require that f_r remains a faithful approximation of f at all points of interest.

In this course, we focus on reconstructing f using polynomials. More precisely, given $n+1$ points with distinct abscissae $m_i(x_i, f(x_i)), i = 0, 1, \ldots, n$ in the plane, the polynomial interpolation problem consists of finding a polynomial $P(x)$ whose graph passes through all $n+1$ points m_i, that is:

$$P(x_i) = f(x_i), \quad \forall i = 0, 1, \ldots, n \tag{5.1}$$

In this chapter, we present numerical method for approximating f by a polynomial form, that is:

$$P(x) = a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0$$

where a_i $(i = 0, 1, 2, \ldots, n)$ are coefficients to be determined. The polynomials we will study differ only in how the coefficients a_i $(i = 0, 1, \ldots, n)$ are determined, since for a given set of data points, the interpolation polynomial is unique.

5.1 Lagrange Interpolation

Let $(n+1)$ distinct points $x_0, x_1, x_2, \ldots, x_n$ be given, and let f be a function whose values at these points are $f(x_0), f(x_1), \ldots, f(x_n)$. Then, there exists a unique polynomial of degree less than or equal to n that coincides with the interpolation points, i.e.:

$$f(x_i) = P(x_i), \quad i = 0, 1, \ldots, n.$$

This polynomial is given by:

$$P(x) = \textstyle\sum_{i=0}^{n} f(x_i)L_i(x) = f(x_0)L_0(x) + f(x_1)L_1(x) + \cdots + f(x_n)L_n(x),$$

where

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}, \quad i = 0, \ldots, n.$$

The polynomial $P(x)$ is called the **Lagrange interpolation polynomial** of the function f at the points $x_0, x_1, \ldots, x_n$, and the polynomials $L_i(x)$ are called the **Lagrange basis polynomials** associated with these points.

Theorem 5.1 (Uniqueness and Error Bound). *Let $f \in C^{n+1}[a, b]$, and let $P(x)$ be the interpolation polynomial of f at the points $m_i(x_i, f(x_i))$, for $i = 1, \ldots, n$. Then for all $x \in [a, b]$, there exists $\xi_x \in$*

$]\min\limits_{1,\dots,n}\{x_i\}, \max\limits_{1,\dots,n}\{x_i\}[$ *such that the error* $f(x) - P(x)$ *is given by*

$$E(x) = \frac{\gamma_{n+1}(x)}{(n+1)!} f^{(n+1)}(\xi_x),$$

where $\gamma_{n+1}(x) = \prod\limits_{i=0}^{n}(x - x_i)$. *If we set* $M_{n+1} = \max\limits_{a \le x \le b} |f^{(n+1)}(x)|$, *then we have*

$$E(x) \le \frac{|\gamma_{n+1}(x)|}{(n+1)!} M_{n+1}.$$

Example 5.1. *Let's determine the Lagrange polynomial which interpolates the function defined by values of the following table*

x_i	0	2	3	5
$y_i = f(x_i)$	-1	2	9	87

We have

$$P(x) = \sum_{i=0}^{3} f(x_i)L_i(x) = f(x_0)L_0(x) + f(x_1)L_1(x) + f(x_2)L_2(x) + f(x_3)L_3(x).$$

with

$$L_i(x) = \prod_{j=0, j\neq i}^{3} \frac{x - x_j}{x_i - x_j}, i = 0, 1, 2, 3$$

Then

$$\begin{aligned}
P(x) &= -\frac{(x-2)(x-3)(x-5)}{(0-2)(0-3)(0-5)} + 2\frac{(x-0)(x-3)(x-5)}{(2-0)(2-3)(2-5)} \\
&+ 9\frac{(x-0)(x-2)(x-5)}{(3-0)(3-2)(3-5)} + 87\frac{(x-0)(x-2)(x-3)}{(5-0)(5-2)(5-3)} \\
&= \frac{53}{30}x^3 - 7x^2 + \frac{253}{30}x - 1.
\end{aligned}$$

Example 5.2. *Using Lagrange method, construct the interpolation polynomial* $P(x)$ *of degree four that interpolates the points* $(x_0, f(x_0)) =$

$(0,0), (x_1, f(x_1)) = (1,5), (x_2, f(x_2)) = (2,15), (x_3, f(x_3)) = (3,0)$ *and* $(x_4, f(x_4)) = (4,3)$.
We have,

$$P(x) = \sum_{i=0}^{4} f(x_i)L_i(x) = f(x_0)L_0(x) + f(x_1)L_1(x) + f(x_2)L_2(x) + f(x_3)L_3(x) + f(x_4)L_4(x).$$

with

$$L_i(x) = \prod_{j=0, j\neq i}^{4} \frac{x - x_j}{x_i - x_j}, i = 0,1,2,3,4.$$

Furthermore $L_0(x) = L_3(x) = 0$ *because they will be multiplied by zero. Following the same procedure as in the previous example we obtain:*

$$\begin{aligned} L_1(x) &= \frac{(x-0)(x-2)(x-3)(x-4)}{(1-0)(1-2)(1-3)(1-4)} \\ &= -\frac{1}{6}(x^4 - 9x^3 + 26x^2 - 24x) \\ L_2(x) &= \frac{(x-0)(x-1)(x-3)(x-4)}{(2-0)(2-1)(2-3)(2-4)} \\ &= \frac{1}{4}(x^4 - 8x^3 + 19x^2 - 12x) \\ L_3(x) &= \frac{(x-0)(x-1)(x-2)(x-3)}{(4-0)(4-1)(4-2)(4-3)} \\ &= \frac{1}{24}(x^4 - 6x^3 + 11x^2 - 6x) \end{aligned}$$

Finally by substituting the polynomial coefficients, we obtain:

$$f(x) \simeq P(x) = 3.0416x^4 - 23.25x^3 + 50.9583x^2 - 25.75x^4.$$

5.2 Hermite Interpolation

Hermite interpolation generalizes Lagrange interpolation by ensuring that not only the function values $f(x)$ and $P(x)$ coincide at the nodes x_i, but also their first derivatives.

Let $x_0, x_1, \dots, x_n$ be $(n+1)$ distinct points in the interval $[a, b]$, and let f be a function defined on $[a, b]$ with known derivatives $f'(x_0), f'(x_1), \dots, f'(x_n)$. Then, there exists a **unique** polynomial $P(x)$ of degree at most $2n+1$ satisfying the conditions:

$$P(x_i) = f(x_i) \quad \text{and} \quad P'(x_i) = f'(x_i) \quad \text{for all } i = 0, 1, \dots, n.$$

This polynomial $P(x)$ is given by:

$$P(x) = \sum_{i=0}^{n} H_i(x) f(x_i) + \sum_{i=0}^{n} K_i(x) f'(x_i),$$

where the **Hermite basis polynomials** $H_i(x)$ and $K_i(x)$ are defined as:

$$\begin{cases} H_i(x) &= \left[1 - 2(x - x_i) L_i'(x_i)\right] L_i^2(x), \\ K_i(x) &= (x - x_i) L_i^2(x), \end{cases}$$

with $L_i(x)$ is the **Lagrange basis polynomials**:

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Theorem 5.2. *(Error bound Hermite interpolation method)*
Let $f \in \mathcal{C}^{2n+2}[a, b]$ and $P(x)$ be the polynomial interpolation of f over the points $m_i(x_i, f(x_i))$, for $i = 1, \dots, n$. Then for all $x \in [a, b]$, there exists $\xi_x \in]\min_{1,\dots,n}\{x_i\}, \max_{1,\dots,n}\{x_i\}[$ such that the error $f(x) - P(x)$ satisfies

$$E(x) = \frac{\gamma_{n+1}^2(x)}{(2n+2)!} f^{(2n+2)}(\xi_x).$$

By setting $M_{2n+2} = \max\limits_{a \le x \le b} |f^{(2n+2)}(x)|$, *then we have*

$$E(x) \le \frac{\gamma_{n+1}^2(x)}{(2n+2)!} M_{2n+2}$$

Example 5.3. *Let's determine the Hermite polynomial which interpolates the function* $f(x) = \frac{1}{1+x^2}$ *at the points* $x_0 = 0$ *and* $x_1 = 5$.

The Hermite polynomial $P(x)$ *is written as*

$$P(x) = \sum_{i=0}^{1} H_i(x) f(x_i) + \sum_{i=0}^{1} K_i(x) f'(x_i)$$

where

$$\begin{cases} H_i(x) &= [1 - 2(x - x_i) L_i'(x_i)] L_i^2(x) \\ K_i(x) &= (x - x_i) L_i^2(x) \end{cases}$$

Let us compute the polynomials $L_i(x)$, $L_i'(x)$, $H_i(x)$ *and* $K_i(x)$, *knowing that the x-coordinates of the support points are* $x_0 = 0$ *and* $x_1 = 5$.

$$L_0(x) = \frac{(x - x_1)}{(x_0 - x_1)} = 1 - \frac{x}{5}$$
$$L_1(x) = \frac{(x - x_0)}{(x_1 - x_0)} = \frac{x}{5}$$

and

$$L_0'(x) = -\frac{1}{5}$$
$$L_1'(x) = \frac{1}{5}$$

$$\begin{aligned}
H_0(x) &= [1-2(x-x_0)L_0'(x_0)]L_0^2(x)\\
&= \left(1-2(x-0)\frac{-1}{5}\right)\left(1-\frac{x}{5}\right)^2\\
&= \frac{2}{125}x^3+\frac{3}{25}x^2+1.\\
H_1(x) &= [1-2(x-x_1)L_1'(x_1)]L_1^2(x)\\
&= \left(1-2(x-5)\frac{1}{5}\right)\left(\frac{x}{5}\right)^2\\
&= -\frac{2}{125}x^3+\frac{3}{25}x^2.
\end{aligned}$$

Furthermore, we have

$$\begin{aligned}
K_0(x) &= (x-x_0)L_0^2(x) = (x-0)\left(1-\frac{x}{5}\right)^2 = \frac{1}{25}x^3-\frac{2}{5}x^2+x.\\
K_1(x) &= (x-x_1)L_1^2(x) = (x-5)\left(\frac{x}{5}\right)^2 = \frac{1}{25}x^3-\frac{2}{5}x^2.
\end{aligned}$$

Then,

$$\begin{aligned}
P(x) &= \sum_{i=0}^{1} H_i(x)f(x_i)+\sum_{i=0}^{1} K_i(x)f'(x_i)\\
&= H_0(x)f(x_0)+H_1(x)f(x_1)+K_0(x)f'(x_0)+K_1(x)f'(x_1)\\
&= \left(\frac{2}{125}x^3+\frac{3}{25}x^2+1\right)+\frac{1}{26}\left(-\frac{2}{125}x^3+\frac{3}{25}x^2\right)\\
&\qquad -\frac{10}{26^2}\left(\frac{1}{25}x^3-\frac{2}{5}x^2\right).\\
&= \frac{10}{262}x^3-\frac{76}{262}x^2+1
\end{aligned}$$

5.3 Matlab codes

5.3.1 Pseudo-code of Lagrange interpolation method

```
% Lagrange Interpolation
clc; clear;

% Example data points
x=[0 2 3 5]; % known x-values
y=[-1 2 9 87];% corresponding f(x) values
n=length(x);
syms X; % symbolic variable for the
   polynomial
P=0;  % initialize the Lagrange polynomial
% Construct Lagrange interpolation
   polynomial
for i=1:n
    L=1;
    for j=1:n
        if j ~= i
            L=L*(X-x(j))/(x(i)-x(j));
        end
    end
    P=P+y(i)*L;
end
P=expand(P);  % expand the polynomial for
   readability
disp('The Lagrange interpolation
   polynomial P(x) is:');
pretty(P)
```

5.3.2 Pseudo-code of Hermite interpolation method

```
% Hermite Interpolation
% This constructs the Hermite
   interpolating polynomial H(x)
clc;clear;
% Example data points
x=[1,2,9,28]; % known x-values
f= [0,1,2,3]; % f(x) values at those
   points
df=[0.5,0.5,0.5,0.5];   % f'(x) values at
   those points
n=length(x);
syms X;
H=0;   % Hermite polynomial initialization
for i=1:n
% Construct the Lagrange basis polynomial
   L_i(x)
    Li=1;
    for j=1:n
        if j~=i
            Li=Li*(X-x(j))/(x(i)-x(j));
        end
    end
    Li=expand(Li);
    % Compute derivative of Li
    dLi=diff(Li, X);
    % Construct H_i(x) and K_i(x)
    Hi=(1-2*(X-x(i))*dLi)*(Li)^2;
    Ki=(X-x(i))*(Li)^2;
    % Add to Hermite polynomial
    H=H+f(i)*Hi+df(i)*Ki;
```

```
end
H=expand(H);
disp('The Hermite interpolation polynomial
    H(x) is:');
pretty(H)
```

5.4 Solved Exercises

Exercise 5.1. *Suppose that* $f(x) = \sqrt[3]{x}$ *and* $(x_0, f(x_0)) = (0,0), (x_1, f(x_1)) = (1,1)$ *and* $(x_2, f(x_2)) = (8,2)$.
1) Determine the polynomial interpolation polynomial $P_2(x)$ *that passes through the points* $(x_i, y_i)_{i=0,1,2}$.
2) Calculate $P_2(x)$ *and* $f(x) = \sqrt[3]{x}$ *for* $x = 0.5, 0.95, 1, 1.5$ *and* 3.

Solution
1- Following the Lagrange method,

$$\begin{aligned}
P_2(x) =& f(x_0)L_0(x) + f(x_1)L_1(x) + f(x_2)L_2(x)\\
=& f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} +\\
& + f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}\\
=& 0\frac{(x-1)(x-8)}{(0-1)(0-8)} + 1\frac{(x-0)(x-8)}{(1-0)(1-8)} + 2\frac{(x-0)(x-1)}{(8-0)(8-1)}\\
=& -\frac{3}{28}x^2 + \frac{31}{28}x.
\end{aligned}$$

Then, $P_2(0) = 0$, $P_2(1) = 1$ and $P_2(8) = 2$.
2-

x_i	0.5	0.95	1	1.5	3
$f(x_i)$	0.7937	0.98305	1	1.1447	1.4422
$P_2(x)$	0.52679	0.95509	1	1.4196	2.3571

Exercise 5.2. *Consider the function* $f(x) = 1/x$.
1- Find the Lagrange interpolation polynomial that passes through the points $(2, 0.5), (2.5, 0.4)$ *and* $(4, 0.25)$.
2- Calculate the approximation of $f(2.2)$.
3- Estimate the maximum error.

Solution
1- $P_2(x) = \frac{1}{20}x^2 - \frac{17}{40}x + \frac{23}{20}$.
2- We deduce that

$$f(2.2) \simeq P_2(2.2) = 0.457$$

3- The error of the polynomial $P_2(x)$: From Theorem 5.1, there exists $\xi_x \in [2, 4]$ such that;

$$E(x) = \frac{f^{(3)}(\xi_x)}{3!}(x - x_0)(x - x_1)(x - x_2)$$

then,

$$E(x) = \frac{f^{(3)}(\xi_x)}{3!}(x - 2)(x - 2.5)(x - 4).$$

Let $M_3 = \max\limits_{2 \le x \le 4} |f^{(3)}(x)|$, then

$$E(x) \le \frac{M_3}{3!}|(x - 2)(x - 2.5)(x - 4)|.$$

On the other hand $f^{(3)}(x) = -\frac{6}{x^4}$ et $\max\limits_{2 \le x \le 4} |f^{(3)}(x)| = \dfrac{3}{8}$. Hence

$$E(x) \le \frac{3}{48}|(x - 2)(x - 2.5)(x - 4)|.$$

For $x = 2.2$, the error is bounded as follows:

$$E(x) \le \frac{3}{48}|(2.2 - 2)(2.2 - 2.5)(2.2 - 4)| \simeq 0.0068$$

Exercise 5.3. *Consider the following points:* $(0,0),(1,2),(2,36)$, $(3,252)$ *and* $(4,104)$.

1. *Determine the Lagrange polynomial passing through the first three points.*
2. *Determine the Lagrange polynomial passing through the first four points.*
3. *Provide the analytical expression of the error for the polynomials obtained in (1) and (2).*
4. *Determine approximations of* $f(1.5)$ *using the two polynomials obtained in (1) and (2).*

Solution

1- $P_2(x) = 16x^2 - 14x$.

2- $P_3(x) = x(x-2)(x-3) - 18x(x-1)(x-3) + 42x(x-1)(x-2) = 61x^3 - 203x^2 + 144x$. 3- The error of the polynomial $P_2(x)$:From Theorem 5.1, there exists $\xi_x \in [0,2]$ such that:

$$E_2(x) = \frac{f^{(3)}(\xi_x)}{3!}(x-x_0)(x-x_1)(x-x_2)$$

then,

$$E_2(x) = \frac{f^{(3)}(\xi_x)}{3!}(x-0)(x-1)(x-2)$$

whereas for $P_3(x)$, there exists $\xi_x \in [0,4]$, where the error is given by:

$$E_3(x) = \frac{f^{(4)}(\xi_x)}{4!}(x-0)(x-1)(x-2)(x-3)$$

4- If we approximate f using P_2, we have $f(1.5) \approx P_2(1.5) = 15$, and if we approximate f using P_3, we obtain $f(1.5) \approx P_3(1.5) = 5.625$.

5.4.1 Supplementary Exercises

Exercise 5.4. *We want to design a railway track curve between the points* $(0, 0)$ *and* $(1, 1)$. *The curve is described by a function of the form* $y = f(x)$ *that satisfies:*

$$f(0) = 0 \text{ and } f(1) = 1.$$

Moreover, to ensure a smooth transition, the slope of the curve must satisfy:

$$f'(0) = 0 \quad \text{and} \quad f'(1) = 0.3.$$

The curve is represented by a polynomial on the interval $[0, 1]$.
- Construct, using the Hermite method, the interpolation polynomial $P(x)$ *that interpolates these points.*

Result
The polynomial obtained following the Hermite method is

$$P(x) = -1.7x^3 + 2.7x^2$$

Exercise 5.5. *Consider the following points:* $(0, 0), (1, 2), (2, 36)$ *and* $(3, 252)$.

1. *Determine the Lagrange polynomial passing through the first three points.*

2. *Determine the Lagrange polynomial passing through the four points.*

Result
1- $P_2(x) = 16x^2 - 14x$.
2- $P_3(x) = 25x^3 - 59x^2 + 36x$

Exercise 5.6. *Determine the Lagrange polynomial passing through the points* $(1, 0)$, $(2, 1)$, $(9, 2)$ *and* $(28, 3)$.

Result

- $P_3(x) = x^3 + 1.$

Bibliography

[1] Allaoua, B. (2011). Méthodes numériques avec Matlab: Rappels de cours, corrigés détaillés et applications avec l'environnement Matlab. Editions Universitaires Européennes (EUE).

[2] Altaç, Z. (2024). Numerical Methods for Scientists and Engineers: With Pseudocodes. CRC Press.

[3] Belkacem, M. (2016), Cours de Méthodes Numériques, Editions universitaires européennes EUE, Germany.

[4] Beu, T. A. (2014). Introduction to numerical programming: a practical guide for scientists and engineers using Python and C/C++. CRC Press.

[5] Datta, N. (2003). Computer Programming and Numerical Analysis Revised Edition with C: A Integrated Approach. Universities Press.

[6] Epperson, J. F. (2013). An introduction to numerical methods and analysis. John Wiley & Sons.

[7] Grivet, J. P. (2012). Méthodes numériques appliquées pour le scientifique et l'ingénieur (édition 2009): Edition 2013. EDP sciences.

[8] Otto, S., Denier, J. P. (2005). An introduction to programming and numerical methods in MATLAB. Springer Science & Business Media.

[9] Ray, S. S. (2018). Numerical analysis with algorithms and programming. Chapman and Hall/CRC.

[10] Rice, J. R. (2014). Numerical methods in software and analysis. Elsevier.

Numerous problems cannot be solved using traditional analytical methods; this is why numerical methods have emerged. In many cases, approximating a solution depends on the number of operations to be performed, which presents a challenge to the application of these numerical methods.
The emergence of computers and the expansion of computing have greatly facilitated the use of numerical methods, thanks to the development of algorithms implemented on machines with powerful processors.
Today, technology continues to advance, constantly bringing new developments across various fields. Scientific research has progressed significantly, enabling the understanding and modeling of physical phenomena that were unclear just a few years ago. This progress has been made possible through numerical analysis.
In this booklet, we present the numerical methods essential for second-year bachelor's (LMD) students (Physics/Chemistry) to address many of the challenges they encounter throughout their academic studies.
It is worth noting that our approach is based on two essential points:

1. The course content must be simplified.

2. The acquired knowledge is reinforced through simple examples.

Finally, I hope that readers will find at least something useful in this booklet.